Apprenez à créer votre site web avec HTML5 et CSS3

Par Mathieu Nebra (M@teo21)

le Site du Zéro
www.siteduzero.com

Licence Creative Commons BY-NC-SA 2.0
Dernière mise à jour le 2/03/2012

Sommaire

Apprenez à créer votre site web avec HTML5 et CSS3

Par  Mathieu Nebra (M@teo21)

Mise à jour : 01/11/2011
Difficulté : Facile Durée d'étude : 20 jours
BY-NC-SA
262 346 visites depuis 7 jours, classé 1/779

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

Ce cours vous plaît ?

Si vous avez aimé ce cours, vous pouvez retrouver le livre "*Réalisez votre site web avec HTML5 et CSS3*" du même auteur, en vente sur le Site du Zéro, en librairie et dans les boutiques en ligne. Vous y trouverez ce cours adapté au format papier avec une série de chapitres inédits.

Vous pouvez également obtenir cet ouvrage au format eBook sur Amazon ou sur iTunes.

Plus d'informations

Partie 1 : Les bases de HTML5

Vous n'avez jamais entendu parler du HTML, ou alors seulement de façon très vague ?
Pas de panique, les explications arrivent dès le premier chapitre... et la pratique suit juste après !

Nous commencerons par présenter comment les sites web fonctionnent, puis nous téléchargerons tous les programmes (gratuits) nécessaires pour bien travailler.
A la fin de cette partie, vous saurez déjà insérer du texte, des liens et des images !

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...

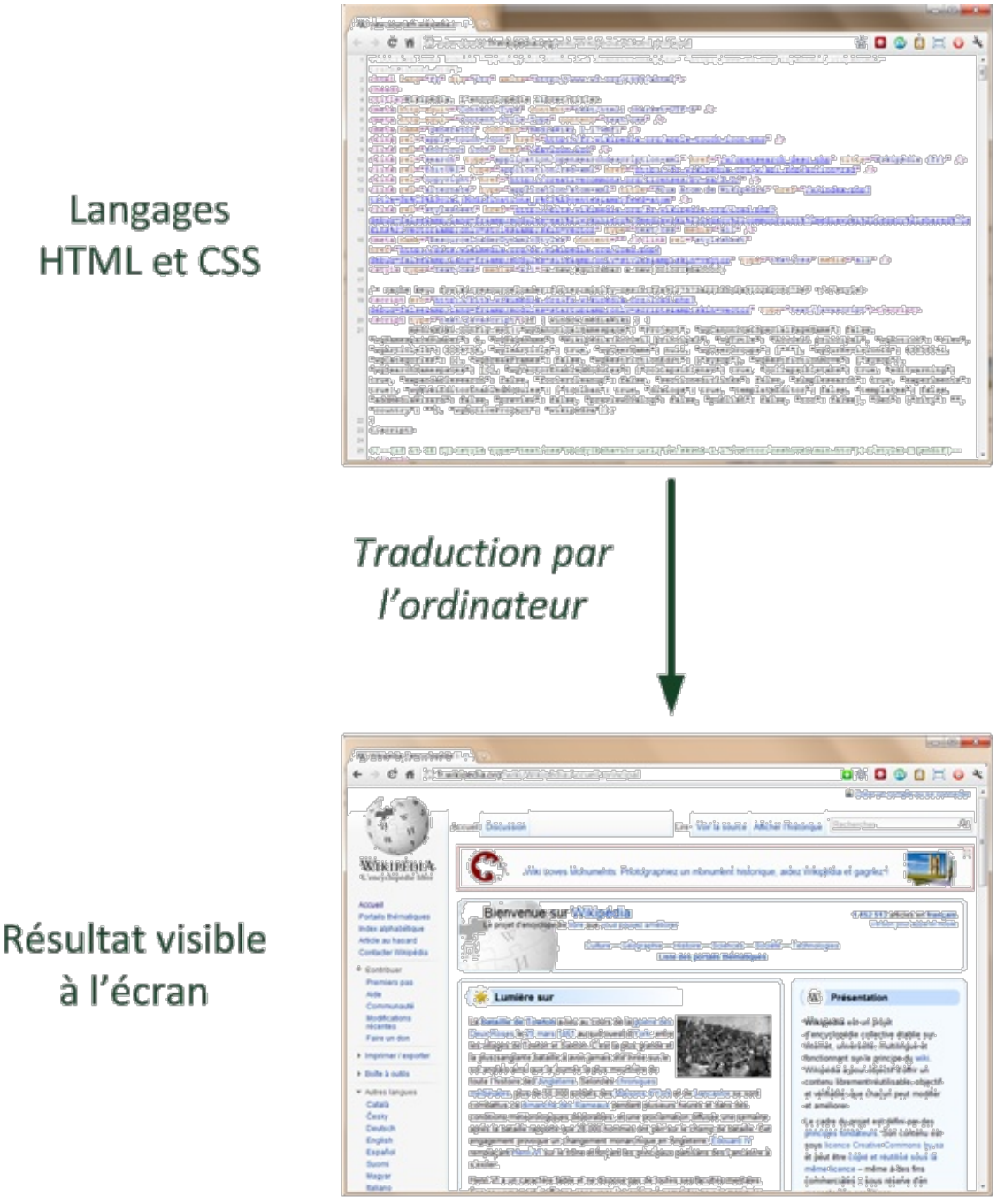
Tim Berners-Lee suit encore aujourd'hui avec attention l'évolution du Web. Il a créé le World Wide Web Consortium (W3C) qui définit les nouvelles versions des langages. Il a par ailleurs créé plus récemment la World Wide Web Foundation, qui analyse et suit l'évolution du Web.


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 :



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 :

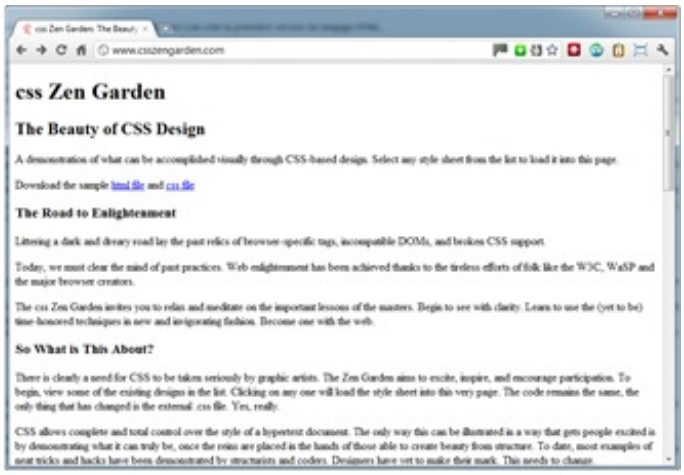
- **HTML** (*HyperText Markup Language*) : il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organiser le contenu. C'est donc en HTML que vous écrirez ce que vous souhaitez que la page affiche : du texte, des liens, des images... Vous direz par exemple : "Ceci est mon 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 versions du HTML. Les règles et le fonctionnement de cette version sont donnés 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 débourser 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 apprendre de langage particulier. 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 !

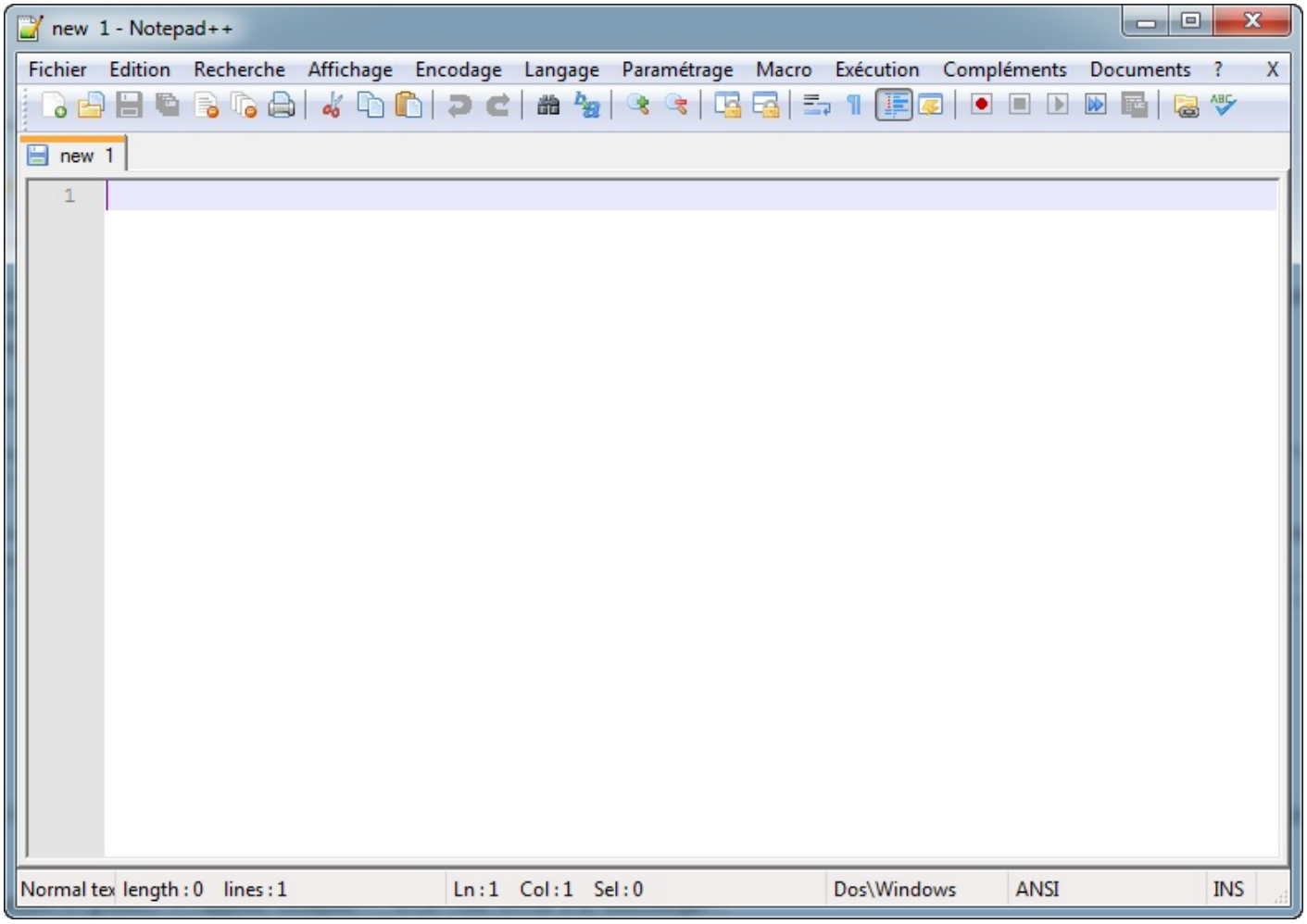
Vous l'aurez compris, je vais vous inviter à utiliser un éditeur de texte dans ce cours. Voici quelques conseils, selon que vous êtes sous Windows, Mac OS X ou Linux.

Sous Windows

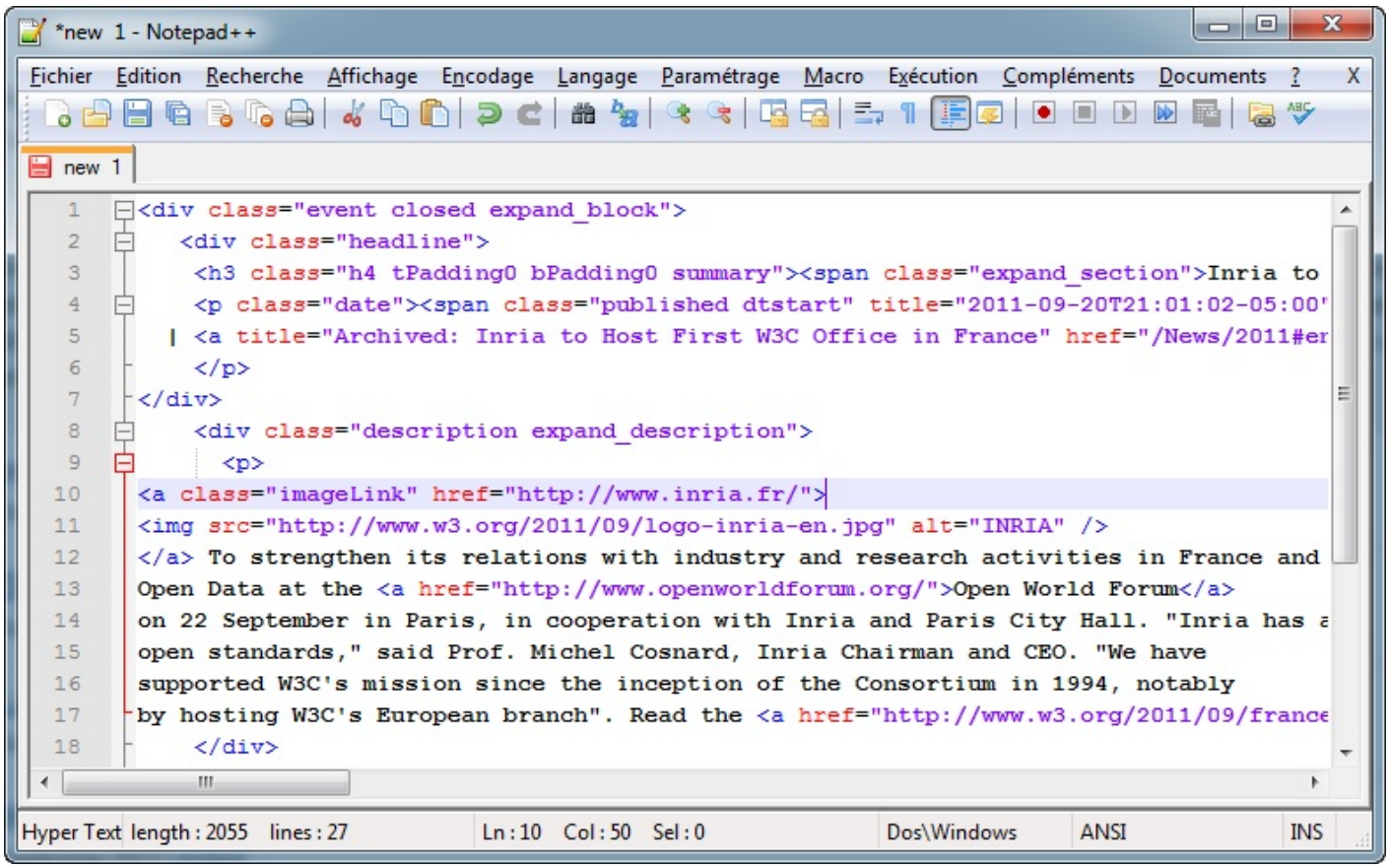
Il existe une grande quantité d'éditeurs de texte, je ne pourrai pas tous vous les présenter. Néanmoins, je vous invite à vous pencher sur Notepad++, l'un des plus utilisés d'entre eux sous Windows. Ce logiciel est simple, en français et gratuit.

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 :



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 rendrez 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

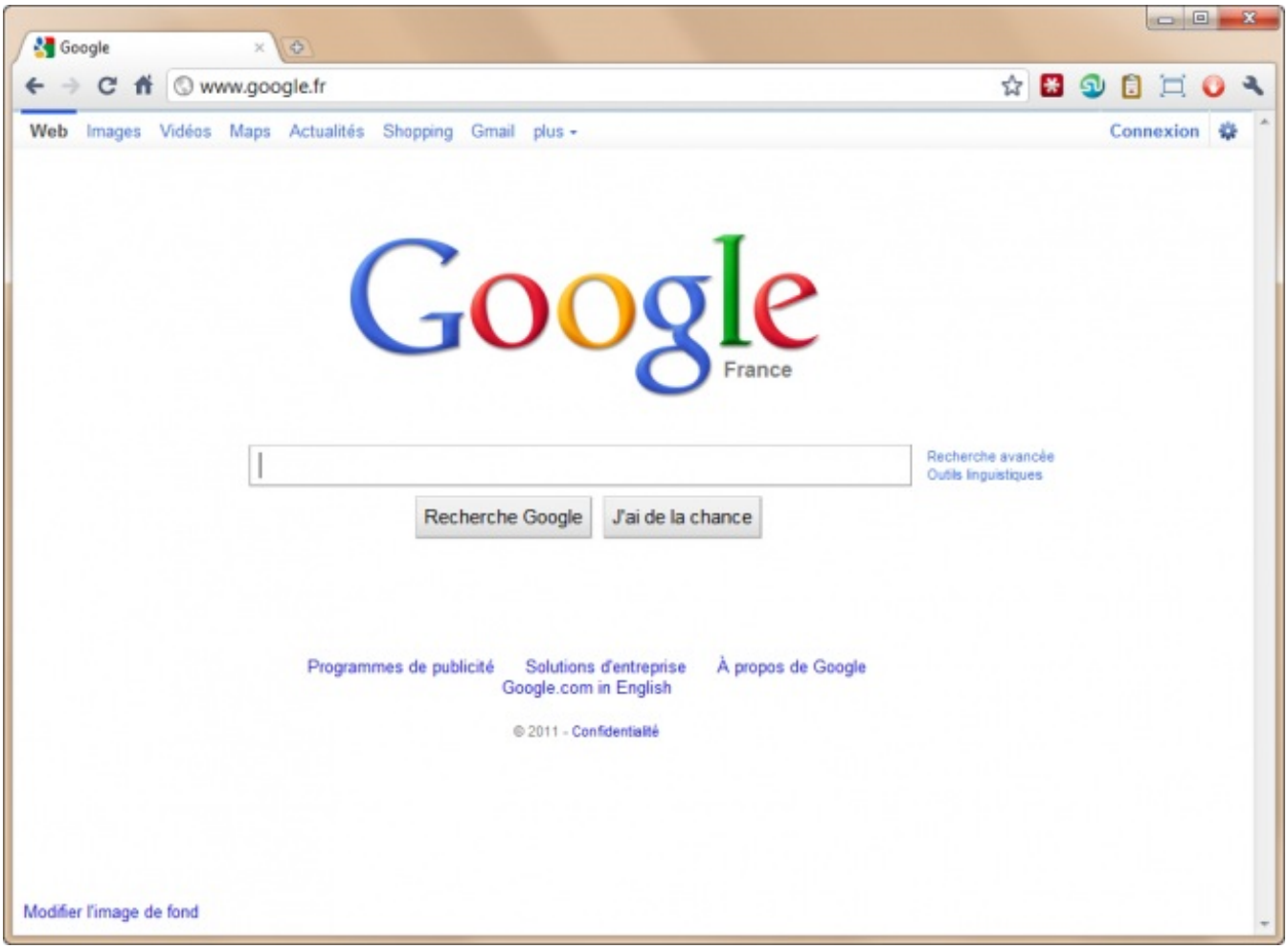
Téléchargez les navigateurs

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

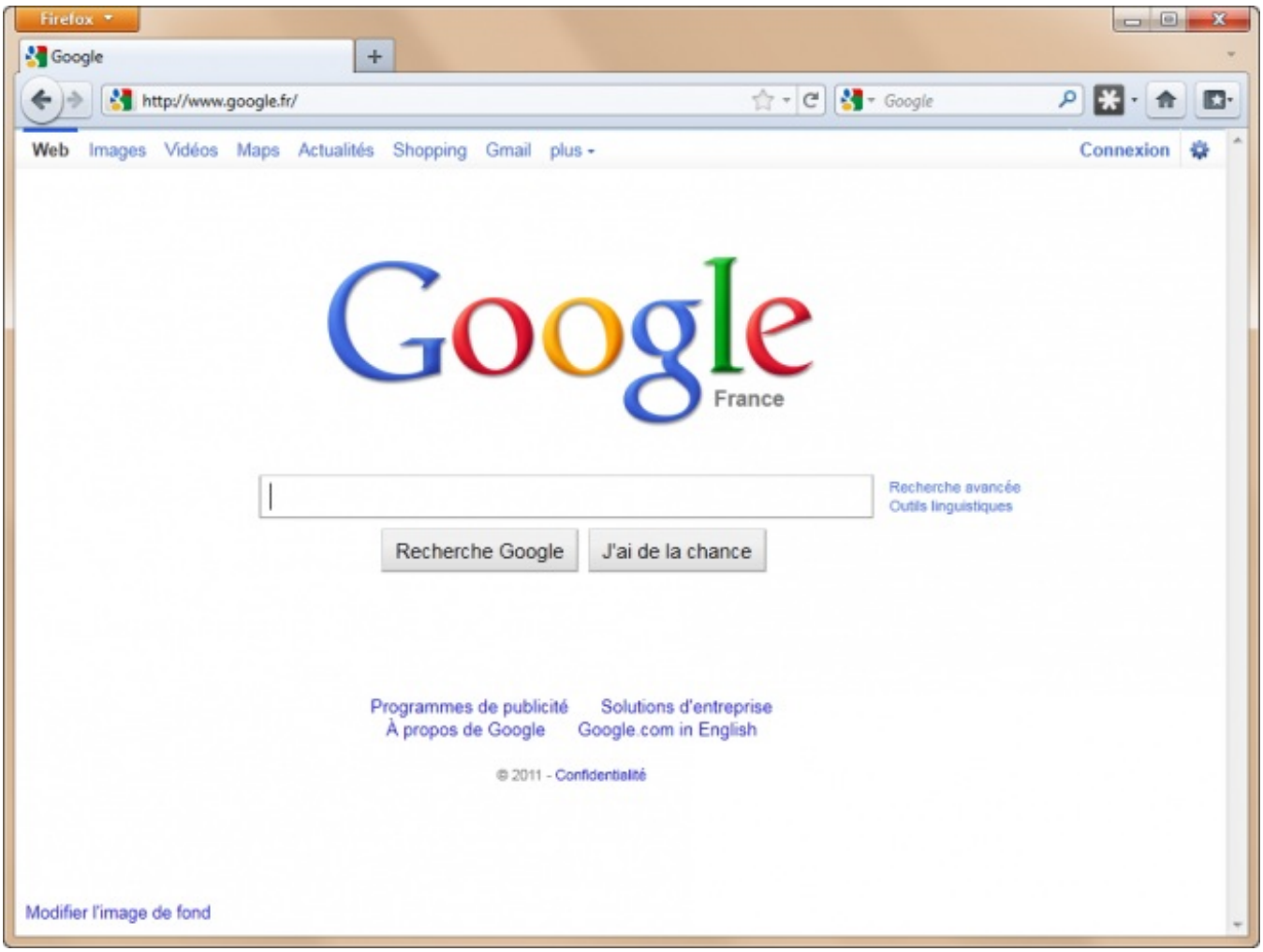
Navigateur	OS	Téléchargement	Commentaires
Google Chrome	Windows Mac Linux	Téléchargement	Le navigateur de Google, simple d'emploi et très rapide. **C'est le navigateur que j'utilise au quotidien.**
Mozilla Firefox	Windows Mac Linux	Téléchargement	Le navigateur de la fondation Mozilla, célèbre et réputé. Je l'utilise fréquemment pour tester mes sites web.
Internet Explorer	Windows	Téléchargement (Déjà installé sur Windows)	Le navigateur de Microsoft, qui équipe tous les PC Windows. Je l'utilise fréquemment pour tester mes sites web.
Safari	Windows Mac	Téléchargement (Déjà installé sur Mac OS X)	Le navigateur d'Apple, qui équipe tous les Mac.
Opera	Windows Mac Linux	Téléchargement	L'éternel *outsider*. 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 :



Télécharger Google Chrome
(Windows, Mac OS X et Linux)



Télécharger Firefox
(Windows, Mac OS X et Linux)

Télécharger Internet Explorer
(Windows uniquement, mise à jour vers la dernière version)

Vous remarquerez qu'ils se ressemblent tous étrangement ! En revanche, sous le capot, des différences (parfois importantes) subsistent dans ces navigateurs...

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 et caniuse.com 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 dépaysés : la plupart des navigateurs sur smartphones sont les mêmes que sur ordinateur, dans une version plus légère adaptée aux mobiles. Tout dépend du type de téléphone.

- **iPhone** : sur l'iPhone d'Apple, le navigateur utilisé est Safari Mobile. Il s'agit d'une version *light* 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

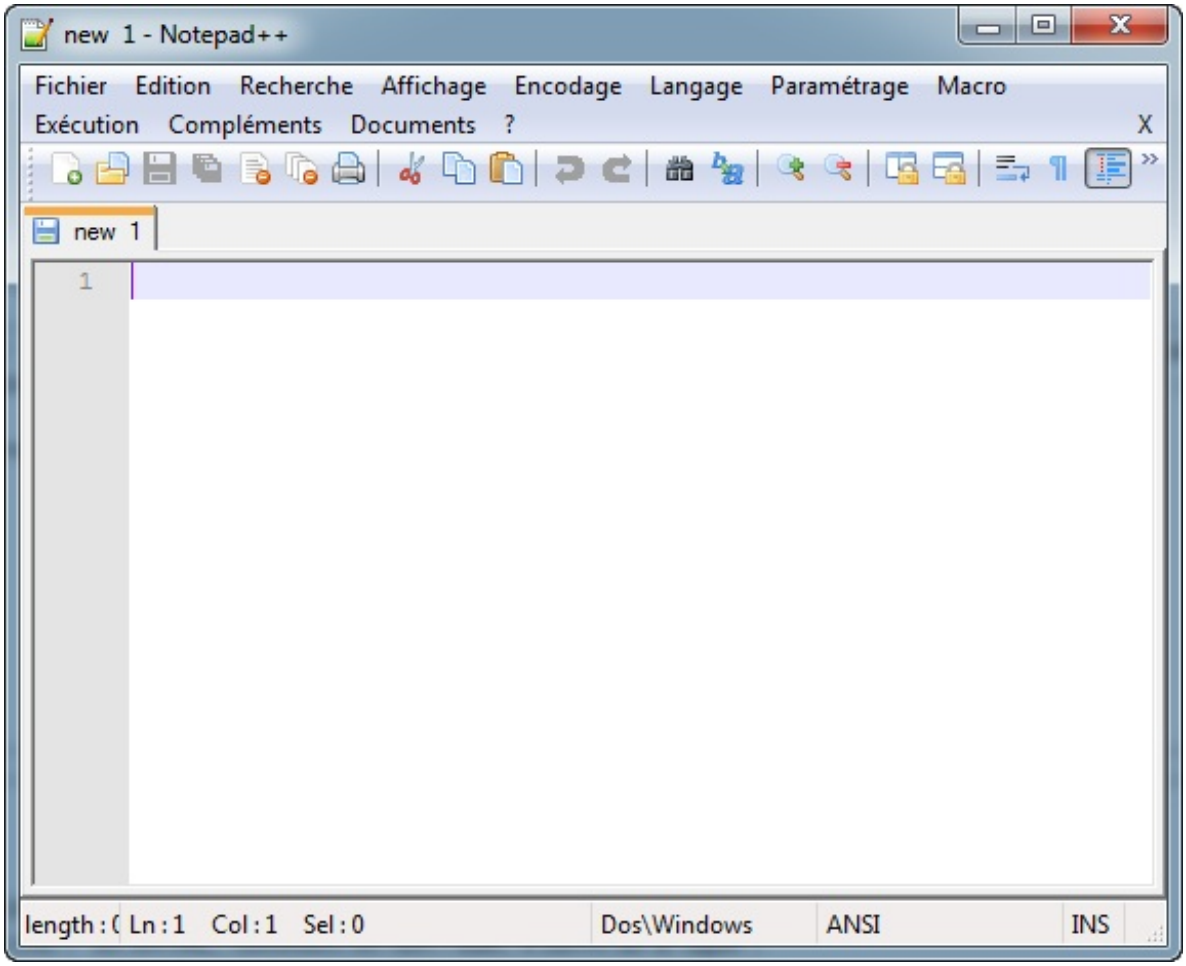
Ça y est, vous avez installé tous les logiciels ?
Vous devriez maintenant avoir un éditeur de texte pour *créer votre site* (comme Notepad++) et plusieurs navigateurs pour le *tester* (Mozilla Firefox, Google Chrome, Internet Explorer...).

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

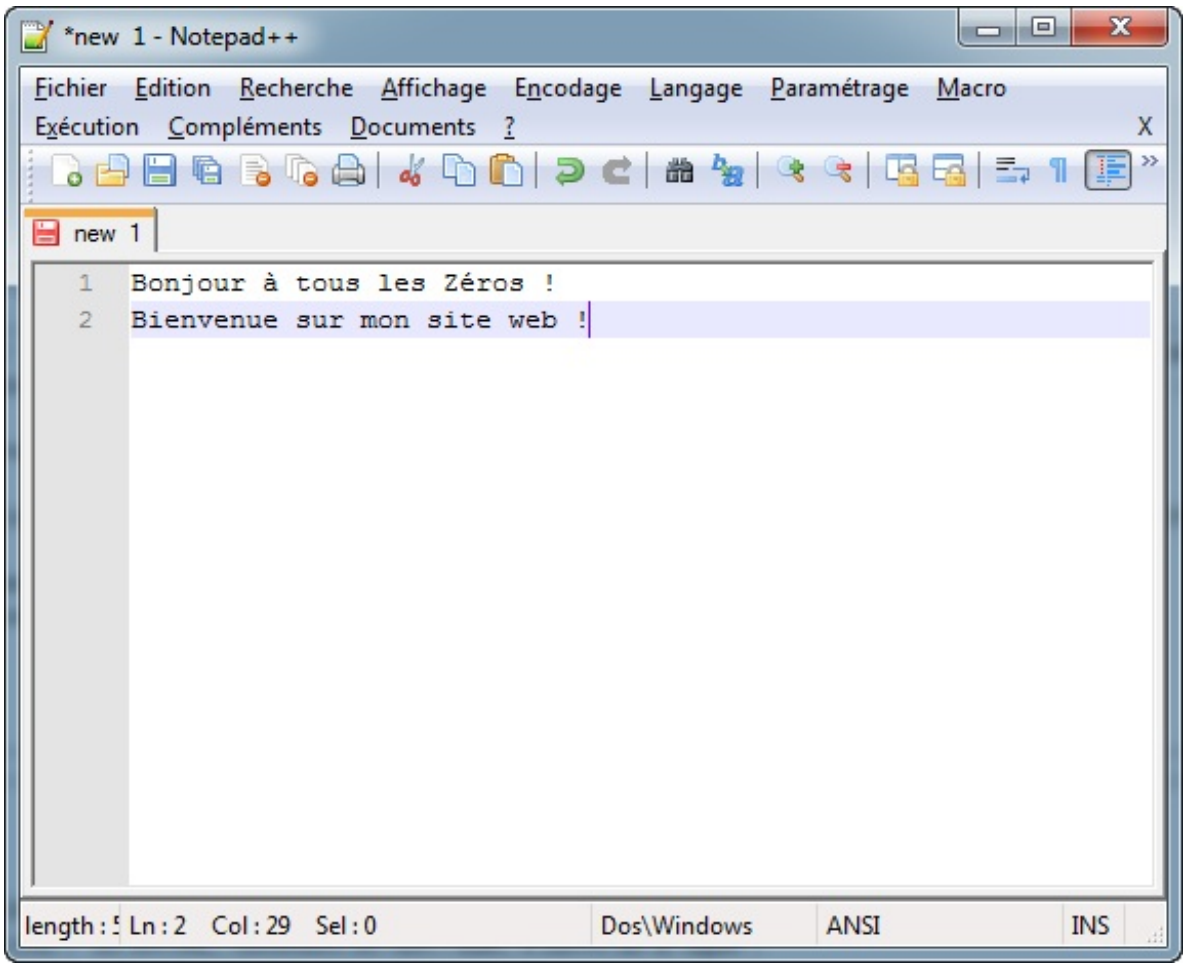
Allez, mettons-nous en situation ! Comme je vous l'ai dit, nous allons créer notre site dans un éditeur de texte. Vous avez dû en installer suite à mes conseils dans le premier chapitre : qu'il s'appelle Notepad++, PSpad, jEdit, vim, TextWrangler... peu importe. Ces logiciels ont un but très simple :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

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



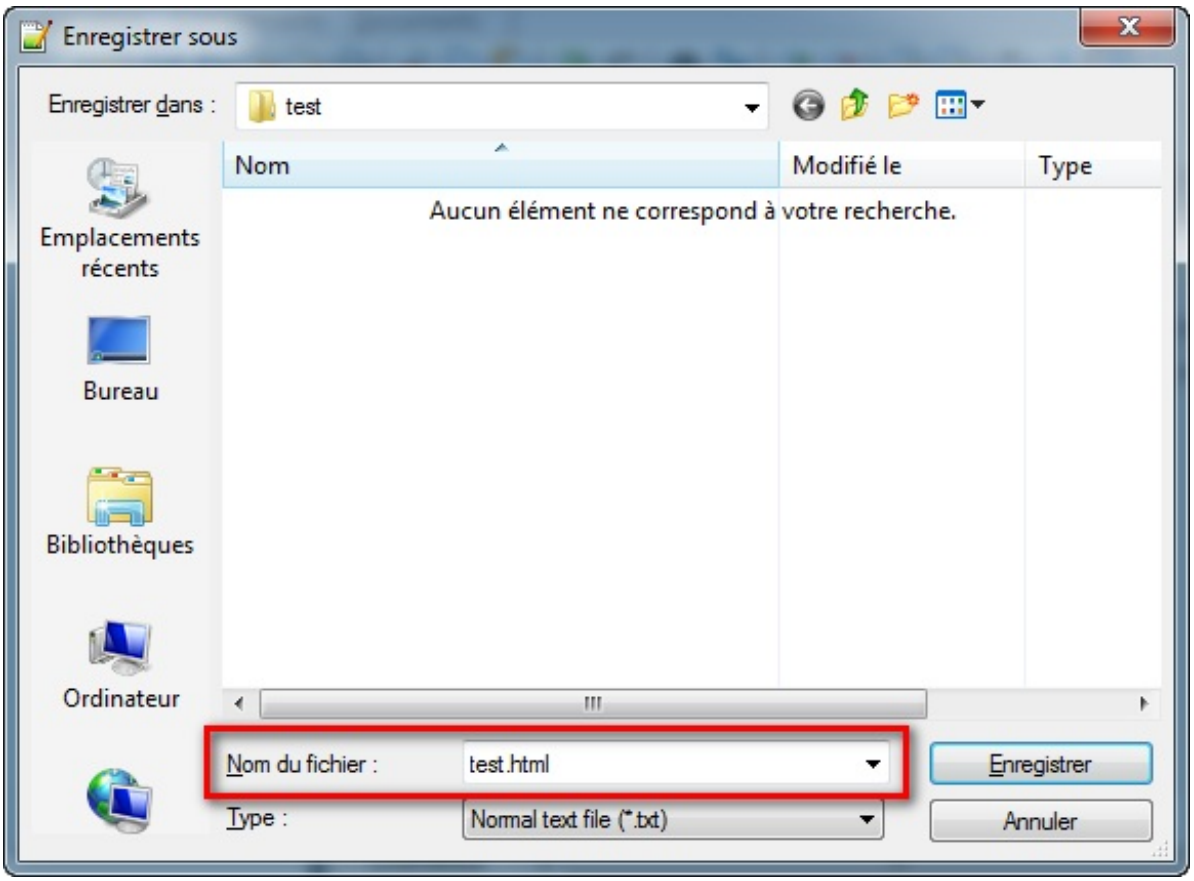
Bon, qu'est-ce qu'on fait maintenant ? Qu'est-ce qu'on écrit sur cette feuille blanche ?

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



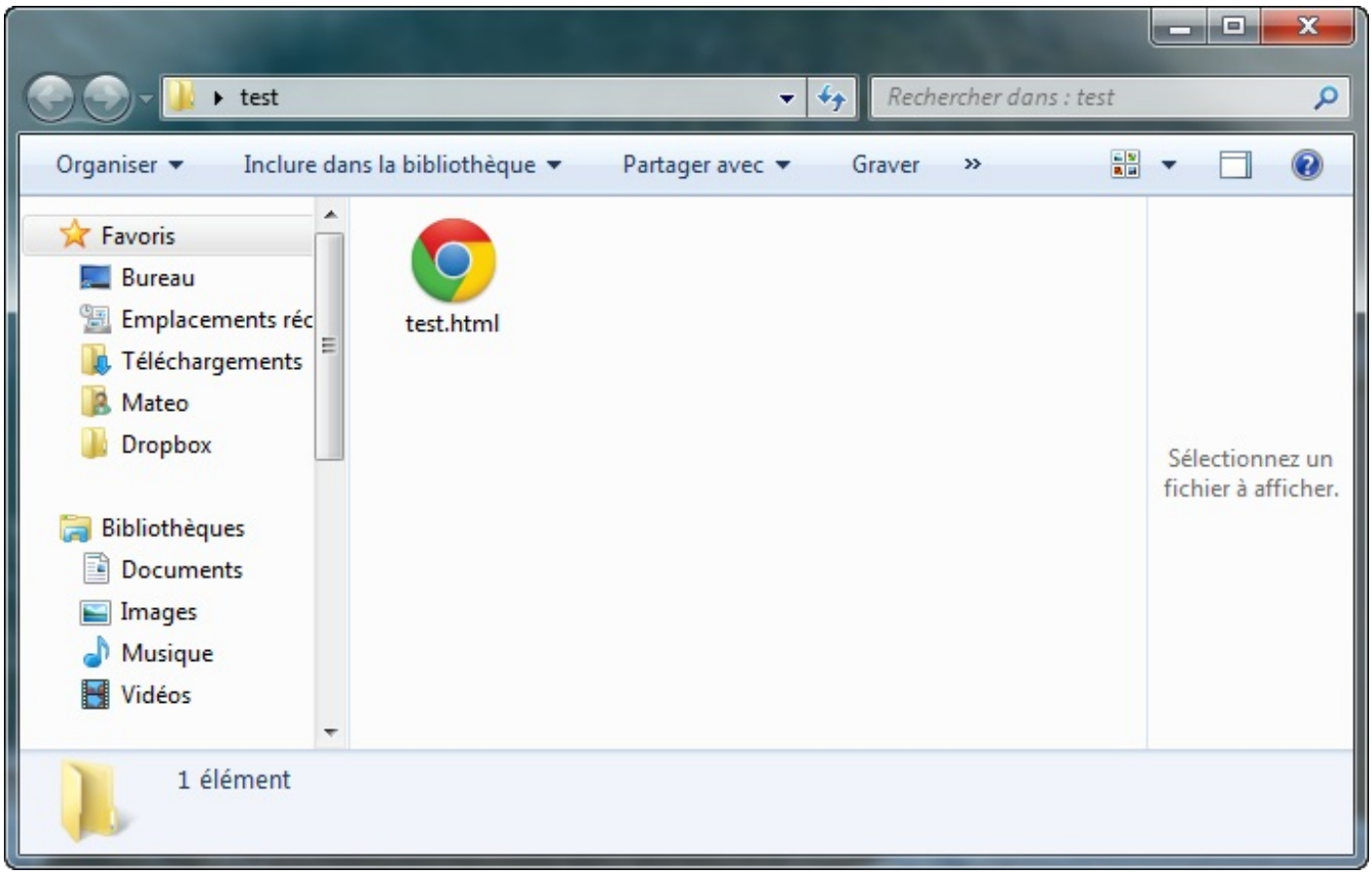
Vous pouvez écrire les mêmes phrases que moi ou ce que vous voulez ; le but est d'écrire quelque chose.

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`.

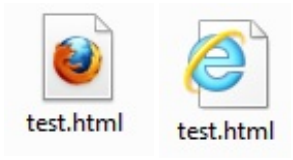


Je vous recommande de créer un nouveau dossier dans vos documents qui contiendra les fichiers de votre site. Pour ma part j'ai créé un dossier `test` dans lequel j'ai mis mon fichier `test.html`.

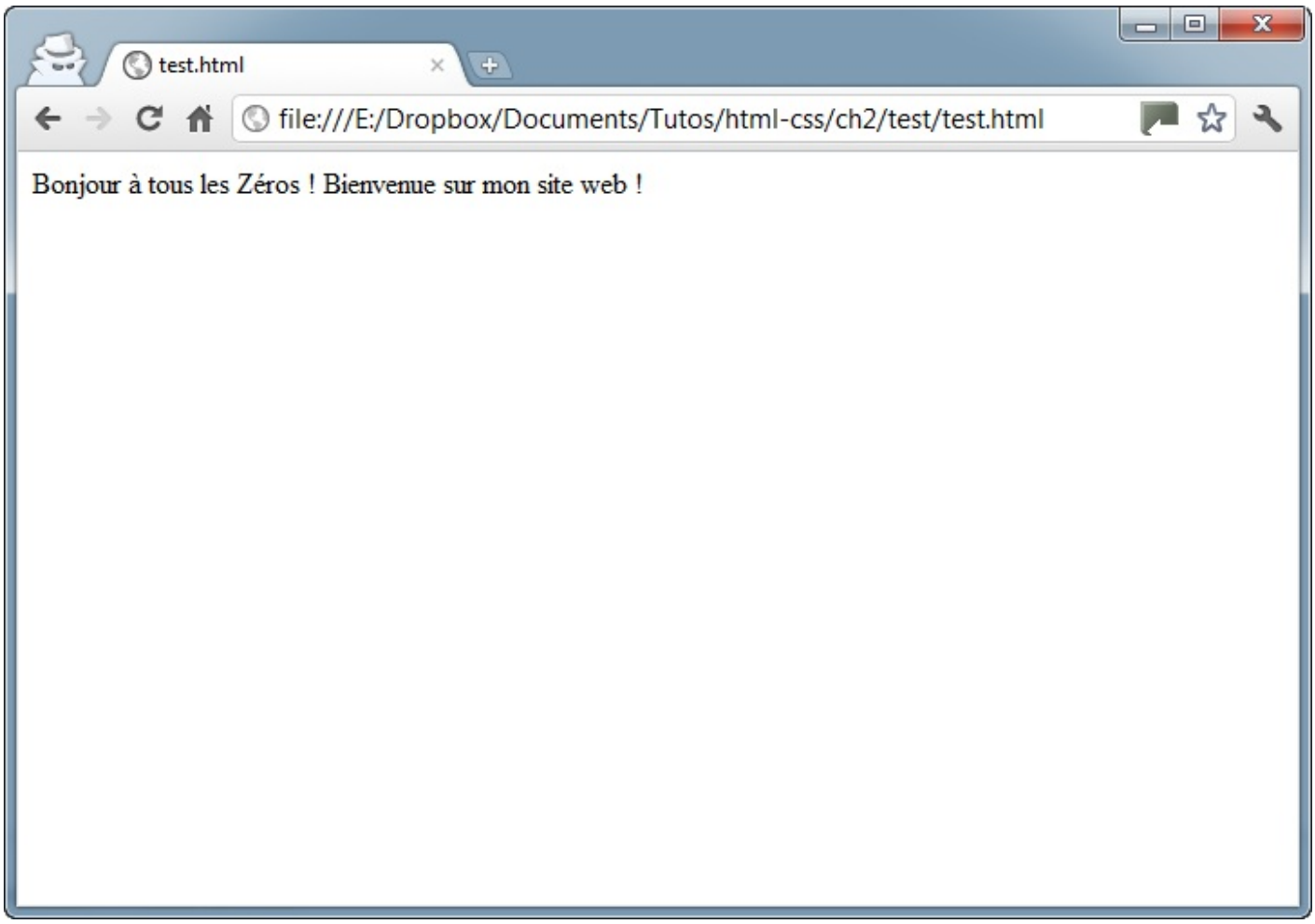
Ouvrez maintenant l'explorateur de fichiers dans le dossier où vous avez enregistré votre page. Vous y verrez le fichier que vous venez de créer :



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 me verrez 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

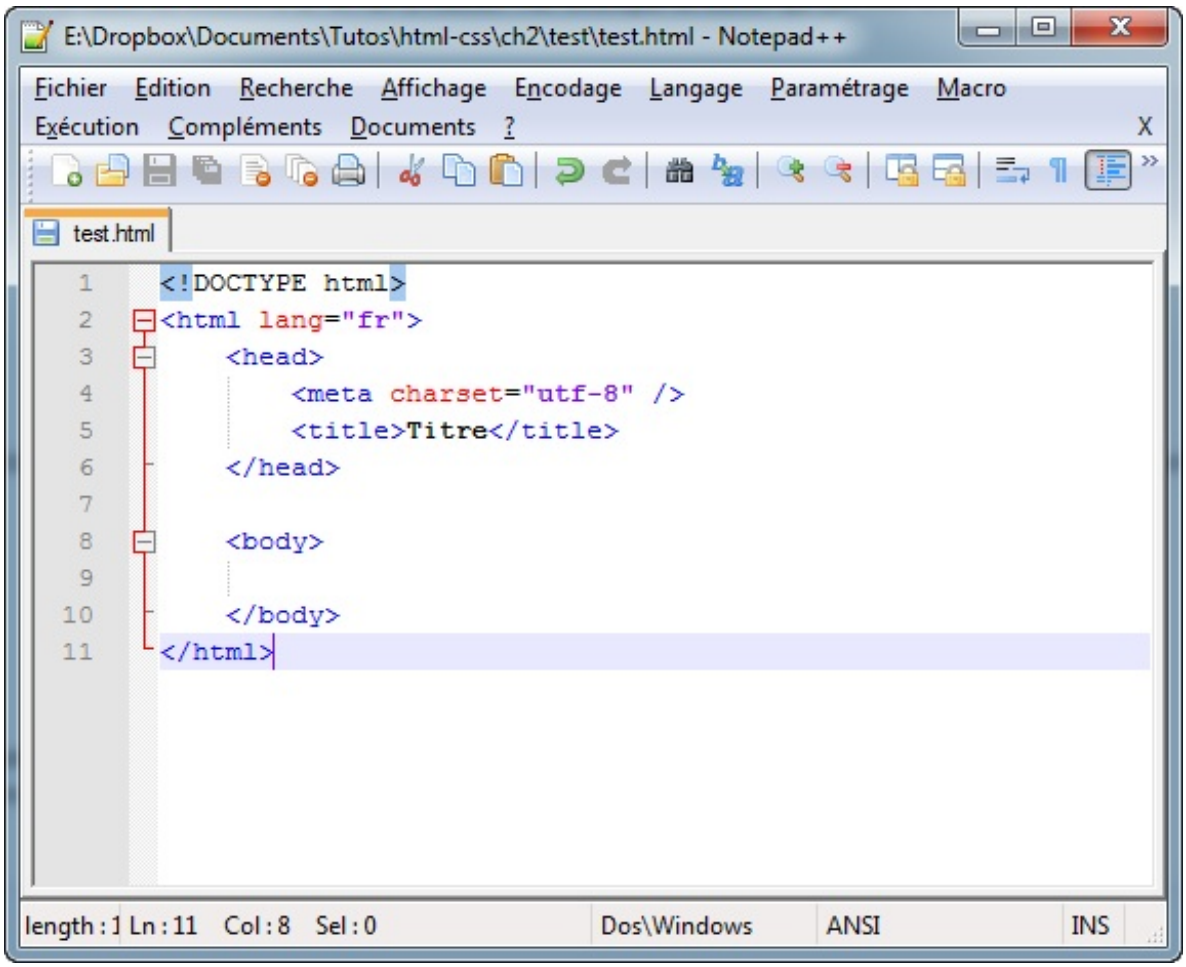
Reprenons notre éditeur de texte (dans mon cas Notepad++). Je vous invite à copier-coller le code source ci-dessous dans Notepad++. Ce code correspond à la base d'une page web en HTML5 :

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 :



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 écrirons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrirons la majeure partie de notre code.

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

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'encodage 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`).

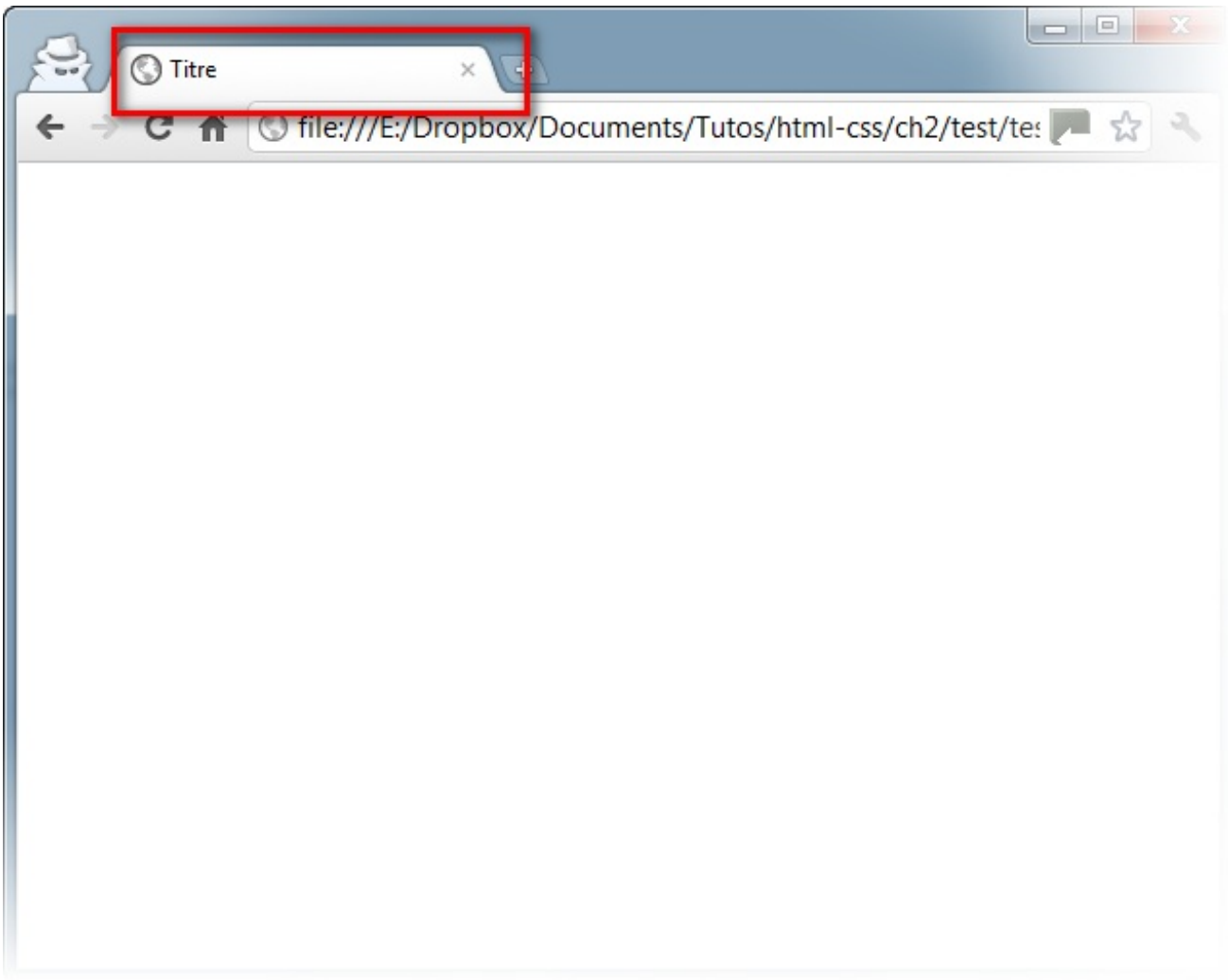
Le titre principal de la page

Code : HTML

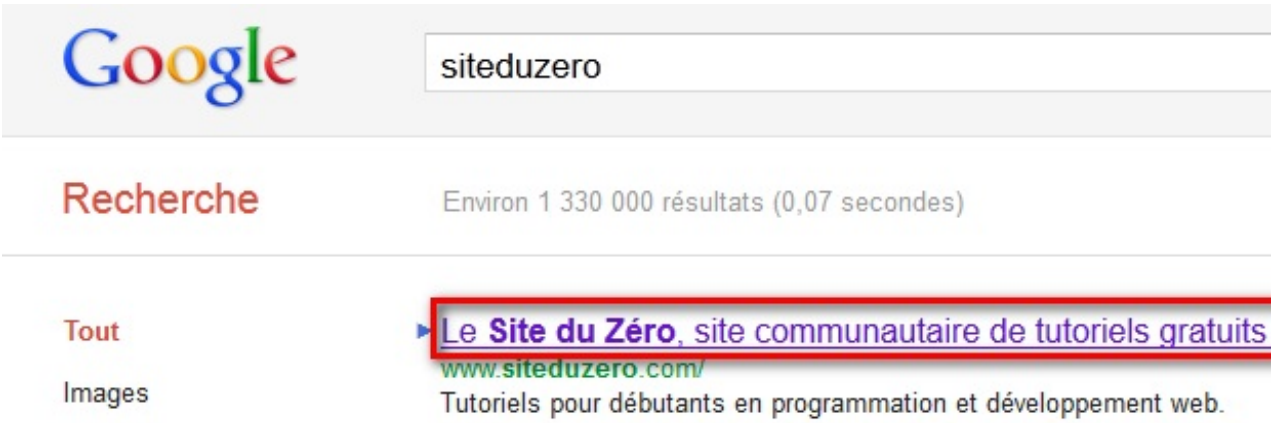
```
<title>
```

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 -->
```

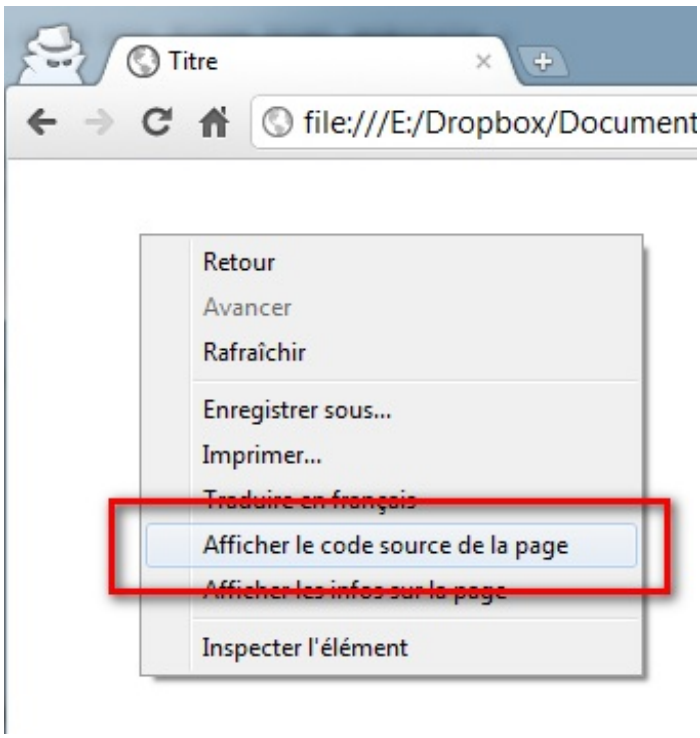
Vous pouvez le mettre où vous voulez au sein de votre code source : il n'a aucun impact sur votre page, mais vous pouvez vous en servir pour vous aider à vous repérer dans votre code source (surtout s'il est long).

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 ** ** pour délimiter les paragraphes.

Code : HTML

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

- ** ** signifie "Début du paragraphe"
- ** ** 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>Paragraphes</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 : **
. 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 **
** 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 ?

- ** ** : pour organiser son texte en paragraphes.
- **
** : 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>
</body>
</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 !<br />

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

Essayer !

Comme vous pouvez le voir, utiliser la balise **** 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 **** qui signifie "fort", ou "important" si vous préférez. Elle s'utilise exactement de la même manière que **** :

Code : HTML

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

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

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

        Ceci est mon premier test, alors <mark>soyez
indulgents</mark> s'il vous plaît, j'apprends petit à petit comment
ça marche.
    </p>
</body>
</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 ****, ****, **<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 **** 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, **** 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 **** 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 **
**.
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 **
** par **
**.
À l'intérieur de la liste, on ne change rien : on utilise toujours des balises **
** pour délimiter les éléments.

L'ordre dans lequel vous mettez les éléments de la liste est important. Le premier **
** 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.

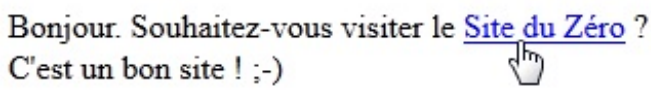
Comme vous le savez, un site web est composé de plusieurs pages. Comment faire pour aller d'une page vers une autre ? À l'aide de liens pardi ! Dans ce chapitre, nous allons justement apprendre à créer des liens entre nos pages.

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 :



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 & dans votre lien comme ceci :
`http://www.site.com/?data=15&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

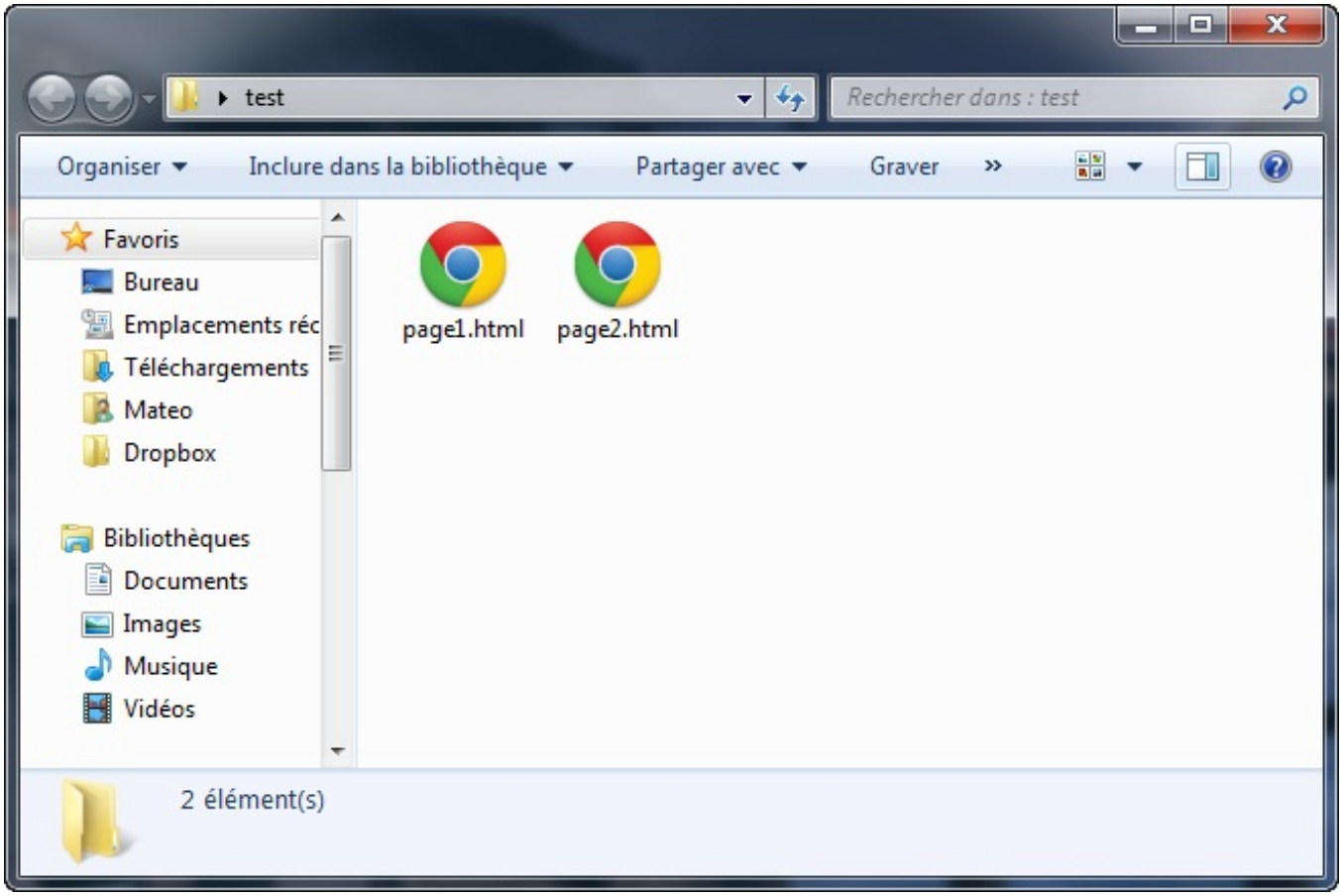
Nous venons d'apprendre à créer des liens vers des sites déjà existants. Mais je suis sûr que vous aimeriez faire des liens entre les différentes pages de votre site, non ?

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 !

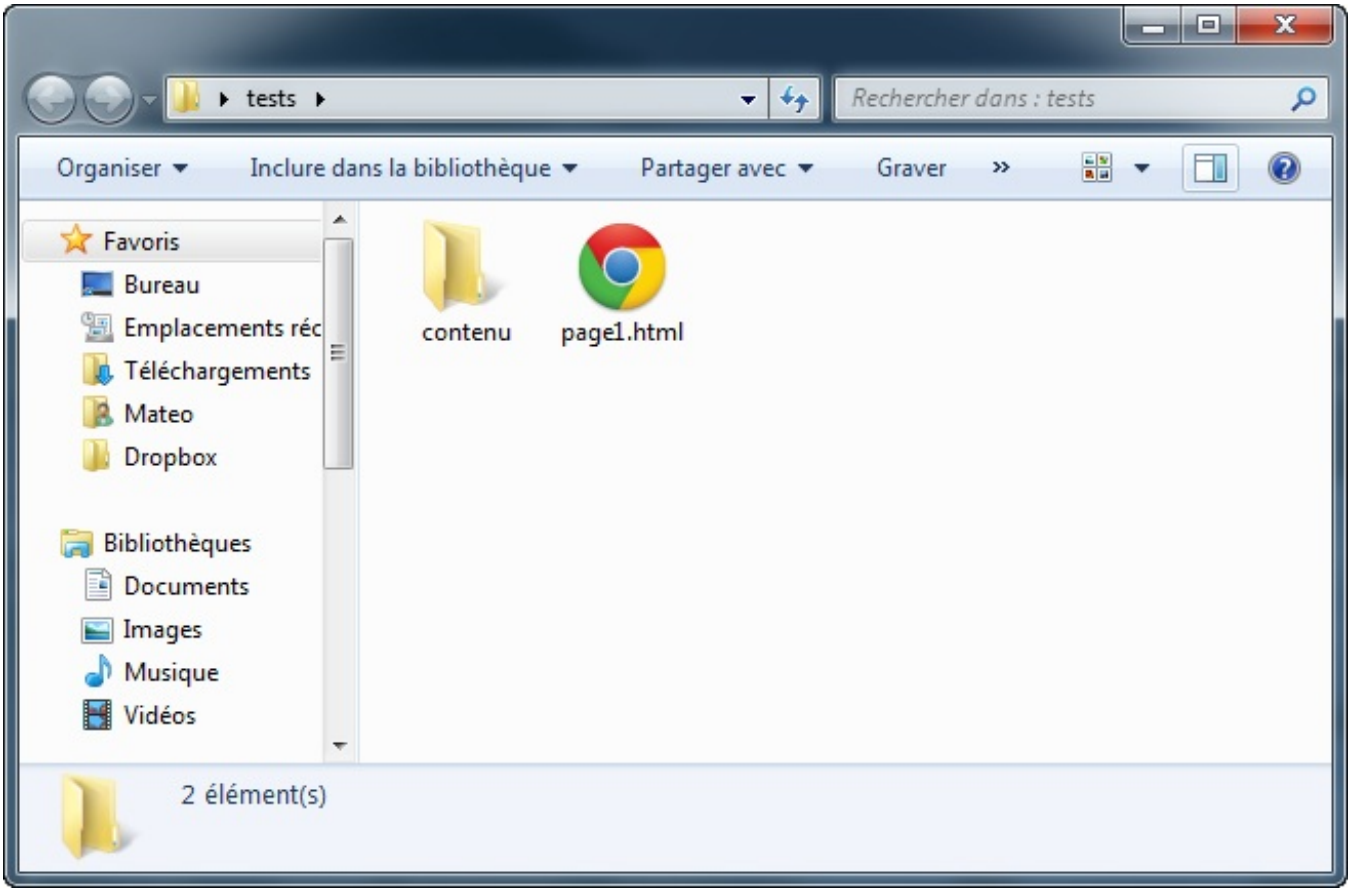
Le lien ci-dessous ouvre `page1.html` :

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 pas 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 ?

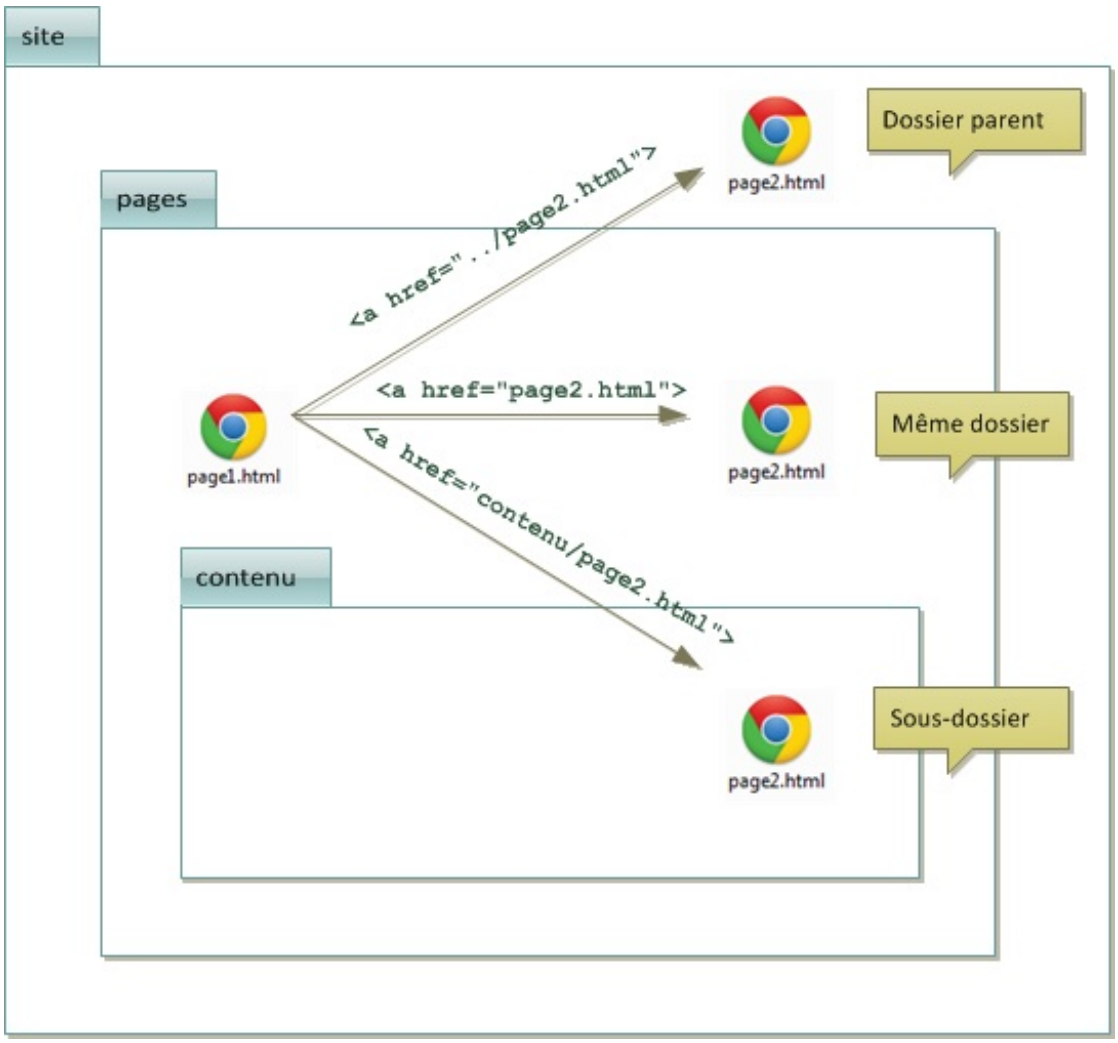
Si votre fichier cible est placé dans un dossier qui se trouve "avant" dans l'arborescence, il faut écrire deux points comme ceci :

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 appelée "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 pour télécharger 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'embarras 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 !).

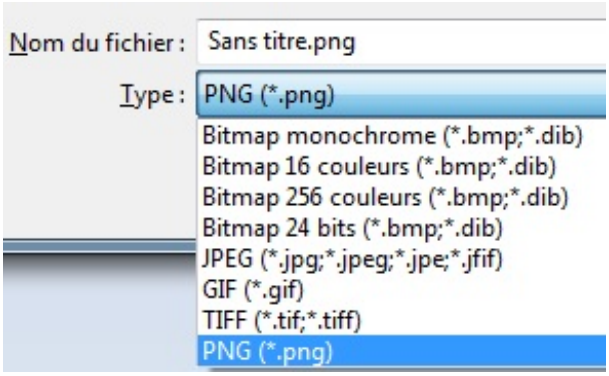
Pour faire en sorte que vos pages restent lisibles et rapides à télécharger, suivez donc activement mes conseils !

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 **
**). 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 (** **). 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 />
     <img src="images/montagne.jpg" alt="Photo de montagne" />
</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 />
```

```
        <img src="images/montagne.jpg" alt="Photo de montagne"
title="C'est beau les Alpes quand même !" />
</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"><img src="img/montagne_mini.jpg"
alt="Photo de montagne" title="Cliquez pour agrandir" /></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.

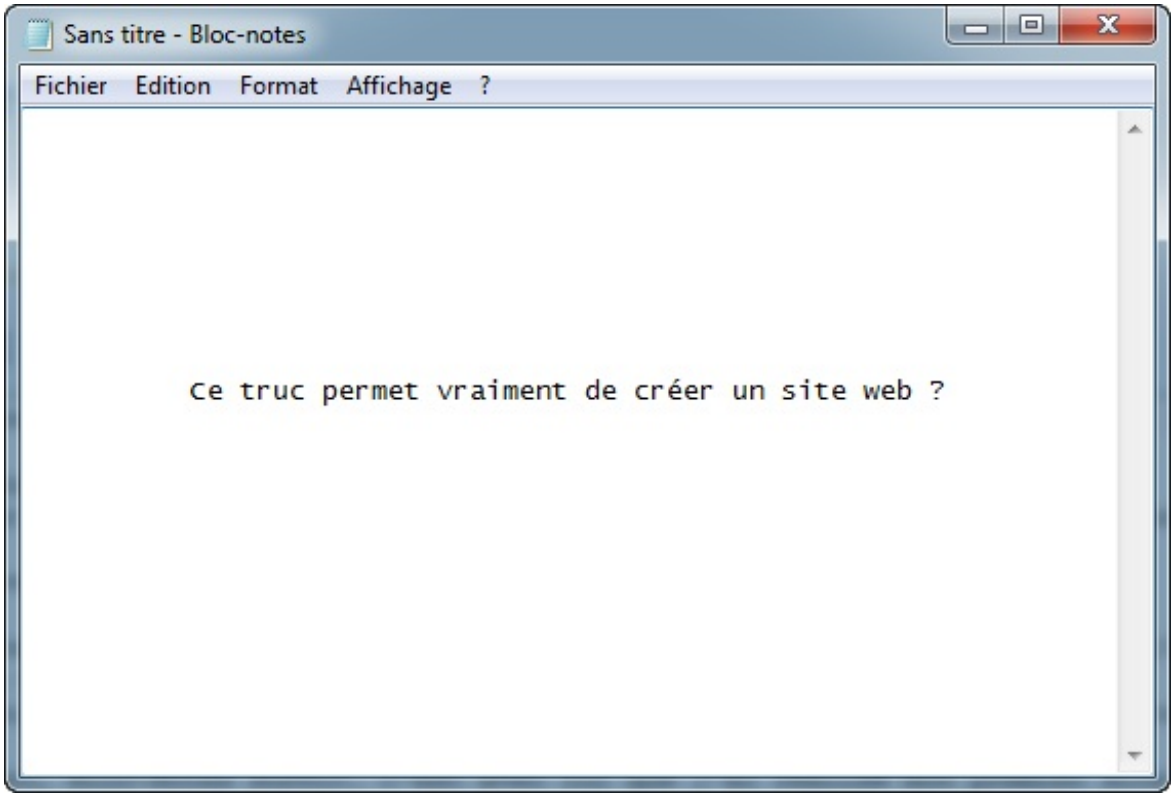
Les figures peuvent être de différents types :

- 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 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>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
</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>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
    <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 ** **). 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>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
    <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>
    <img src="images/internetexplorer.png" alt="Logo Internet
Explorer" />
    <img src="images/firefox.png" alt="Logo Mozilla Firefox" />
    <img src="images/chrome.png" alt="Logo Google Chrome" />
    <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 verrez tout ça dans l'un des chapitres suivants !

Partie 2 : 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 cœur 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 ? À 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 cœur en CSS, le reste pourra être retrouvé dans le mémo en annexe.

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

La petite histoire du CSS

Je vous avais averti dès le début de ce cours : nous allons apprendre deux langages. Nous avons déjà bien entamé notre découverte du HTML, même s'il reste encore de nombreuses choses à apprendre (nous y reviendrons dans quelques chapitres). En revanche, il est temps maintenant de nous intéresser au CSS.

CSS (*Cascading Style Sheets*), c'est cet autre langage qui vient compléter le HTML.
Vous vous souvenez de son rôle ? **Gérer la mise en forme de votre site.**

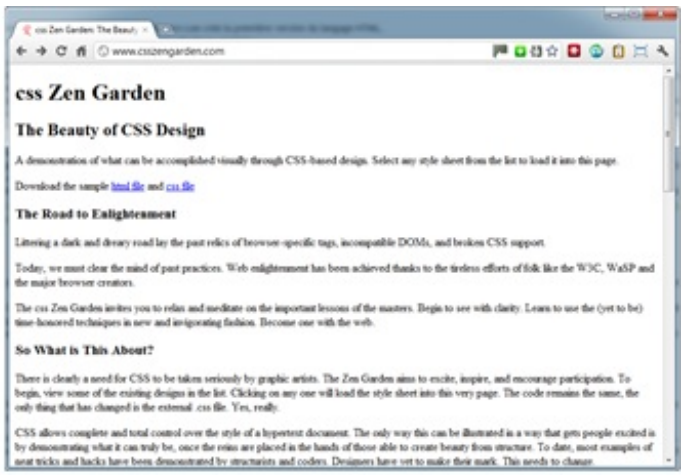
Petit rappel : à quoi sert CSS ?

CSS ? C'est lui qui vous permet de choisir la couleur de votre texte.
Lui qui vous permet de sélectionner la police utilisée sur votre site.
Lui encore qui permet de définir la taille du texte, les bordures, le fond...

Et aussi, c'est lui qui permet de faire la mise en page de votre site. Vous pourrez dire : je veux que mon menu soit à gauche et occupe telle largeur, que l'en-tête de mon site soit calé en haut et qu'il soit toujours visible, etc.

Souvenez-vous de ce petit schéma que nous avions vu dès le premier chapitre :

HTML
(pas de CSS)



HTML + CSS

Grâce au HTML, nous avons pu rédiger le contenu de notre site, mais c'est brut de décoffrage. Le CSS le complète pour mettre tout cela en forme et donner au contenu l'apparence que l'on souhaite.

CSS : des débuts difficiles

Il faut savoir qu'au tout début du Web, CSS n'existait pas. En fait, au départ, il n'y avait que le langage HTML.
Le HTML est né en 1991, et CSS est né en 1996. Alors, vous vous dites sûrement : comment faisait-on la mise en forme de 1991 à 1996 ? Eh bien, uniquement en HTML ! Il y avait en effet des balises HTML dédiée à la mise en forme. `<font color="#aab1c3">` par exemple, permettait de définir la couleur du texte.

Cependant, les pages HTML commençaient à devenir assez complexes. Il y avait de plus en plus de balises et c'était un joyeux mélange entre le fond et la forme, qui rendait la mise à jour des pages web de plus en plus complexe. C'est pour cela que l'on a créé le langage CSS.

Cependant, le CSS n'a pas été adopté par les webmasters immédiatement, loin de là. Il fallait se défaire de certaines mauvaises habitudes, et ça a pris du temps. Encore aujourd'hui on peut trouver des sites web avec des vieilles balises HTML de mise en forme qui n'existent plus, comme `<font>` !

CSS : le support des navigateurs

Tout comme le HTML, le CSS a évolué. Je vous avais indiqué qu'il y avait 4 versions importantes de CSS :

- CSS 1
- CSS 2.0
- CSS 2.1
- CSS 3

En fait, la version CSS 3 n'est pas encore totalement finalisée (ce n'est pas encore une version officielle). Cependant, elle est bien avancée et déjà bien supportée par de nombreux navigateurs aujourd'hui, ce qui fait qu'on peut déjà s'en servir.
Il serait dommage de passer à côté, car CSS 3 ajoute de nombreuses fonctionnalités à CSS (il y a le double de fonctionnalités par rapport à CSS 2.1 !). Nous nous baserons donc dans ce cours sur CSS 3, qui reprend et complète la plupart des fonctionnalités de CSS 2.1.

Ce sont les navigateurs web qui font le travail le plus complexe : ils doivent *lire* le code CSS et *comprendre* comment afficher la page.

Au début des années 2000, Internet Explorer était le navigateur le plus répandu, mais son support du CSS est resté longtemps assez médiocre (pour ne pas dire carrément mauvais). C'était la grande époque de la version 6 (IE6), hélas encore utilisée par une petite partie des internautes aujourd'hui (heureusement, cette proportion tend à diminuer).

Depuis, de nombreux navigateurs sont arrivés et ont chahuté Internet Explorer : Mozilla Firefox bien sûr, mais aussi Google Chrome. Et je ne vous parle pas du succès des Mac et iPhone avec leur navigateur Safari. Cela a incité Microsoft a réagir, qui a publié (après une longue période d'inactivité) IE 7, puis IE 8 et IE 9. On parle déjà de IE 10.

Bon ton cours d'histoire c'est bien joli, mais en quoi ça me concerne aujourd'hui ?

Que faut-il retenir de tout ça ? Que les navigateurs ne connaissent pas toutes les propriétés CSS qui existent. Plus le navigateur est vieux, moins il connaît de fonctionnalités CSS.

Je vais vous présenter dans ce cours un certain nombre de fonctionnalités de CSS qui ne fonctionnent pas forcément sur les navigateurs les plus vieux. Je ne peux pas l'éviter, c'est comme ça : *aucun navigateur ne connaît parfaitement toutes les fonctionnalités CSS* de toute façon ! Au pire, si le navigateur ne connaît pas une fonctionnalité de CSS, il l'ignore et ne met pas en forme, mais ça ne fait pas planter votre page, ce qui fait que celle-ci sera toujours lisible.

Je vous recommande fortement de mettre dans vos favoris les sites web www.caniuse.com et normansblog.de qui affichent des tables de compatibilité des fonctionnalités de HTML et CSS en fonction des différents navigateurs (et de leurs différentes versions). Regardez en particulier les tables de compatibilité pour CSS de caniuse.com.

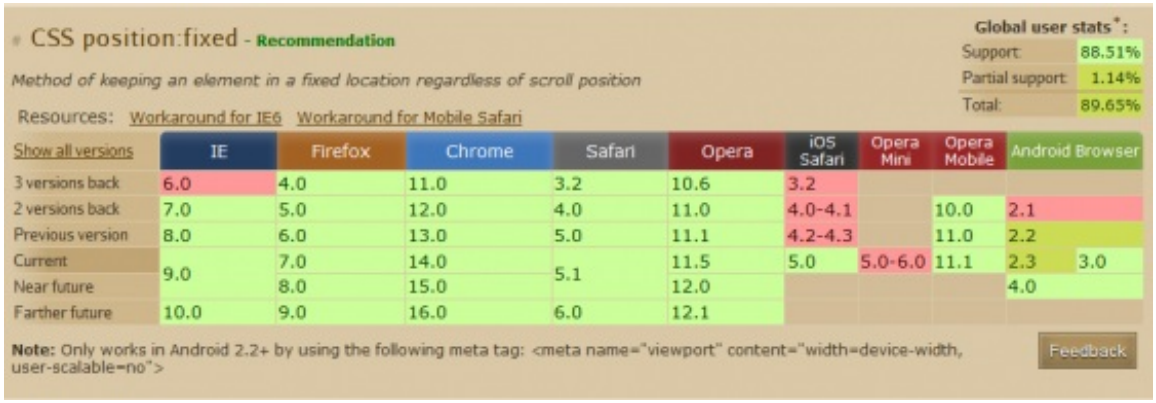


Table de compatibilité CSS de caniuse.com

Où écrit-on le CSS ?

Vous avez le choix, car on peut écrire du code en langage CSS à 3 endroits différents :

- Dans un fichier `.css` (**méthode la plus recommandée**) ;
- Dans l'en-tête `<head>` du fichier HTML ;
- Directement dans les balises du fichier HTML via un attribut `style` (**méthode la moins recommandée**) ;

Je vais vous présenter ces 3 méthodes, mais sachez d'ores et déjà que la première... est la meilleure.

Dans un fichier `.css` (recommandé)

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 HTML qui ont l'extension `.html`). C'est la méthode la plus pratique et la plus flexible. Ca nous évite de tout mélanger dans un même fichier. J'utiliserai cette technique tout au long de la suite de ce cours.

Commençons à pratiquer dès maintenant ! On va partir du fichier HTML suivant :

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>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>
```

Vous noterez la ligne `<link rel="stylesheet" href="style.css" />` : c'est elle qui indique que ce fichier HTML va être associé à un fichier appelé `style.css` qui va gérer sa mise en forme.

Enregistrez ce fichier avec le nom que vous voulez (par exemple `page.html`). Pour le moment, rien d'extraordinaire à part la nouvelle balise que nous avons ajoutée. 

Maintenant, créez un *nouveau* fichier vide dans votre éditeur de texte (par exemple Notepad++) et copiez-y ce bout de code CSS (rassurez-vous je vous expliquerai ce qu'il veut dire tout à l'heure) :

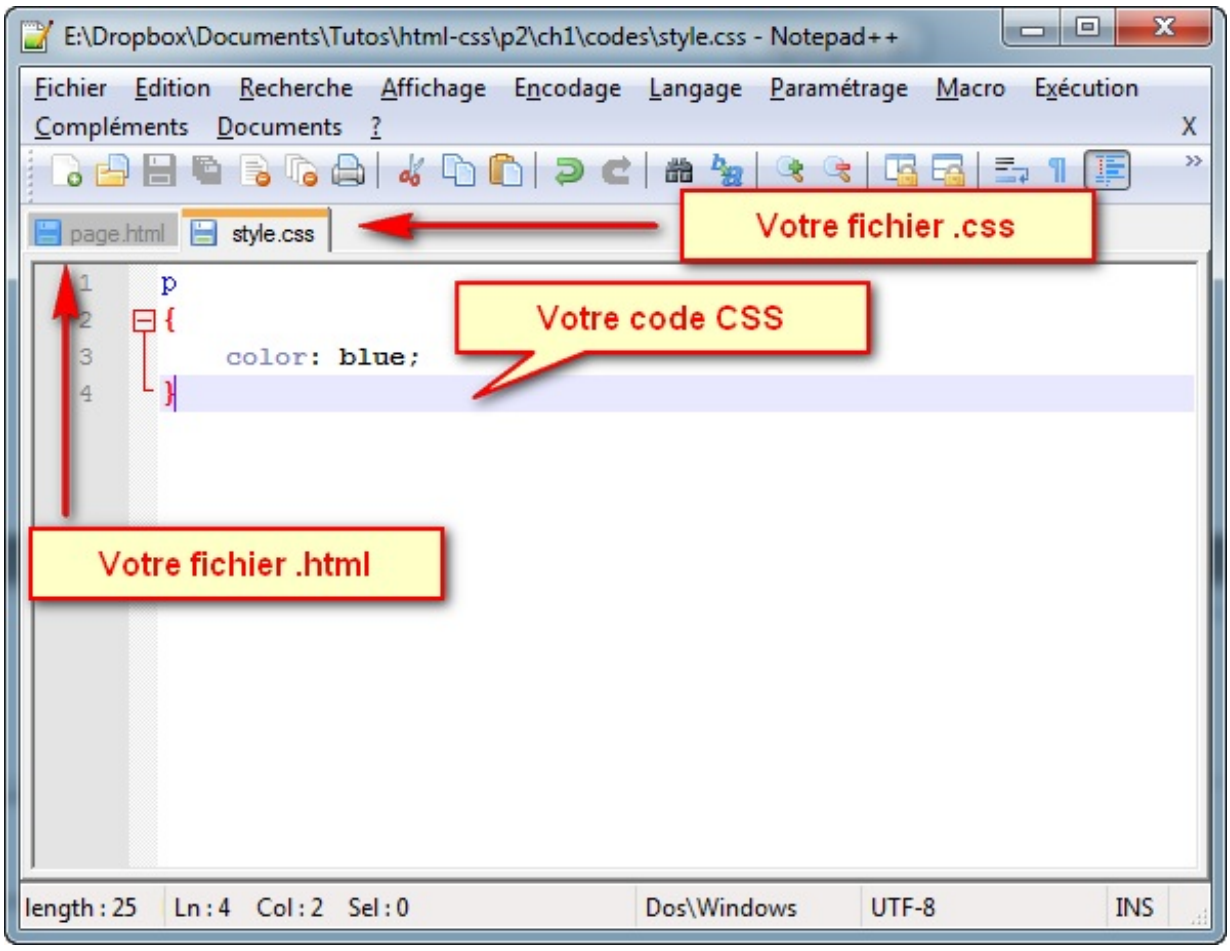
Code : CSS

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

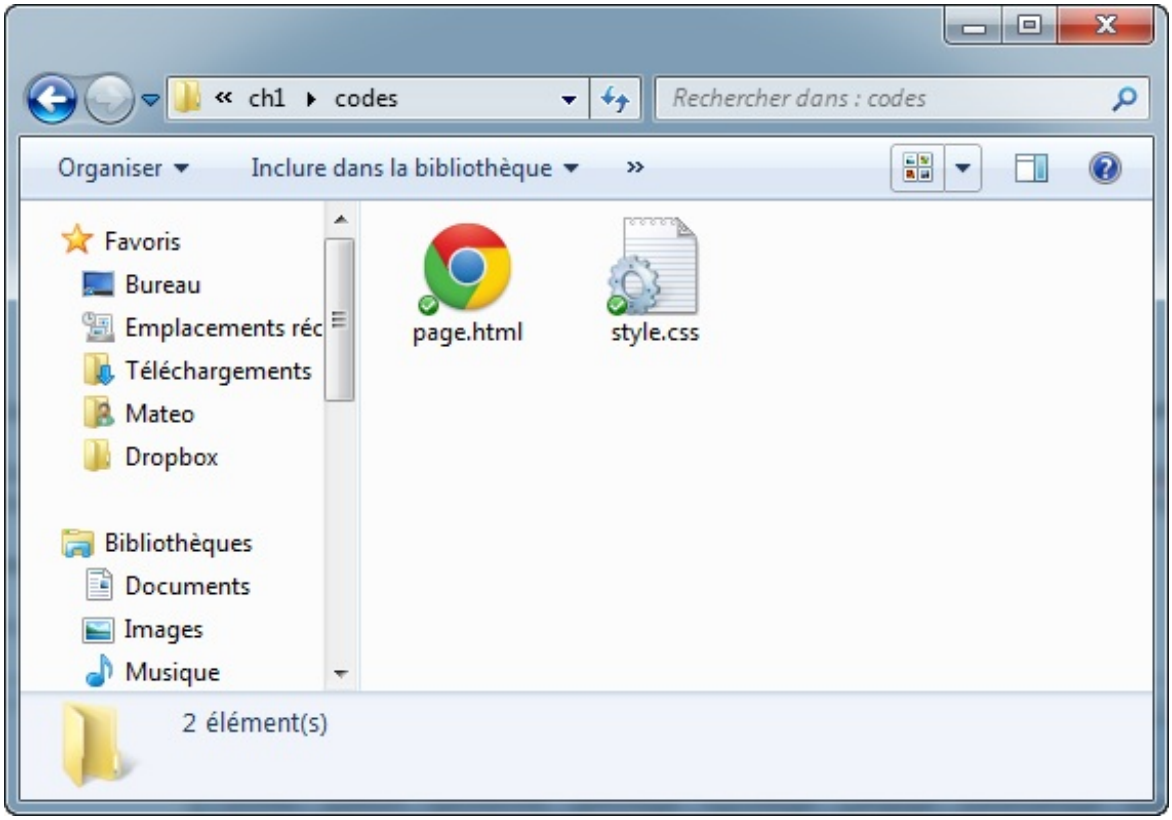
Pour activer la coloration du code dans Notepad++, allez dans le menu `Langage > C > CSS`.

Enregistrez le fichier en lui donnant un nom qui se termine par `.css`, comme `style.css`. Placez ce fichier `.css` dans le même dossier que votre fichier `.html`.

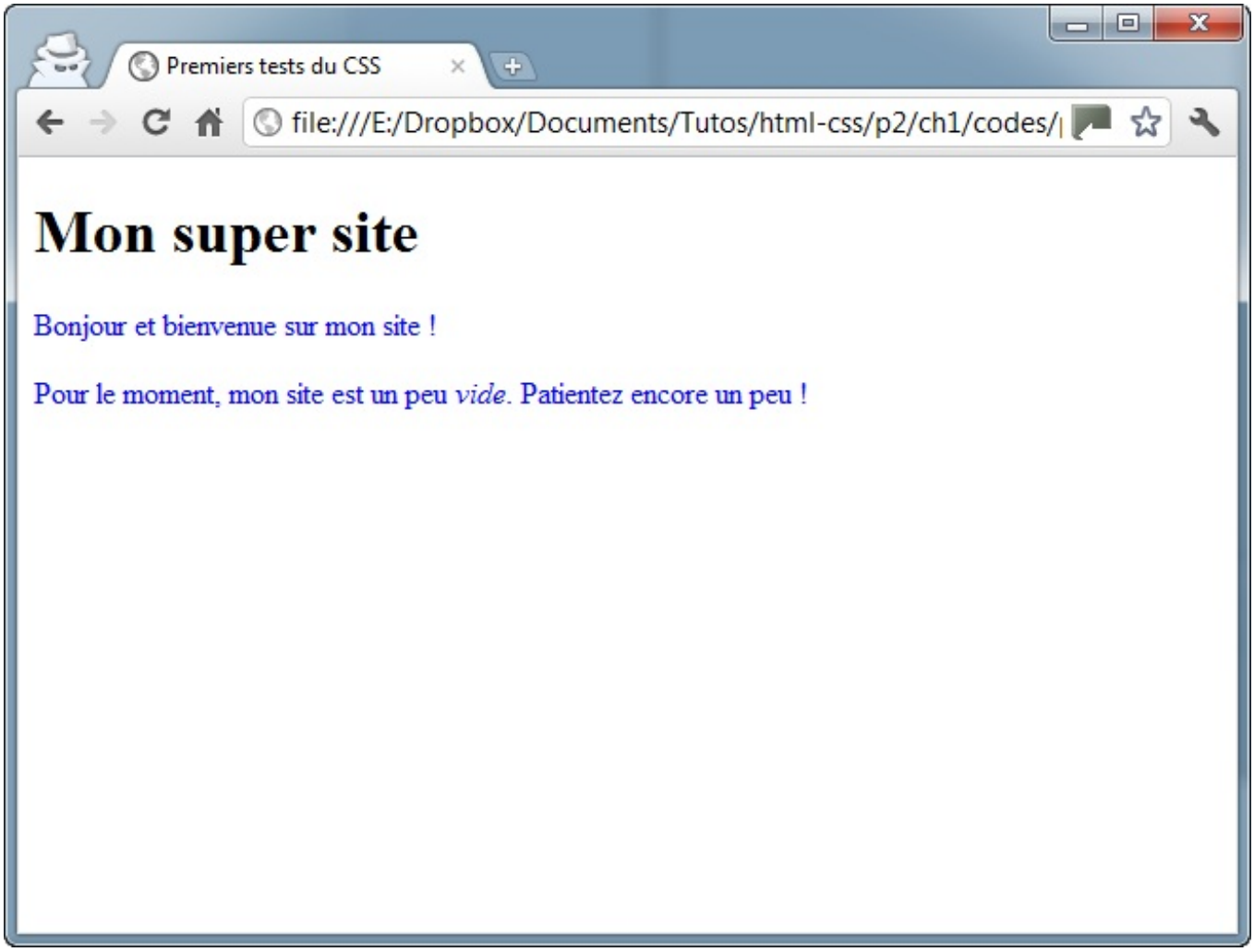
Dans Notepad++, vous devriez voir quelque chose comme ça :



Dans votre explorateur de fichiers, vous devriez les voir placés côte à côte. D'un côté le `.html`, de l'autre le `.css` :



Ouvrez maintenant votre fichier `page.html` dans votre navigateur pour l'essayer, comme vous le faites d'habitude. Regardez, c'est magique : vos paragraphes sont écrits en bleu !



Essayer !

Il est inutile d'ouvrir directement le fichier `style.css` dans le navigateur. Il faut ouvrir le fichier `page.html` (il fera automatiquement appel au fichier `style.css`).

Dans l'en-tête `<head>` du fichier HTML

Il existe une autre méthode pour utiliser du CSS dans ses fichiers HTML : cela consiste à insérer le code CSS directement dans une balise `<style>` à l'intérieur de l'en-tête `<head>`.

Voici comment on peut obtenir exactement le même résultat avec un seul fichier `.html` qui contient le code CSS :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
<style>
p
{
color: blue;
}
</style>
        <title>Premiers tests du CSS</title>
    </head>
```

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

        <p>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>
```

Testez, vous verrez que le résultat est le même :

Essayer !

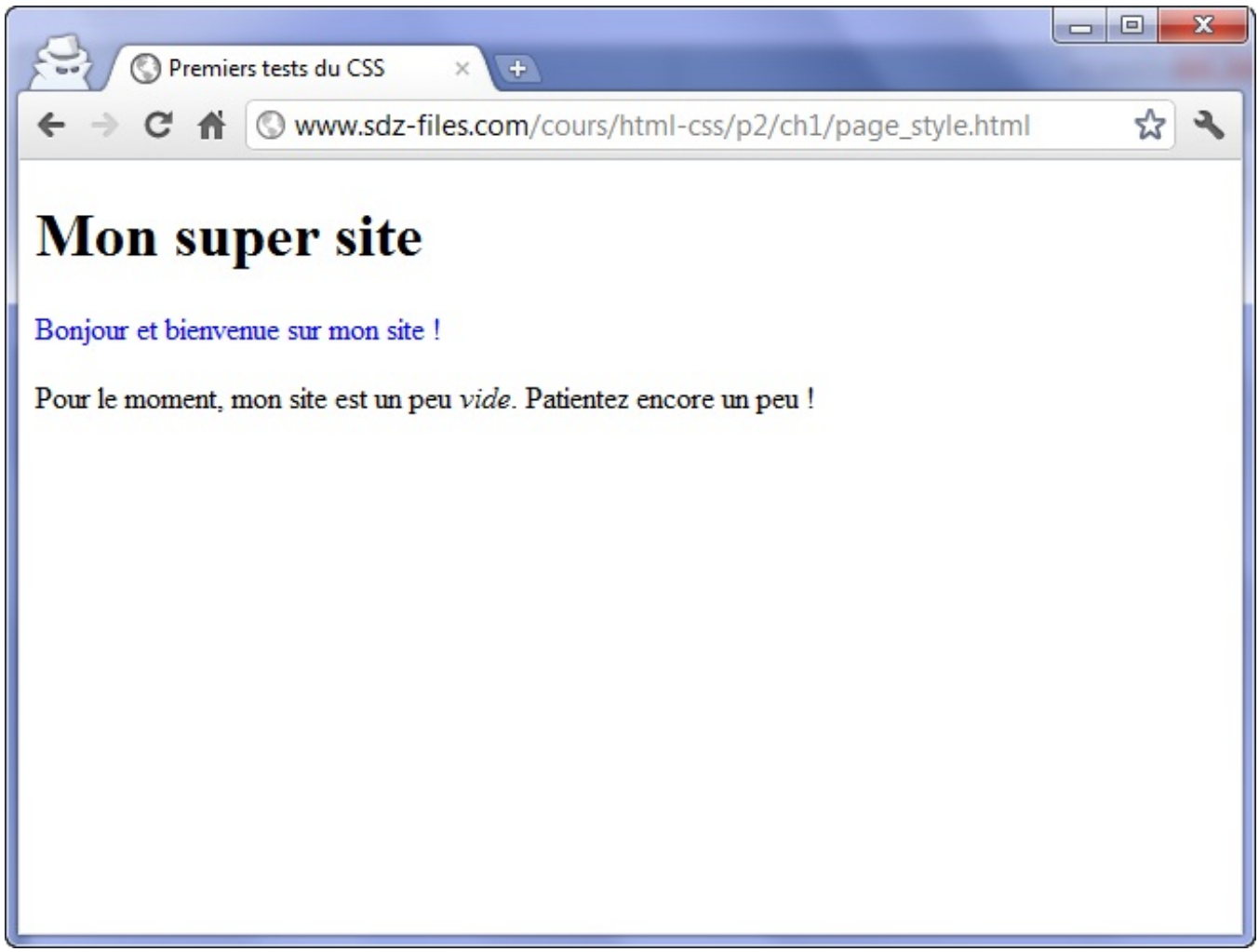
Directement dans les balises (non recommandé)

Dernière méthode, à manipuler avec précaution : vous pouvez ajouter un attribut `style` à n'importe quelle balise. Vous insèrerez votre code CSS directement dans cet attribut :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Premiers tests du CSS</title>
    </head>

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

<p style="color: blue;">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>
```

Cette fois, seule la balise du premier paragraphe qui contient le code CSS sera colorée en bleu :



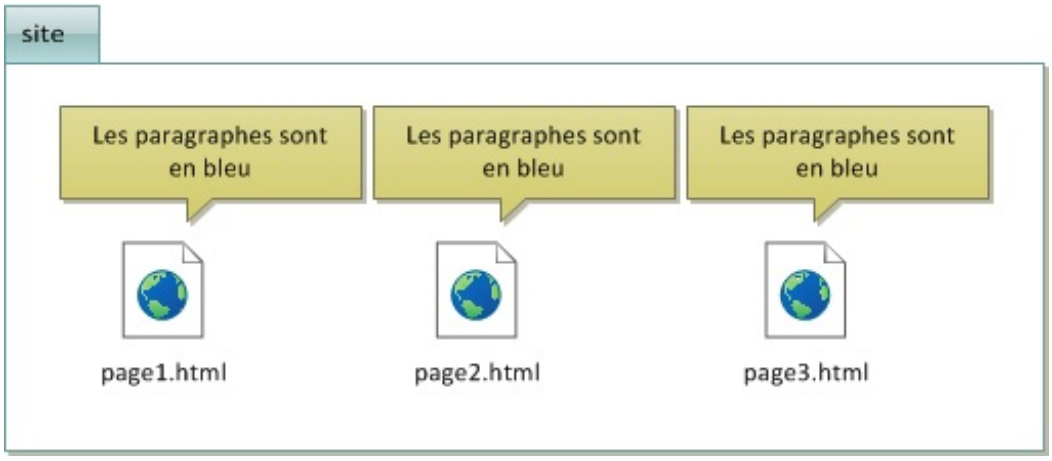
Essayer !

Quelle méthode choisir ?

Je trouve que la première méthode que tu recommandes est plus compliquée que les deux autres ! Pourquoi nous conseilles-tu de créer 2 fichiers, j'étais bien moi avec juste un fichier `.html` !

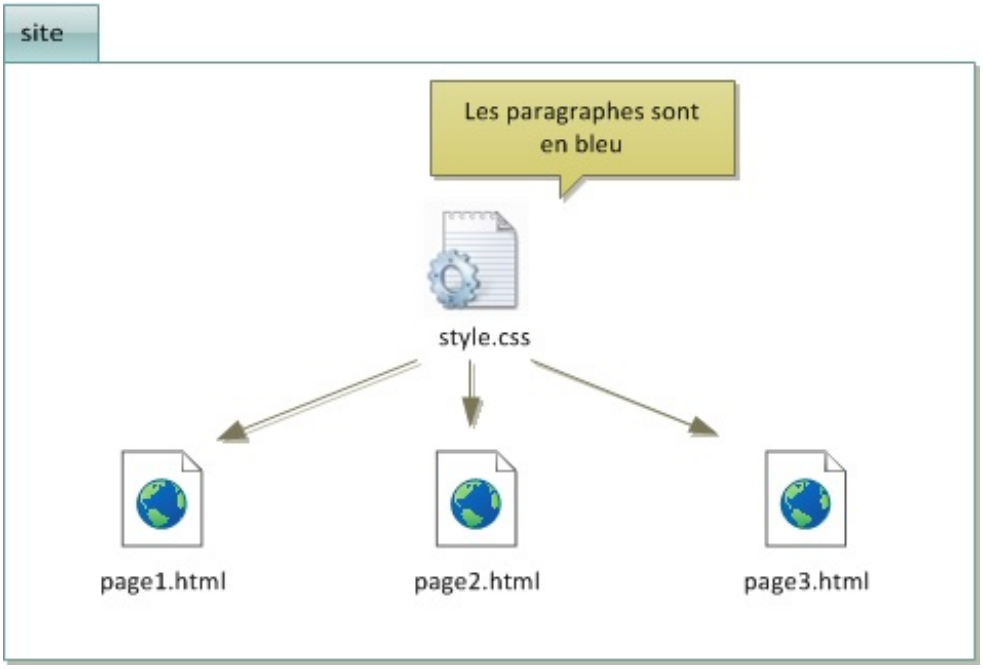
Je vous recommande fortement de prendre l'habitude de travailler avec la première méthode parce que c'est celle utilisée par la majorité des webmasters... Pourquoi ?

Pour le moment, vous faites vos tests dans un seul fichier HTML. Cependant, votre site sera plus tard constitué de plusieurs pages HTML, on est d'accord ?
Imaginez : si vous placez le code CSS directement dans le fichier HTML, il faudra copier ce code dans tous les fichiers HTML de votre site ! Et si demain vous changez d'avis, par exemple si vous voulez que vos paragraphes soient écrits en rouge et non en bleu, il faudra modifier chaque fichier HTML un à un !



Le code CSS est répété dans chaque fichier HTML : pas pratique !

Si vous travaillez avec un fichier CSS externe, vous n'aurez besoin d'écrire cette instruction qu'une seule fois pour tout votre site !



Le code CSS est indiqué une fois pour toutes dans un fichier CSS : c'est plus simple !

Appliquer un style : sélectionner une balise

Maintenant que nous savons où placer le code CSS, intéressons-nous à ce 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 paragraphes `<p>`, je dois écrire `p`.
- **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 cœur (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 donc 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 (;).

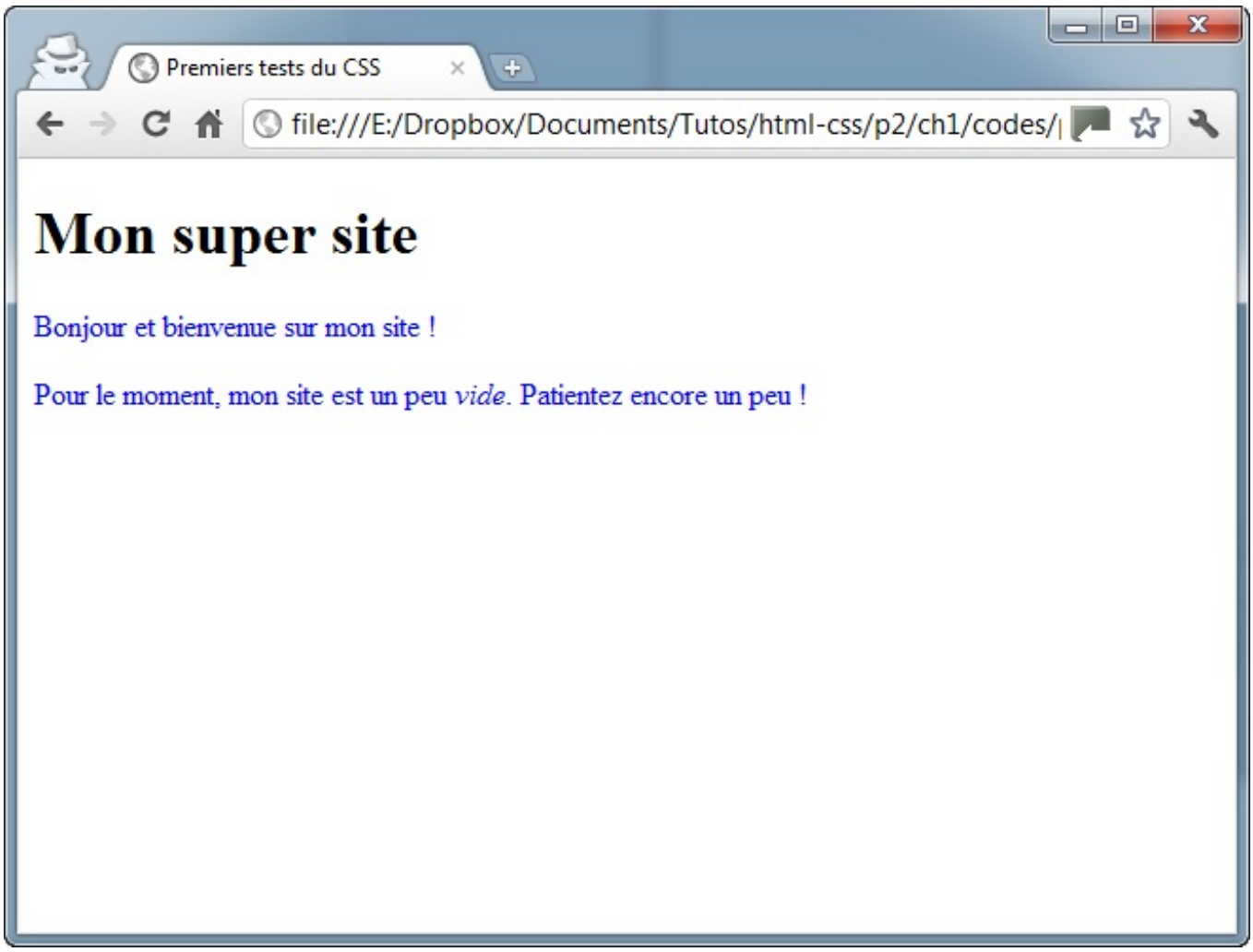
Je vous apprendrai de nombreuses propriétés dans les chapitres suivants. Pour le moment, on va juste changer la couleur pour s'entraîner dans nos exemples.

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 :

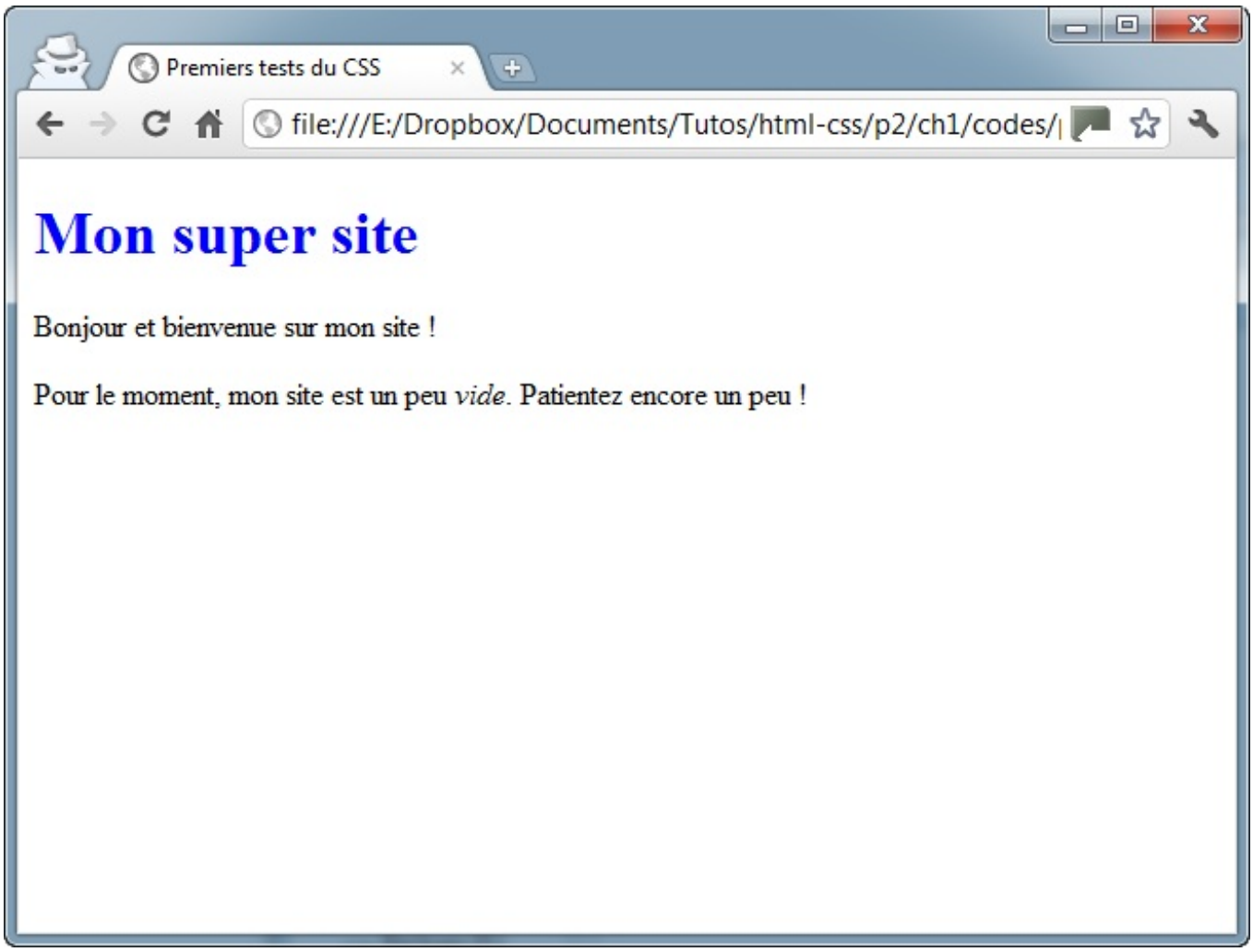


Essayez de changer le nom de la balise affectée par le code CSS. Par exemple, si j'écris `h1`, c'est le titre qui sera écrit en bleu. Modifiez votre fichier `style.css` comme ceci :

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;
}
```

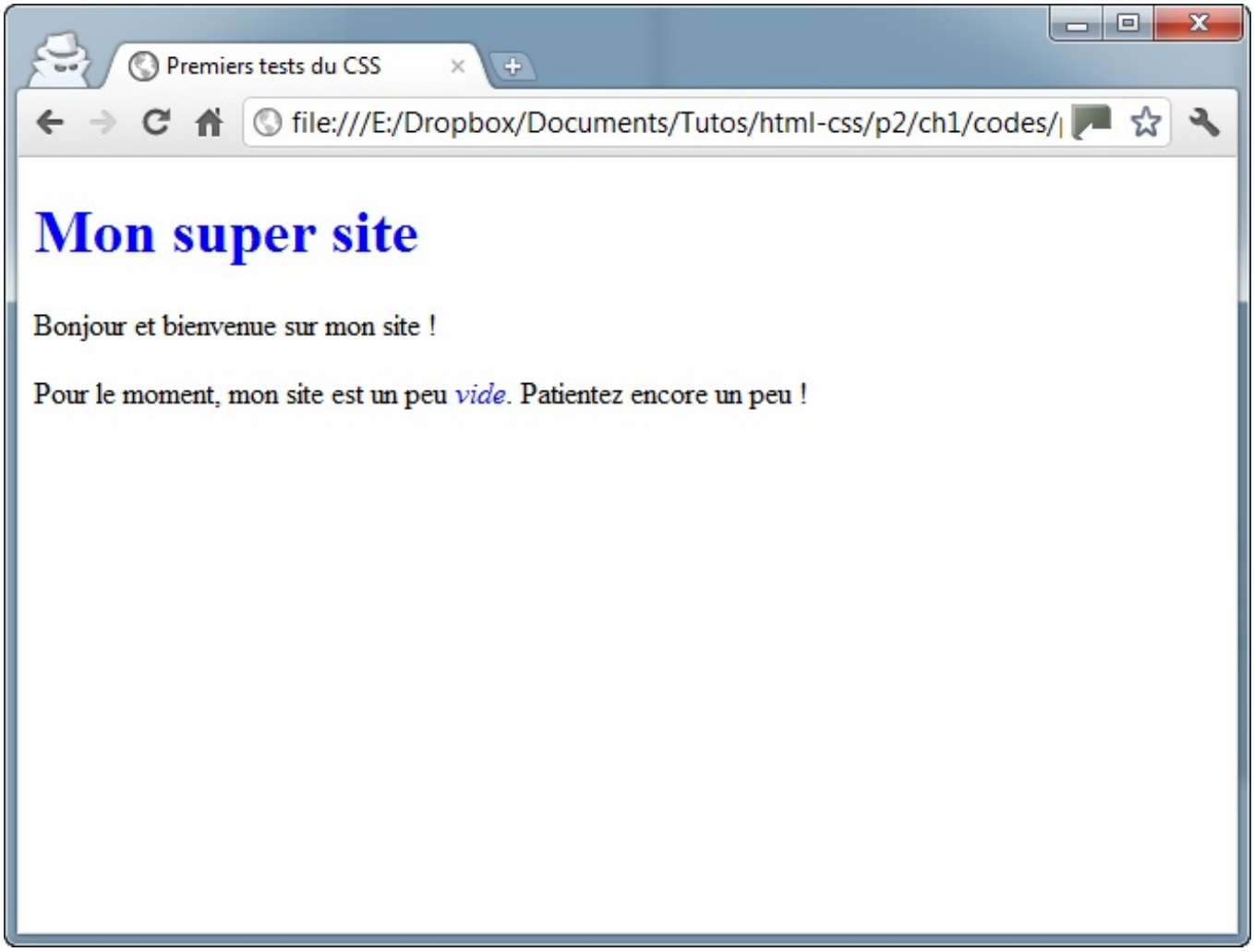
Il signifie que nos titres **<h1>** et nos textes importants **** doivent s'afficher en bleu. Par contre, c'est un peu répétitif, vous ne trouvez pas ?

Heureusement, il existe un moyen en CSS d'aller plus vite si les 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* **** *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 looong 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 écrits 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 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.

Code : HTML

```
<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, tant que le nom commence par une lettre.

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>
```

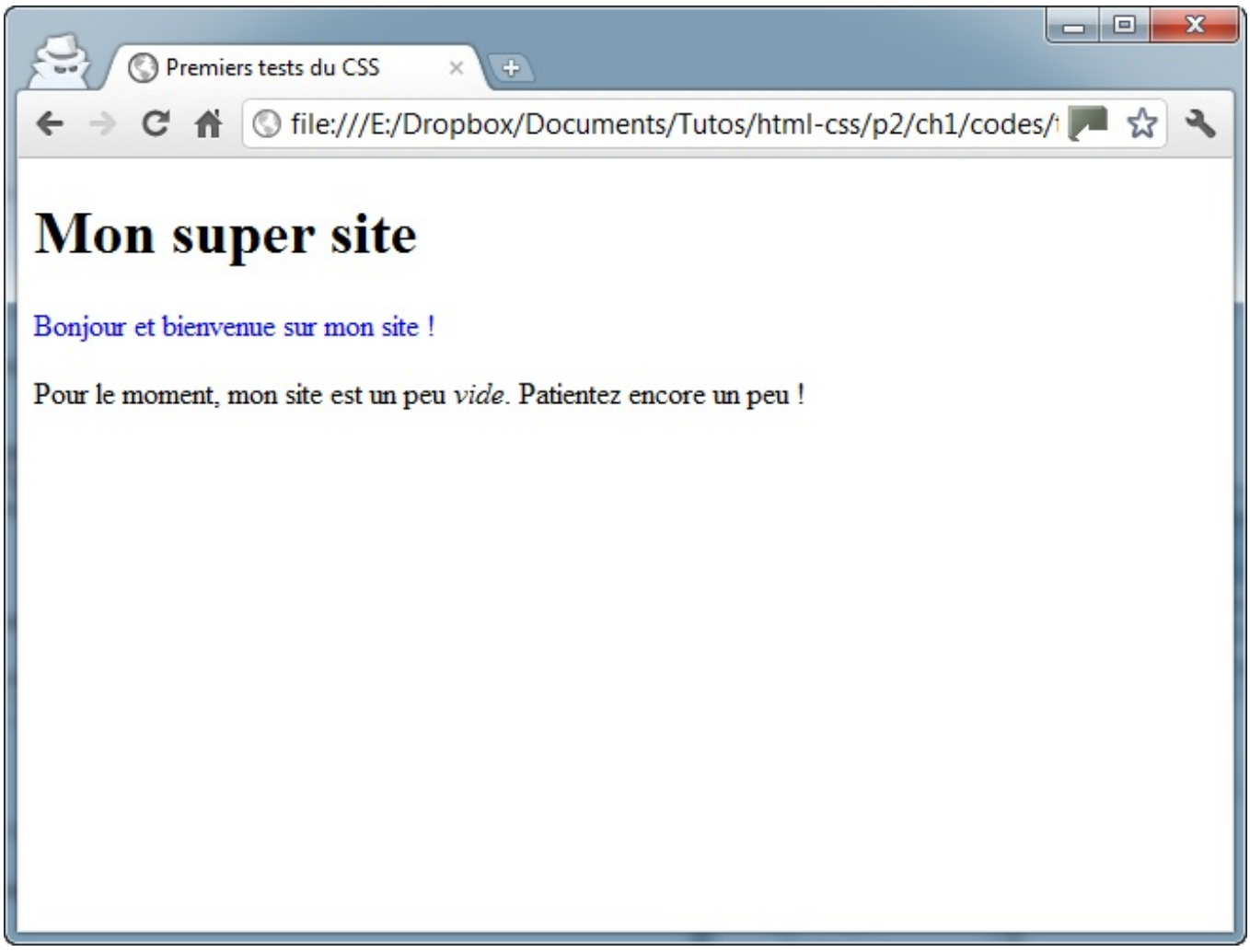
Maintenant que c'est fait, votre paragraphe est identifié. Il a un nom : *introduction*. Vous allez pouvoir réutiliser ce nom dans le fichier CSS pour dire : "*Je veux que seules les balises qui ont comme nom introduction soient affichées en bleu*".

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* dans le code.

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 ancres).

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

Code : HTML

```
<img src="images/logo.png" alt="Logo du site" id="logo" />
```

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

Code : CSS

```
#logo
{
    /* Indiquez 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` retenez que 2 balises peuvent avoir le même nom avec l'attribut class. Un nom d'id doit en revanche être unique dans la page HTML.

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>
```

Ça 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 dites *universelles* qui n'ont aucune signification particulière (elles n'indiquent pas que le mot est important par exemple). Il y a une petite (mais importante !) différence entre ces deux balises :

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

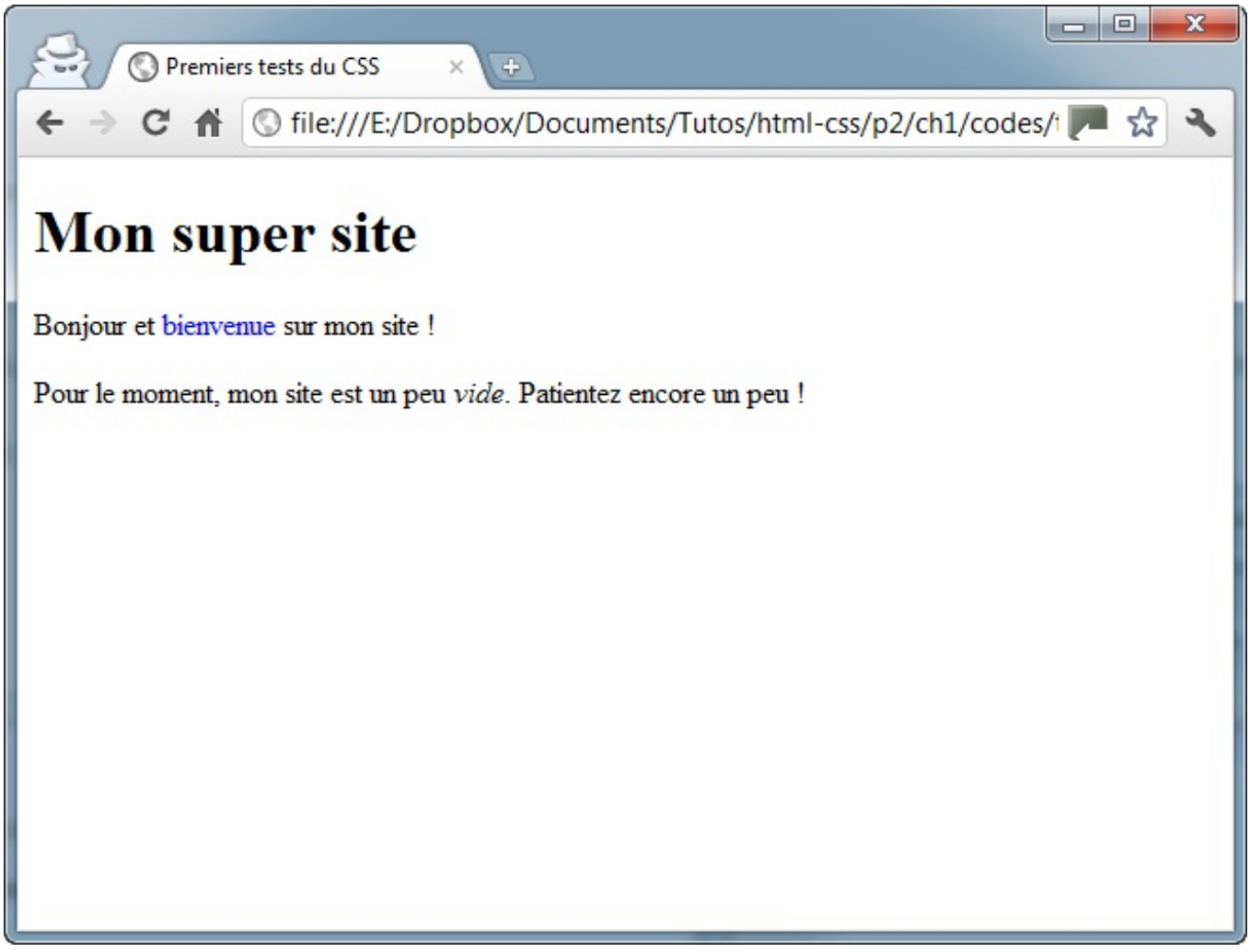
Pour le moment donc, nous allons utiliser plutôt la balise ****. On la met autour de "bienvenue", on lui rajoute une classe (du nom qu'on veut), on crée le CSS et c'est gagné !

Code : HTML

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

Code : CSS

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



Appliquer un style : les sélecteurs avancés

En CSS, le plus difficile est de savoir cibler le texte dont on veut changer la forme. Pour cibler (on dit "sélectionner") les éléments de la page à modifier, on utilise ce qu'on appelle des sélecteurs. Vous en avez déjà utilisé un peu plus tôt dans ce chapitre, résumons-les pour commencer.

Les sélecteurs que vous connaissez déjà

Ces sélecteurs, que nous avons vus précédemment, sont de loin les plus couramment utilisés. Il faut les connaître par coeur.

Commençons par la base de la base :

Code : CSS

```
p
{

}
```

... signifie "Je veux affecter tous les paragraphes". Après, c'est à vous de dire ce que vous faites à ces paragraphes (vous les écrivez en bleu par exemple).

Nous avons aussi vu :

Code : CSS

```
h1, em
{

}
```

... qui signifie "Tous les titres et textes importants". Nous avons sélectionné deux balises d'un coup.

Et enfin, nous avons vu comment sélectionner des balises précises à qui nous avons donné un nom grâce aux attributs `class` et `id` :

Code : CSS

```
.class
{

}

#id
{

}
```

Vous savez quoi ? Il existe des dizaines d'autres façons de cibler des balises en CSS ! Nous n'allons pas toutes les voir, car il y en a beaucoup et certaines sont complexes, mais voici déjà de quoi vous permettre d'être plus efficaces en CSS !

Les sélecteurs avancés

** : sélecteur universel*

Code : CSS

```
*
{

}
```

Sélectionne toutes les balises sans exception. On l'appelle le sélecteur universel.

A B : une balise contenue dans une autre

Code : CSS

```
h3 em
{

}
```

Sélectionne toutes les balises **** situées à l'intérieur d'une balise **<h3>**. Notez qu'il n'y a pas de virgule entre les deux noms de balise.

Exemple de code HTML correspondant :

Code : HTML

```
<h3>Titre avec <em>texte important</em></h3>
```

A + B : une balise qui en suit une autre

Code : CSS

```
h3 + p
{

}
```

Sélectionne la première balise **<p>** située après un titre **<h3>**.

Exemple :

Code : HTML

```
<h3>Titre</h3>

<p>Paragraphe</p>
```

A[attribut] : une balise qui possède un attribut

Code : CSS

```
a[title]
{

}
```

Sélectionne tous les liens **<a>** qui possèdent un attribut `title`.

Exemple :

Code : HTML

```
<a href="http://site.com" title="Infobulle">
```

A[attribut="Valeur"] : une balise, un attribut et une valeur exacte

Code : CSS

```
a[title="Cliquez ici"]
{

}
```

Idem, mais l'attribut doit en plus avoir exactement pour valeur "Cliquez ici".

Exemple :

Code : HTML

```
<a href="http://site.com" title="Cliquez ici">
```

A[attribut="Valeur"] : une balise, un attribut et une valeur*

Code : CSS

```
a[title*="ici"]
{

}
```

Idem, l'attribut doit cette fois contenir dans sa valeur le mot "ici" (peu importe sa position).

Exemple :

Code : HTML

```
<a href="http://site.com" title="Quelque part par ici">
```

D'autres sélecteurs existent !

Je ne vous ai présenté qu'une partie des sélecteurs CSS ici, mais sachez qu'il en existe beaucoup d'autres. Si vous voulez une

liste complète, vous pouvez vous renseigner directement à la source : sur le site du W3C ! C'est très complet.

Sachez que nous découvrirons certains de ces autres sélecteurs dans la suite de ce cours !
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, HTML 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").

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, on ne sait rien faire de bien palpitant.

Formatage du texte

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").

Pas de surprise particulière : nous sommes toujours dans le CSS, et nous allons réutiliser ce que nous venons d'apprendre dans le chapitre précédent. Nous allons donc travailler directement au sein du fichier `.css` que nous avons créé.

Ce chapitre va être l'occasion de découvrir de nombreuses propriétés CSS : nous allons voir comment modifier la taille du texte, changer la police, aligner le texte...

La taille

Pour modifier la taille du texte, on utilise la propriété CSS **`font-size`**. Mais comment indiquer la taille du texte ? C'est là que ça se corse car plusieurs techniques vous sont proposées :

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

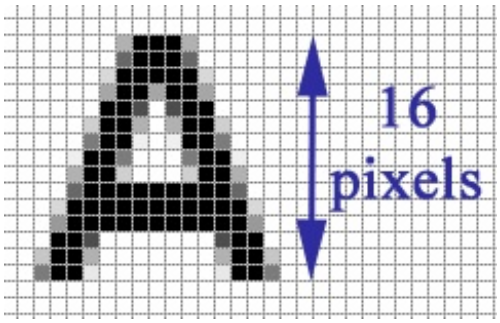
Une taille absolue

Pour indiquer une taille absolue, on utilise généralement les pixels. Pour avoir un texte de 16 pixels de hauteur, vous devez écrire :

Code : CSS

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

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



Voici un exemple d'utilisation (placez ce code dans votre fichier .css) :

Code : CSS

```
p
{
    font-size: 14px; /* Paragraphes de 14 pixels */
}
h1
{
    font-size: 40px; /* Titres de 40 pixels */
}
```

Essayer !

Si vous le souhaitez, des tailles en centimètres ou millimètres sont aussi disponibles. Remplacez "px" par "cm" ou "mm". Ces unités sont cependant moins bien adaptées aux écrans.

Une valeur relative

C'est la méthode recommandée car le texte s'adapte alors plus facilement aux préférences de tous les visiteurs.

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

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

Vous pouvez tester l'utilisation de ces valeurs dans votre code CSS :

Code : CSS

```
p
{
    font-size: small;
}
h1
{
    font-size: large;
}
```

Bon, cette technique a un défaut : il n'y a que 7 tailles disponibles (car il n'y a que 7 noms). Heureusement il existe d'autres moyens. Ma technique préférée consiste à indiquer la taille en "em". C'est une unité spécifique au CSS.

- Si vous écrivez 1em, le texte a une taille normale.
- Si vous voulez grossir le texte, vous pouvez inscrire une valeur supérieure à 1, comme 1.3em.
- Si vous voulez réduire le texte, inscrivez une valeur inférieure à 1, comme 0.8em.

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" !

Exemple :

Code : CSS

```
p
{
    font-size: 0.8em;
}
h1
{
    font-size: 1.3em;
}
```

D'autres unités sont disponibles. Vous pouvez essayer le "ex" (qui fonctionne sur le même principe que le em mais qui est plus petit de base) et le pourcentage (80%, 130%...).

La police

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.

La bonne nouvelle, c'est que depuis CSS 3, il est possible de faire télécharger automatiquement une police au navigateur. Je vous expliquerai dans un second temps comment faire cela. 

Modifier la police utilisée

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

Code : CSS

```
balise
{
    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 précise en général *plusieurs* noms de police, séparés par des virgules :

Code : CSS

```
balise
{
    font-family: police1, police2, police3, police4;
}
```

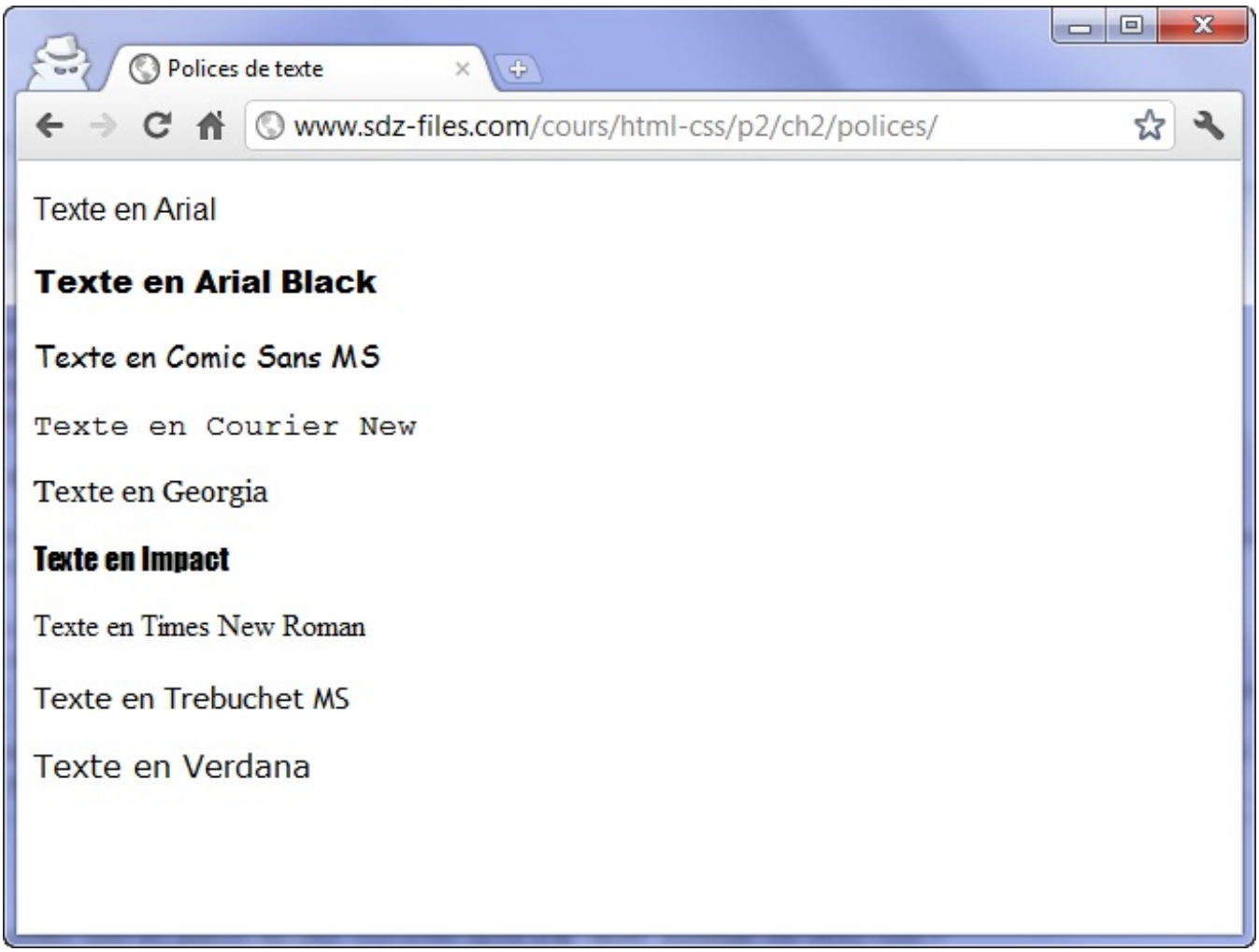
Le navigateur essaiera d'abord de mettre la `police1`. S'il ne l'a pas, il essaiera de mettre la `police2`. S'il ne l'a pas, il essaiera la `police3` et ainsi de suite.
En général, on indique en tout dernier `serif`, ce qui correspond à une police standard (qui ne se met que si aucune autre police n'a été trouvée).

Il existe aussi une autre police par défaut appelée `sans-serif`. Il y a une petite différence entre ces deux polices par défaut : `serif` possède des pattes de liaison en bas des lettres, tandis que `sans-serif` n'en possède pas. Oui c'est subtil.

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 :

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

Voici à quoi ressemblent ces polices :



Essayer !

Ainsi, si j'écris :

Code : CSS

```
p
{
    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, je conseille de l'entourer de guillemets, comme je l'ai fait pour "Arial Black".

Utiliser une police personnalisée avec @font-face

Je trouve le choix des polices trop limité.
Comment puis-je utiliser ma police préférée sur mon site web ?

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

Mais attention, il y a des défauts (ça serait trop beau sinon) :

- Il faudra que le navigateur de vos visiteurs **télécharge** automatiquement le fichier de la police, qui peut peser parfois 1 Mo voire plus...
- La plupart des polices sont soumises au droit d'auteur, il n'est donc **pas légal** de les utiliser sur son site. Heureusement, il existe des sites comme fontsquirrel.com et dafont.com qui en proposent un certain nombre libres de droit à télécharger. Je recommande en particulier fontsquirrel.com car il permet de télécharger des packs prêts à l'emploi pour CSS 3. A noter aussi le service Google Web Fonts qui est très bien fait.
- Il existe **plusieurs formats** de fichier de police, et ils ne fonctionnent pas sur tous les navigateurs.

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

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

En CSS, pour définir une nouvelle police, vous devez la déclarer comme ceci :

Code : CSS

```
@font-face {
    font-family: 'MaSuperPolice';
    src: url('MaSuperPolice.eot');
}
```

Le fichier de police (ici *MaSuperPolice.eot*) doit ici être situé dans le même dossier que le fichier CSS (ou dans un sous-dossier si vous utilisez un chemin relatif).

Je croyais qu'il y avait plusieurs formats de police ?

Oui, d'ailleurs les .eot ne marchent que sur Internet Explorer. L'idéal est de proposer plusieurs formats de police : le navigateur téléchargera celui qu'il sait lire. Voici comment indiquer plusieurs formats :

Code : CSS

```
@font-face {
    font-family: 'MaSuperPolice';
    src: url('MaSuperPolice.eot') format('eot'),
         url('MaSuperPolice.woff') format('woff'),
         url('MaSuperPolice.ttf') format('truetype'),
         url('MaSuperPolice.svg') format('svg');
}
```

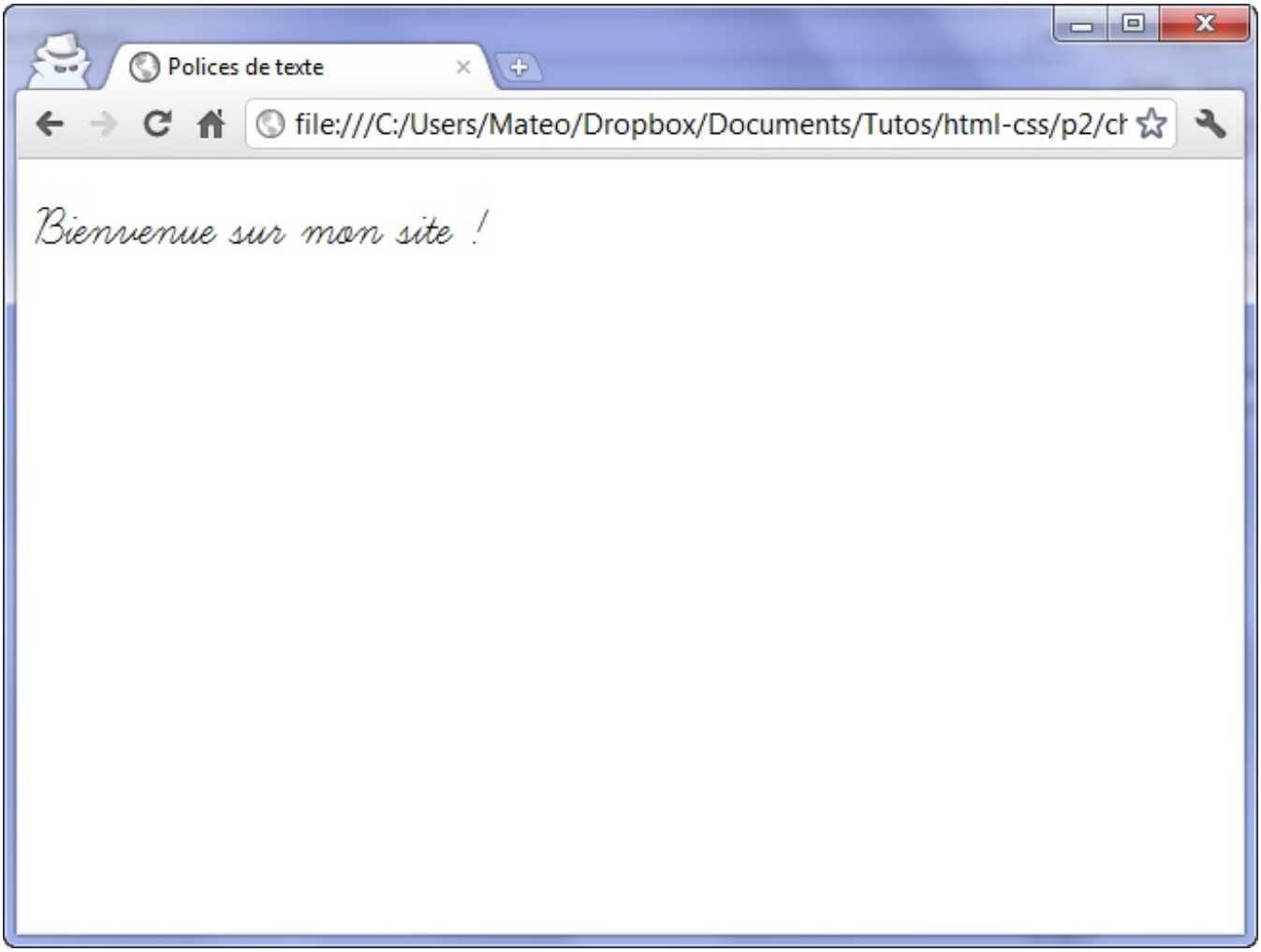
Pour tester le fonctionnement, je vous propose de télécharger une police sur fontsquirrel, par exemple Learning Curve Pro. Cliquez sur "@font-face Kit", cela vous permettra de télécharger un kit prêt à l'emploi avec tous les formats de police.

Votre fichier CSS ressemblera à ceci au final :

Code : CSS

```
@font-face { /* Définition d'une nouvelle police nommée
LearningCurveProRegular */
    font-family: 'LearningCurveProRegular';
    src: url('LearningCurve_OT-webfont.eot');
    src: url('LearningCurve_OT-webfont.eot?#iefix')
format('embedded-opentype'),
         url('LearningCurve_OT-webfont.woff') format('woff'),
         url('LearningCurve_OT-webfont.ttf') format('truetype'),
         url('LearningCurve_OT-webfont.svg#LearningCurveProRegular')
format('svg');
}

h1 /* Utilisation de la police qu'on vient de définir sur les
titres */
{
    font-family: 'LearningCurveProRegular', Arial, serif;
}
```

La première (grosse) section `@font-face` permet de définir un nouveau nom de police qui pourra être utilisé dans le fichier CSS. Ensuite, nous utilisons ce nom de police avec la propriété `font-family` que nous connaissons, pour modifier l'apparence des titres `<h1>`. Et voilà le résultat :



Essayer !

Vous noterez quelques bizarreries dans le CSS généré par le site fontsquirrel. Le but est de pallier certains bugs sur Internet Explorer, car les anciennes versions ne comprennent pas quand on définit plusieurs formats, d'où la présence d'un `?#iefix` dans le code.

Italique, gras, souligné...

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

Mettre en italique

Attends attends là ! Je croyais que la balise **** 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 **** était faite pour mettre le texte en italique (de même que je n'ai jamais dit que **** était fait pour mettre en gras).

****, 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 choisissent d'afficher le texte en italique (mais ce n'est pas une obligation).

Le CSS lui, permet de dire réellement : "Je veux que ce texte soit en italique". Rien ne vous empêche par exemple de décider que tous vos titres seront en italique.

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 aussi mis en italique (en penchant les lettres).
- `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 **** 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

```
h2
{
    font-style: italic;
}
```

Mettre en gras

Et si nous passions à la mise en gras ?
Alors là, pareil, n'oubliez pas que ce n'est pas **** qui permet de mettre en gras (son rôle est d'indiquer que le texte est important, *donc* le navigateur l'affiche en gras). 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).

Voici par exemple comment écrire les titres en gras :

Code : CSS

```
h1
{
    font-weight: bold;
}
```

Soulignement et autres décorations

Cette propriété CSS porte bien son nom : **text-decoration**. Elle permet entre autres de souligner le texte, mais pas seulement. Voici les différentes valeurs qu'elle peut prendre :

- `underline` : souligné.
- `line-through` : barré.
- `overline` : ligne au-dessus.
- `blink` : clignotant. Ne marche pas sur tous les navigateurs (Internet Explorer et Google Chrome notamment).
- `none` : normal (par défaut).

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

Code : CSS

```
h1
{
    text-decoration: blink;
}
.souligne
{
    text-decoration: underline;
}
.barre
{
    text-decoration: line-through;
}
.ligne_dessus
{
    text-decoration: overline;
}
```

Essayer !

L'alignement

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;
}

p
{
    text-align: justify;
}

.signature
{
    text-align: right;
}
```

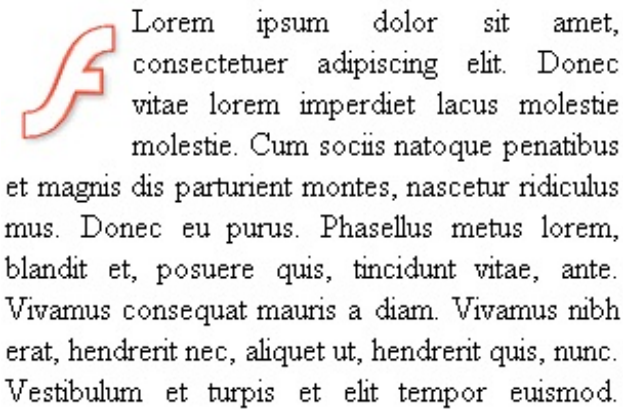
Essayer !

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>`, `<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.

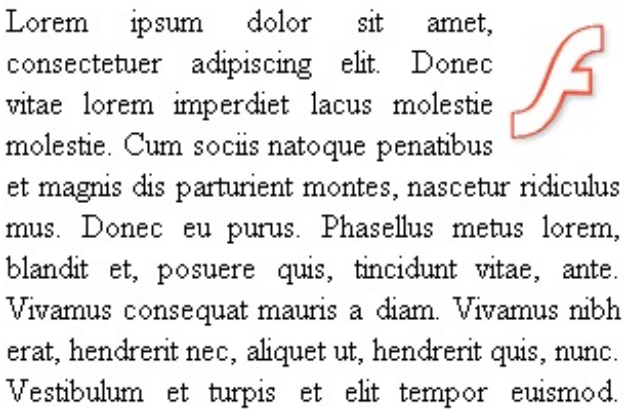
Les flottants

Le CSS nous permet de faire flotter un élément autour du texte. On dit aussi qu'on fait un "habillage".

Pour que vous voyiez bien de quoi on parle, voici ce que nous allons apprendre à faire :



Une image flottant à gauche



Une image flottant à droite

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 très simples :

- `left` : l'élément flottera à gauche.
- `right` : l'élément flottera... à droite ! Oui bravo.

L'utilisation des flottants est très simple :

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. Il est courant de faire flotter une image pour qu'elle soit habillée par du texte, comme sur les exemples précédents.

Faire flotter une image

Nous allons apprendre ici à faire flotter une image. Voici le code HTML que nous devons taper dans un premier temps :

Code : HTML

```
<p><img src="flash.gif" class="imageflottante" alt="Image flottante"
/> Ceci est un texte normal de paragraphe, écrit à la suite de
l'image et qui l'habillera car l'image est flottante.</p>
```

Vous devez placer l'élément flottant en premier dans le code HTML. Si vous placez l'image après le paragraphe, l'effet ne fonctionnera pas.

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;
}
```

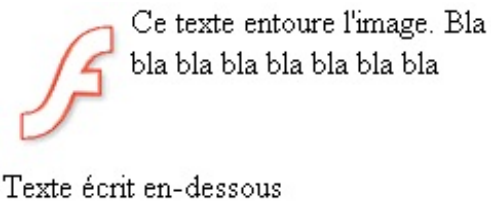
Essayer !

Amusez-vous aussi à faire flotter l'image à droite, c'est tout bête : il suffit d'indiquer la valeur `right` et le tour est joué !

Stopper un flottant

Quand vous mettez en place un flottant, le texte autour l'habille. 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 ne serait ni élégant ni très propre...

En gros, on aimerait pouvoir faire ça :



Il existe en fait une propriété CSS 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 prendre ces trois 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 donc). Pour illustrer son fonctionnement, on va prendre ce code HTML :

Code : HTML

```
<p><img src="flash.gif" class="imageflottante" alt="Image flottante"
/></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>
```

Et ce code CSS :

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 !
Vous êtes rentrés en plein dans le bain du CSS. Vous avez découvert vos premières propriétés CSS !

Dès que vous vous sentez en forme, rendez-vous au chapitre suivant pour découvrir encore de nombreuses propriétés CSS. 

La couleur et le fond

Continuons notre tour d'horizon des propriétés CSS existantes. Nous allons nous intéresser ici aux propriétés liées de près ou de loin à la couleur. Nous verrons entre autres :

- Comment changer la couleur du texte
- Comment mettre une couleur ou une image de fond
- Comment ajouter des ombres
- Comment jouer avec les niveaux de transparence

Le CSS n'a pas fini de nous étonner !

Couleur du texte

Passons maintenant au vaste sujet de la couleur.

Comment ça vaste ?

Vous connaissez déjà la propriété qui permet de modifier la couleur du texte : il s'agit de `color`. Nous allons nous intéresser aux différentes façons d'indiquer la couleur, car il y en a plusieurs.

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	
silver	
grey	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	

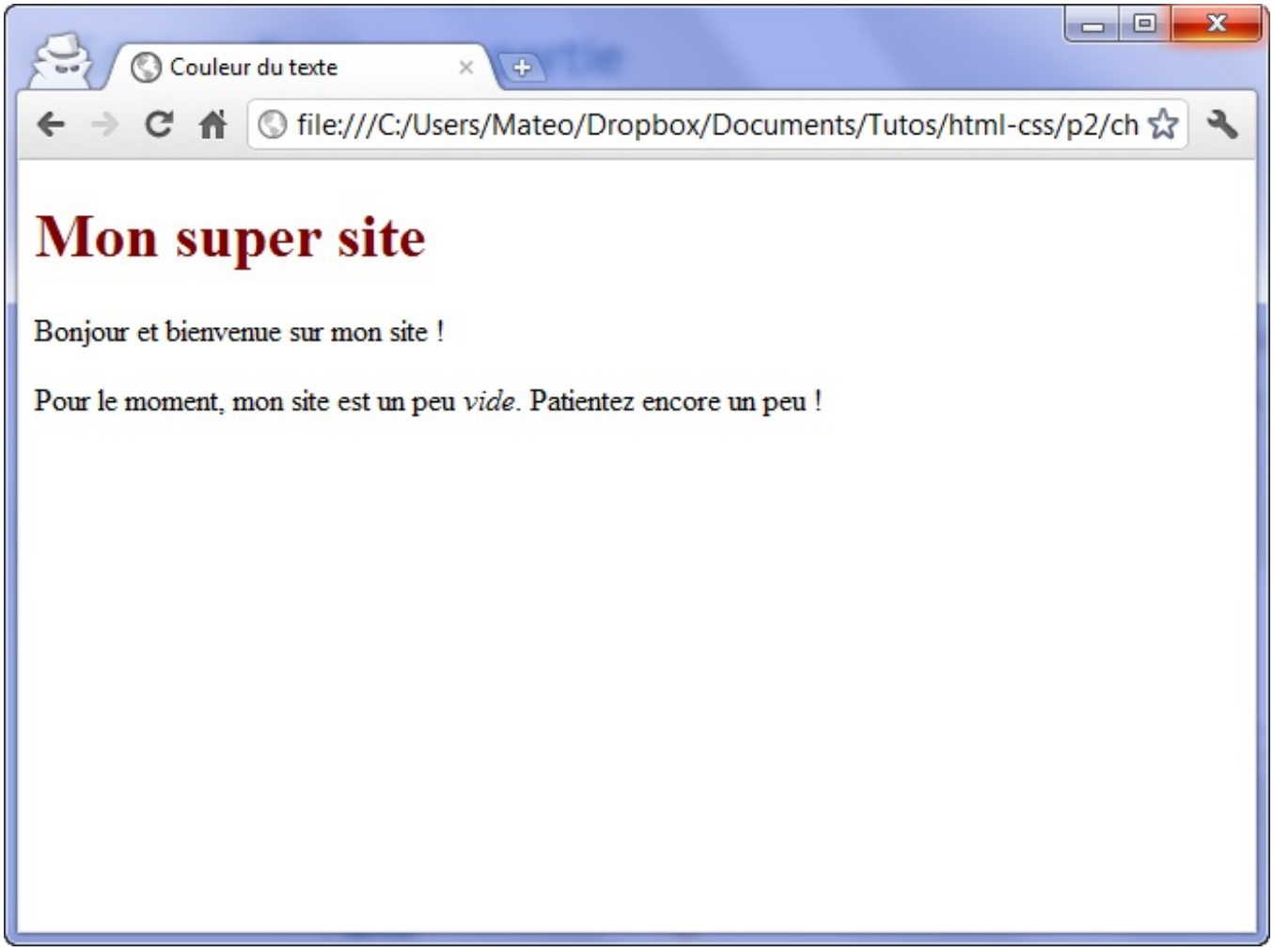
aqua	
teal	

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

Pour passer tous les titres en marron, on peut donc écrire :

Code : CSS

```
h1
{
    color: maroon;
}
```



Essayer !

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. Elle est couramment utilisée sur le Web, mais il existe aussi une autre méthode que nous verrons plus loin.

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.


Certains logiciels de dessin, comme Photoshop, Gimp et Paint .NET, vous indiquent les couleurs en hexadécimal. Il vous est 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 sur les paragraphes :

Code : CSS

```
p
{
    color: #FFFFFF;
}
```


Notez qu'il existe une notation raccourcie : on peut écrire une couleur avec seulement 3 caractères. Par exemple : #FA3 est équivalent à écrire #FFAA33.

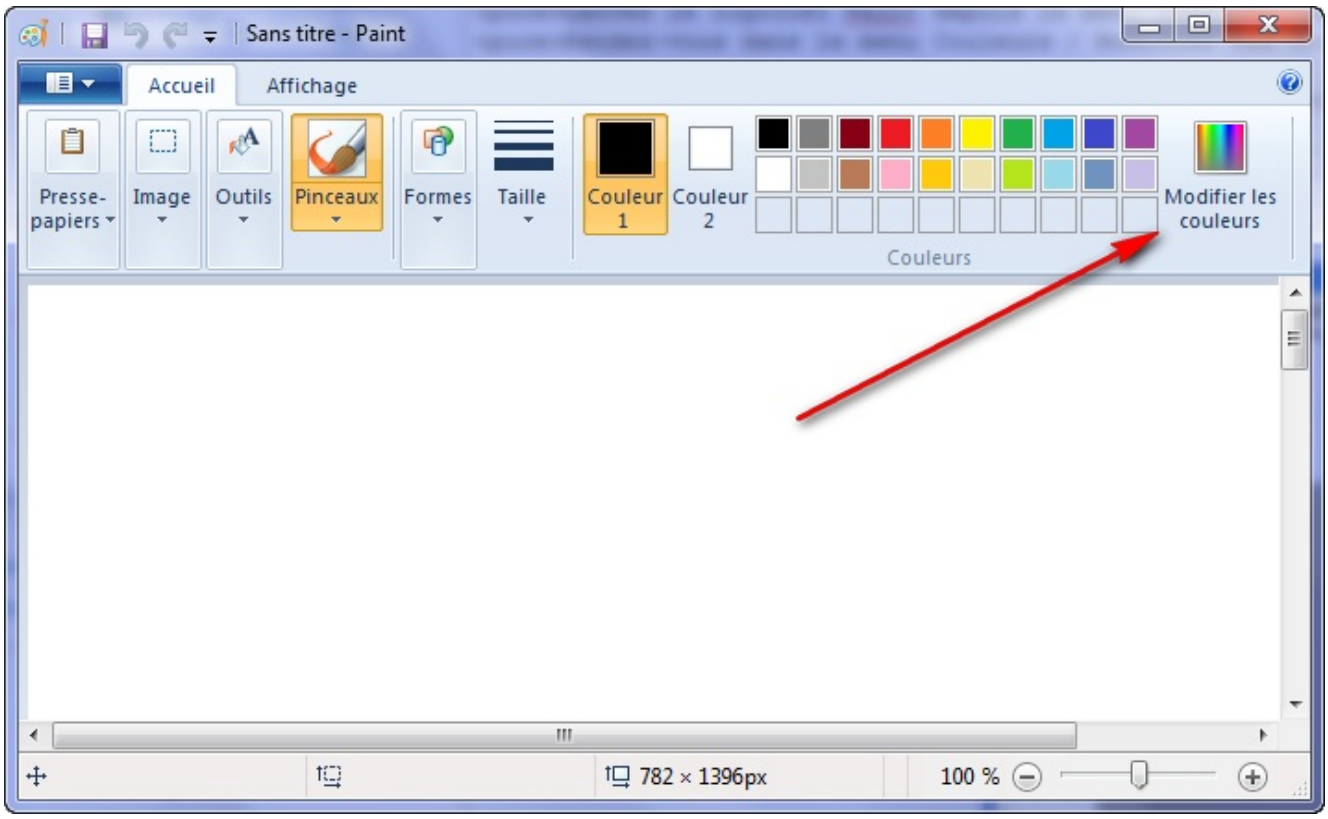
La méthode RGB

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.

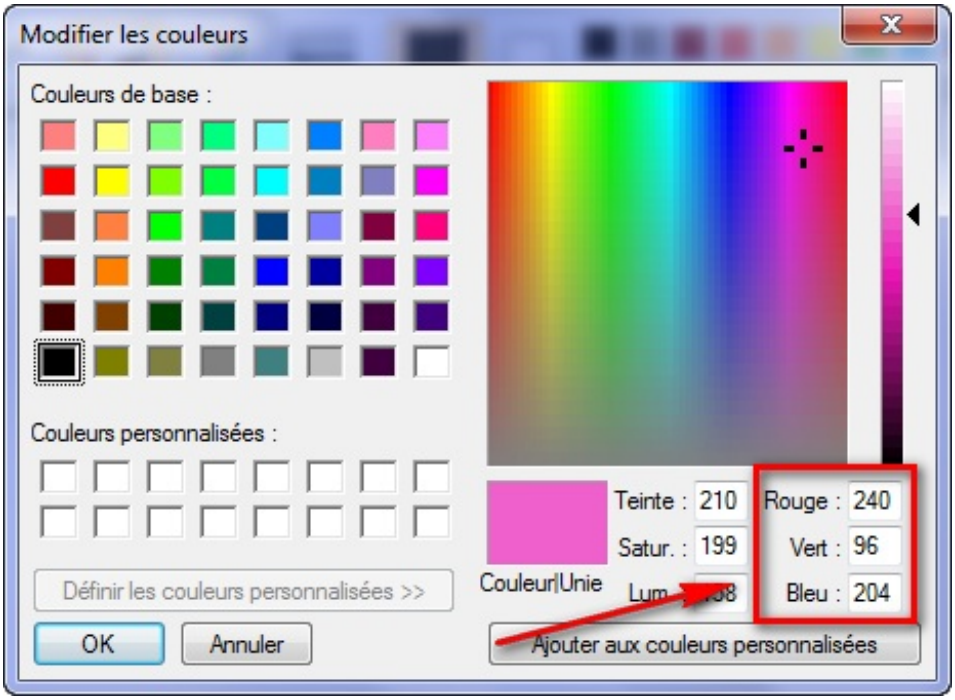

Encore cette histoire tordue de quantités de rouge-vert-bleu ?

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 la section *Modifier les couleurs* :



3. Une fenêtre s'ouvre. Dans la zone qui apparaît à droite, faites bouger les curseurs pour sélectionner la couleur qui vous intéresse. 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 :



4. On relève les quantités de Rouge-Vert-Bleu correspondantes indiquées en bas à droite de la fenêtre (ici 240-96-204). Je recopie ces valeurs dans cet ordre dans le fichier CSS, comme ceci :

Code : CSS

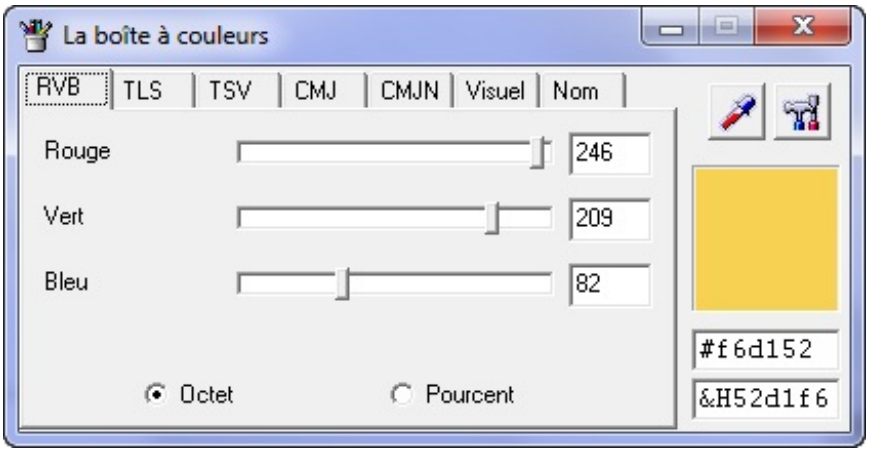
```
p
{
    color: rgb(240,96,204);
```

```
}
```

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 information, ces quantités sont toujours comprises entre 0 et 255.

Et en Bonus Track...

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



Télécharger "La Boîte à Couleurs" (1,45 Mo)

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.

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 la pipette en haut à droite pour récupérer n'importe quelle couleur s'affichant sur votre écran !
Amusez-vous bien !

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 demandé à ce que le texte de la balise **<body>** soit écrit en blanc, 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**. Je vous rassure tout de suite, personne n'est mort.

Le CSS et l'héritage

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

C'est d'ailleurs de là que vient le nom "CSS", qui signifie "Cascading Style Sheets", c'est-à-dire "Feuilles de style en cascade". Les propriétés CSS sont héritées en cascade : si vous donnez un style à un élément, tous les sous-éléments auront le même style.

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 mise en gras dans la balise **<body>**, et tous vos titres et paragraphes seront en gras.

Exemple d'héritage avec la balise <mark>

On a tendance à croire qu'on ne peut modifier que la couleur de fond de la page. C'est faux : vous pouvez changer le fond de n'importe quel élément : vos titres, vos paragraphes, certains mots... Dans ce cas, ils apparaîtront surlignés (comme si on avait mis un coup de marqueur dessus).

Vous vous souvenez par exemple de la balise **<mark>** qui permet de mettre en valeur certains mots ? Utilisons-la à nouveau ici :

Code : HTML

```
<h1>Qui a éteint la lumière ?</h1>

<p>Brr, il fait tout noir sur ce site, c'est un peu
<mark>inquiétant</mark> comme ambiance non vous trouvez pas ?</p>
```

Par défaut, le texte s'affiche sur un fond jaune. Vous pouvez changer ce comportement en CSS :

Code : CSS

```
body
{
   background-color: black;
   color: white;
}

mark
{
    /* La couleur de fond est prioritaire à celle de toute la page
*/
    background-color: red;
}
```

Sur le texte de la balise **<mark>**, c'est la couleur de fond rouge qui s'applique. En effet, même si le fond de la page est noir, c'est la propriété CSS de l'élément le plus précis qui a la priorité.

Essayer !

Le même principe vaut pour toutes les balises HTML et toutes les propriétés CSS ! Si vous dites :

- Mes paragraphes ont une taille de 1.2em
- Mes textes importants (`<strong>`) ont une taille de 1.4em

... on pourrait penser qu'il y a un conflit. Le texte important fait partie d'un paragraphe, quelle taille lui donner ? 1.2em ou 1.4em ? Le CSS décide que c'est la déclaration la plus précise qui l'emporte : comme `<strong>` correspond à un élément plus précis que les paragraphes, le texte sera écrit en 1.4em.

Images de fond

Dans les exemples qui suivent, je vais modifier l'image de fond de la page. Cependant, tout comme pour la couleur de fond, n'oubliez pas que l'image de fond ne s'applique pas forcément à la page entière. On peut aussi mettre une image de fond aux titres, paragraphes, etc.

Appliquer une image de fond

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

Code : CSS

```
body
{
    background-image: url("neige.png");
}
```

Essayer !

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 écrite en absolu (http://...) ou en relatif (fond.png).

Attention lorsque vous écrivez 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`. Je vous conseille de placer l'image de fond dans le même dossier que le fichier `.css` pour simplifier les choses (ou dans un sous-dossier).

Options disponibles pour l'image de fond

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

background-attachment : fixer le fond

La propriété CSS **`background-attachment`** permet de "fixer" le fond. L'effet obtenu est intéressant, car on voit alors le texte "glisser" par-dessus le fond. Deux valeurs sont disponibles :

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

Code : CSS

```
body
{
    background-image: url("neige.png");
    background-attachment: fixed; /* Le fond restera fixe */
}
```

background-repeat : répétition du fond

Par défaut, l'image de fond est répétée en mosaïque. Vous pouvez changer cela avec la propriété **background-repeat** :

- `no-repeat` : le fond ne sera pas répété. L'image sera donc unique sur la page.
- `repeat-x` : le fond sera répété uniquement sur la première ligne, horizontalement.
- `repeat-y` : le fond sera répété uniquement sur la première colonne, verticalement.
- `repeat` : le fond sera répété en mosaïque (par défaut).

Exemple d'utilisation :

Code : CSS

```
body
{
    background-image: url("soleil.png");
    background-repeat: no-repeat;
}
```

background-position : position du fond

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

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

Code : CSS

```
background-position: 30px 50px;
```

... votre fond sera placé à 30 pixels de la gauche et à 50 pixels du haut. Il est aussi possible d'utiliser 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 :

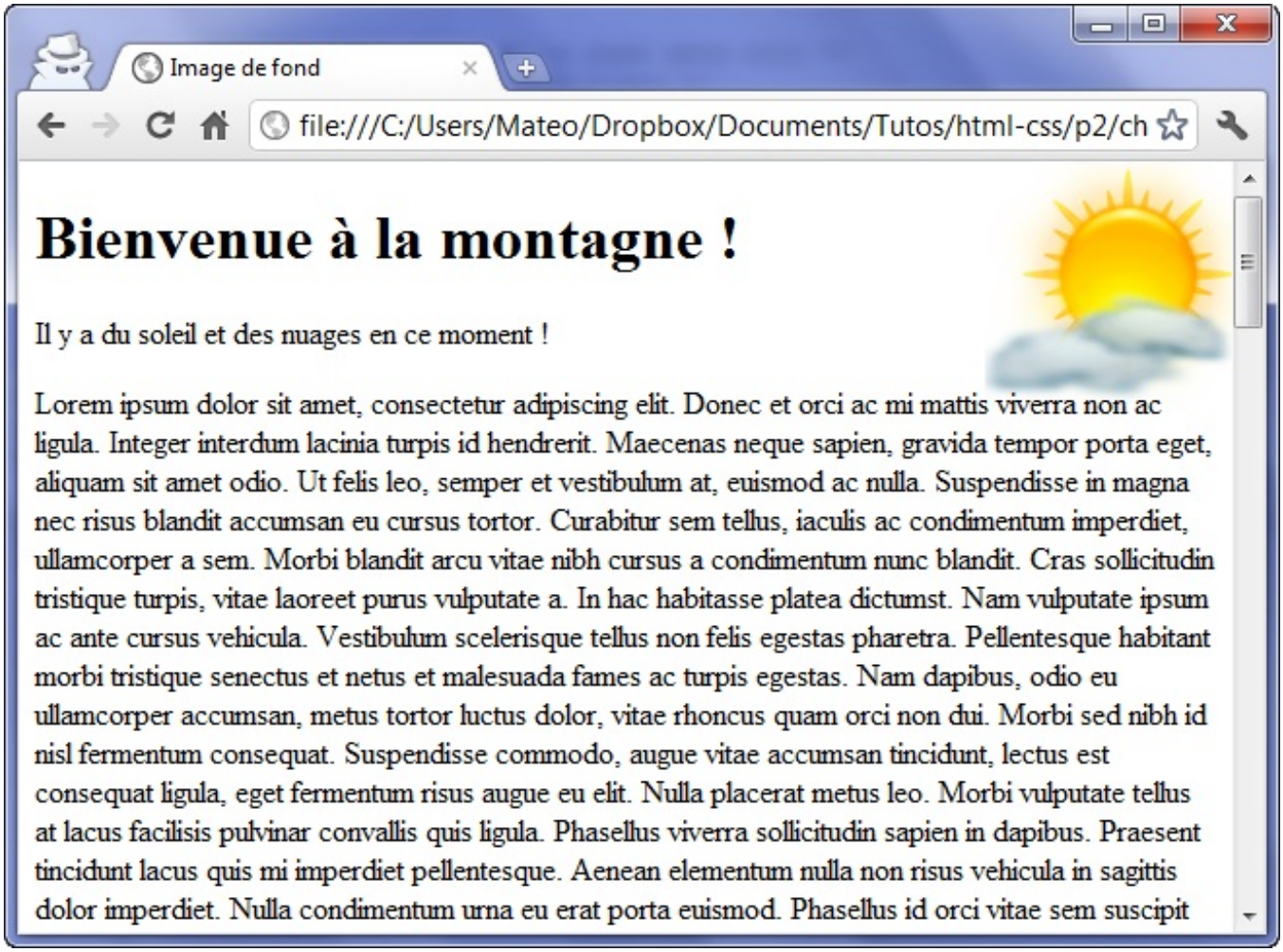
Code : CSS

```
background-position: top right;
```

Ainsi, si je veux afficher un soleil en image de fond, en un unique exemplaire (no-repeat), toujours visible (fixed) et positionné en haut à droite (top right), je vais écrire ceci :

Code : CSS

```
body
{
    background-image: url("soleil.png");
    background-attachment: fixed; /* Le fond restera fixe */
    background-repeat: no-repeat;  /* Le fond ne sera pas répété */
    background-position: top right; /* Le fond sera placé en haut à
droite */
}
```



Essayer !

Combiner les propriétés

Si vous utilisez beaucoup de propriétés en rapport avec le fond (comme c'est le cas sur ce dernier exemple), vous pouvez utiliser une sorte de "super-propriété" appelée **background** qui peut prendre plusieurs valeurs combinées des propriétés vues précédemment : **background-image**, **background-repeat**, **background-attachment** et **background-**

position.

On peut donc tout simplement écrire :

Code : CSS

```
body
{
    background: url("soleil.png") fixed no-repeat top right;
}
```

C'est la première "super-propriété" que je vous montre, il y en aura d'autres. Il faut savoir que :

- L'ordre des valeurs n'a pas d'importance. Vous pouvez combiner les valeurs dans n'importe quel ordre.
- Vous n'êtes pas obligés de mettre toutes les valeurs. Ainsi, si vous ne voulez pas écrire fixed, vous pouvez l'enlever sans problème.

Plusieurs images de fond

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

Code : CSS

```
body
{
    background: url("soleil.png") fixed no-repeat top right,
url("neige.png") fixed;
}
```

La première image de cette liste sera placée par-dessus les autres. Attention donc, l'ordre de déclaration des images a son importance : si vous inversez le soleil et la neige dans le code CSS précédent, vous ne verrez plus le soleil !

Essayer !

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

Une dernière chose avant d'en terminer avec les images de fond : 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 (contrairement à moi !) vous arriverez certainement à donner une très belle allure à votre page web. 

La transparence

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

La propriété opacity

La propriété `opacity`, très simple, permet d'indiquer le niveau d'opacité (c'est l'inverse de la transparence).

- Avec une valeur de 1, l'élément sera totalement opaque : c'est le comportement par défaut.
- Avec une valeur de 0, l'élément sera totalement transparent.

Il faut donc choisir une valeur comprise entre 0 et 1. Par exemple avec 0.6, votre élément sera opaque à 60%... et on verra donc à travers !

Voici comment on peut l'utiliser :

Code : CSS

```
p
{
    opacity: 0.6;
}
```

Voici un exemple qui va nous permettre d'apprécier la transparence :

Code : CSS

```
body
{
   background: url('neige.png');
}

p
{
    background-color: black;
    color: white;
    opacity: 0.3;
}
```

Essayer !

Notez que la transparence fonctionne sur tous les navigateurs récents, y compris Internet Explorer à partir de IE9.

Si vous appliquez la propriété **opacity** à un élément de la page, *tout* le contenu de cet élément sera rendu transparent (même les images, les autres blocs à l'intérieur, etc.). Si vous voulez juste rendre la couleur de fond transparente, utilisez plutôt la notation RGBa que l'on va découvrir.

La notation RGBa

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

Code : CSS

```
p
{
    background-color: rgba(255, 0, 0, 0.5); /* Fond rouge à moitié transparent */
}
```

C'est aussi simple que cela. Vous pouvez obtenir exactement le même effet qu'avec **opacity** juste en jouant avec la notation RGBa, essayez !

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

Code : CSS

```
p
{
    background-color: rgb(255,0,0); /* Pour les anciens navigateurs */
    background-color: rgba(255,0,0,0.5); /* Pour les navigateurs plus récents */
}
```

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, loin de là.

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 bordures et les ombres

Nouveau chapitre, nouveau lots de propriétés CSS. Ici, nous allons nous intéresser aux bordures et aux effets d'ombre que l'on peut appliquer, aussi bien sur le texte que sur les blocs qui constituent notre page.

Nous réutiliserons en particulier nos connaissances sur les couleurs pour choisir la couleur de nos bordures et de nos ombres.

Prêts à vous en mettre une nouvelle fois plein la vue ?

Bordures standard

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

Pour aller à l'essentiel, je vous propose ici d'utiliser directement la super-propriété **`border`** qui regroupe l'ensemble de ces propriétés. Vous vous souvenez de la super-propriété **`background`** ? Cela fonctionne sur le même principe : on va pouvoir combiner plusieurs valeurs.

Pour **`border`** on peut utiliser jusqu'à 3 valeurs pour modifier l'apparence de la bordure :

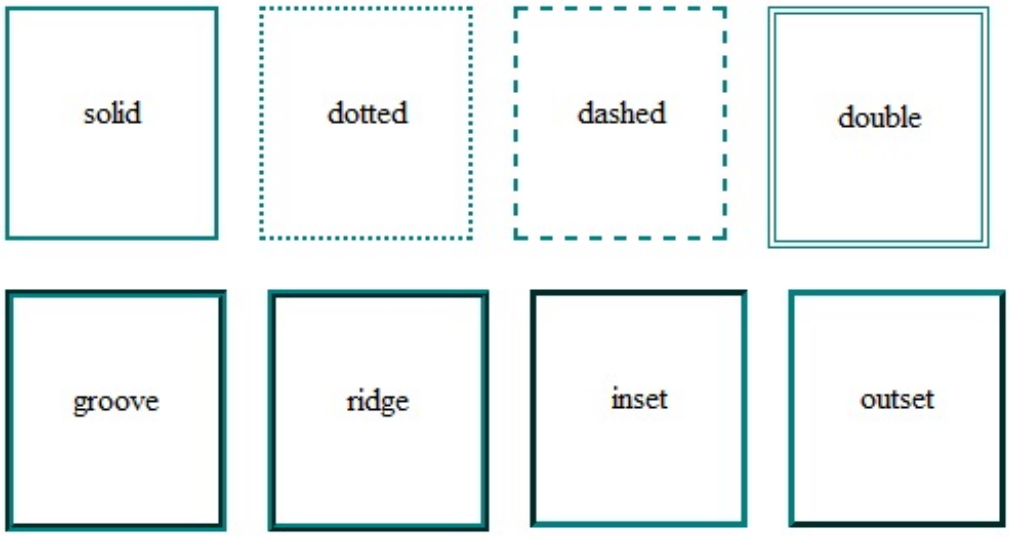
- **La largeur** : indiquez la largeur de votre bordure. Mettez une valeur en pixels (comme `2px`).
- **La couleur** : c'est la couleur de votre bordure. Utilisez, comme on l'a appris, soit un nom de couleur ("black", "red"...), soit une valeur hexadécimale (`#FF0000`), soit une valeur rgb (`rgb(198, 212, 37)`).
- **Le type de bordure** : là, vous avez le 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` : autre effet relief.
  - `inset` : effet 3D enfoncé.
  - `outset` : effet 3D surélevé.

Ainsi, pour avoir une bordure bleue en tirets de 3 pixels autour des mes titres, je vais écrire :

Code : CSS

```
h1
{
    border: 3px blue dashed;
}
```

Voici les différents styles de bordures en images pour vous aider à faire votre choix :



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 élément ?
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.

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

Ce sont aussi des super-propriétés, elles fonctionnent comme **`border`** mais ne s'appliquent donc qu'à un seul côté.

Pour ajouter une bordure uniquement à gauche et à droite des paragraphes, on écrira donc :

Code : CSS

```css
p
{
    border-left: 2px solid black;
    border-right: 2px solid black;
}
```

On peut modifier les bordures de n'importe quel type d'élément sur sa page. Nous l'avons fait ici sur les paragraphes, mais on peut aussi modifier la bordure de ses images, des textes importants comme **`<strong>`**, etc.

Bordures arrondies

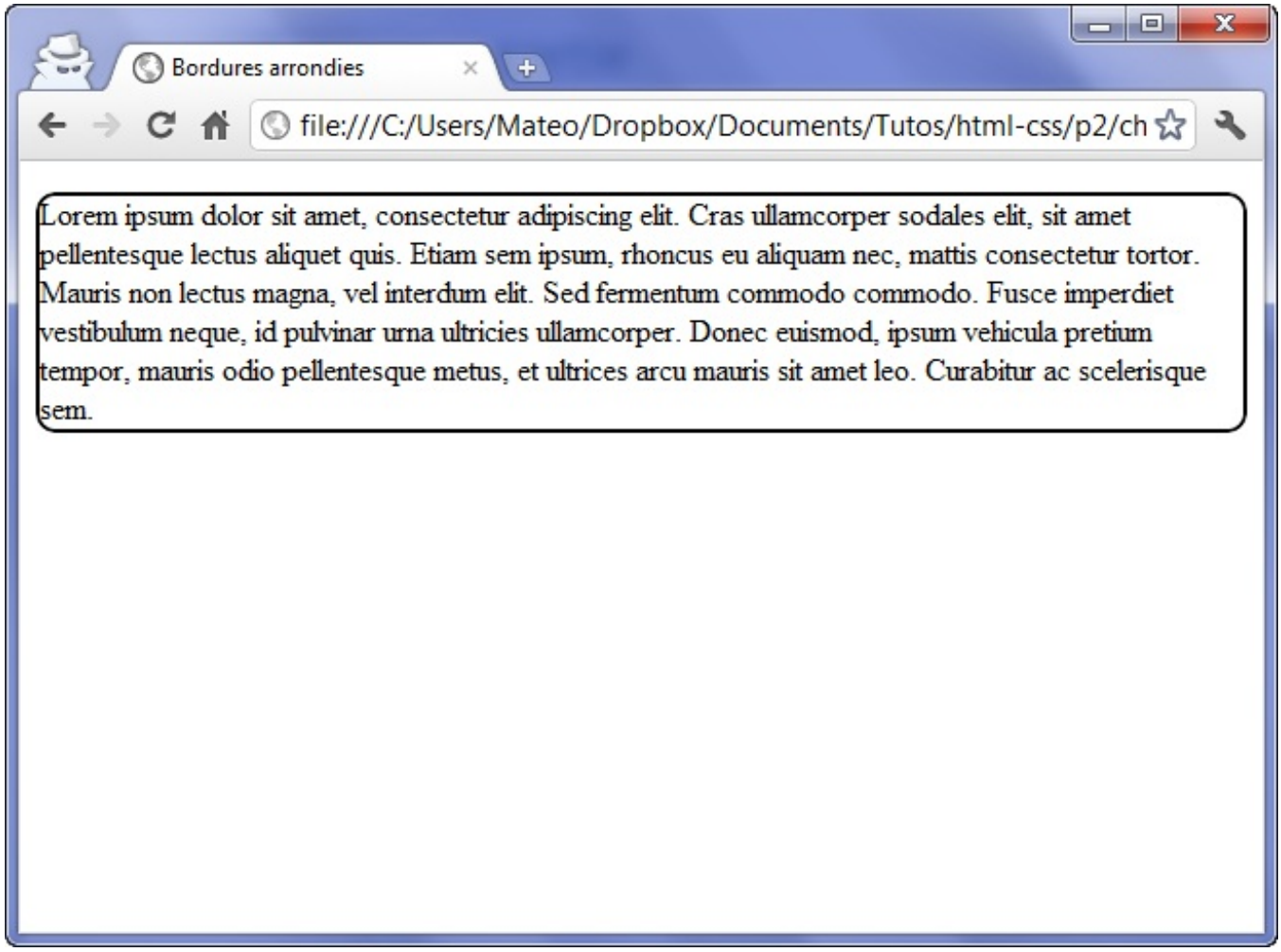
Les bordures arrondies, c'est un peu le Saint Graal attendu par les webmasters depuis des millénaires (ou presque). Depuis que CSS3 est arrivé, c'est enfin possible d'en créer facilement !

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

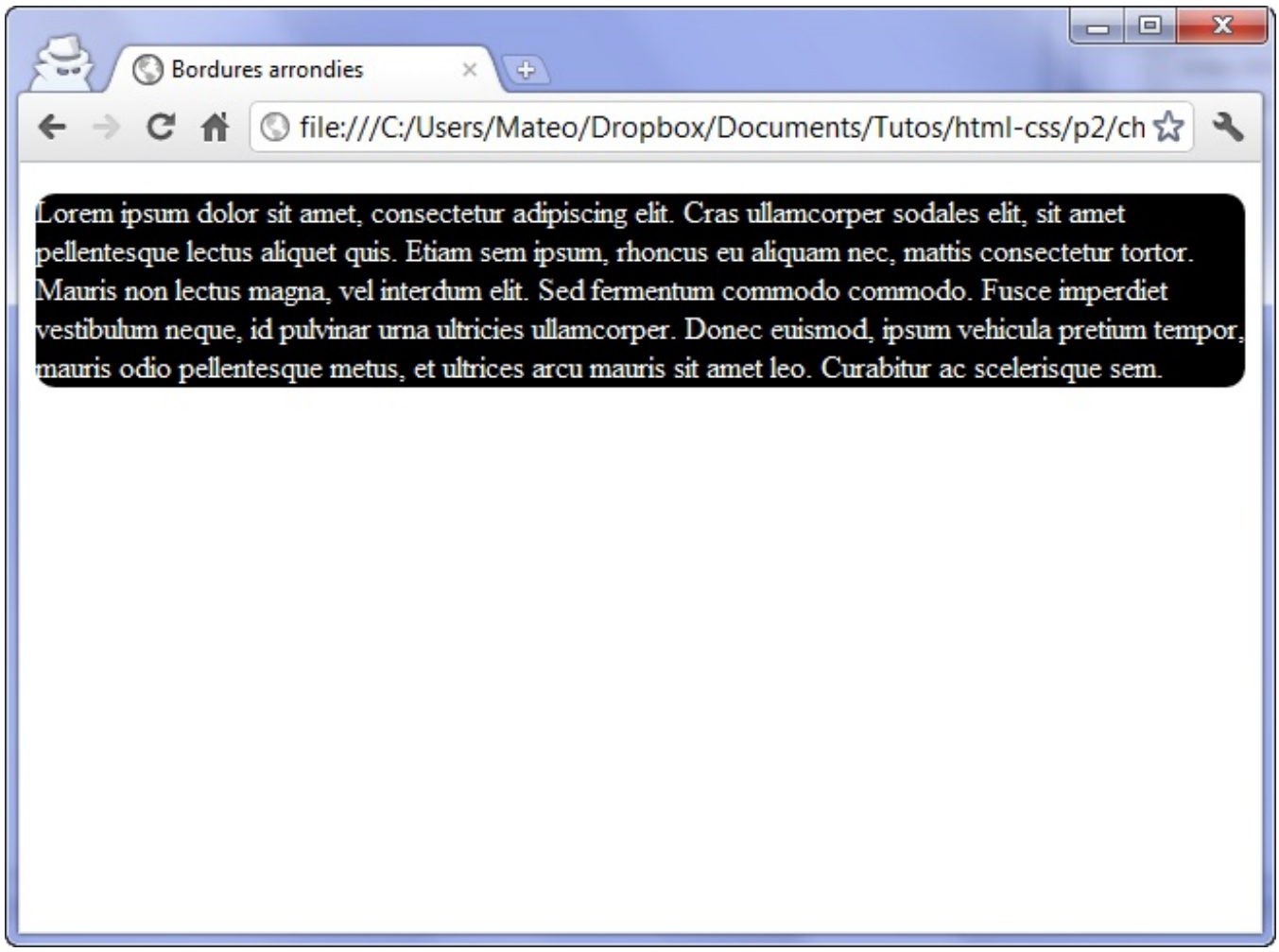
Code : CSS

```
p
{
    border-radius: 10px;
}
```

L'arrondi se voit notamment si l'élément a des bordures :



... ou s'il a une couleur de fond :



Essayer !

On peut aussi préciser la forme de l'arrondi pour chaque coin. Dans ce cas, indiquez 4 valeurs :

Code : CSS

```
p
{
    border-radius: 10px 5px 10px 5px;
}
```

Les valeurs correspondent aux angles suivants dans cet ordre :

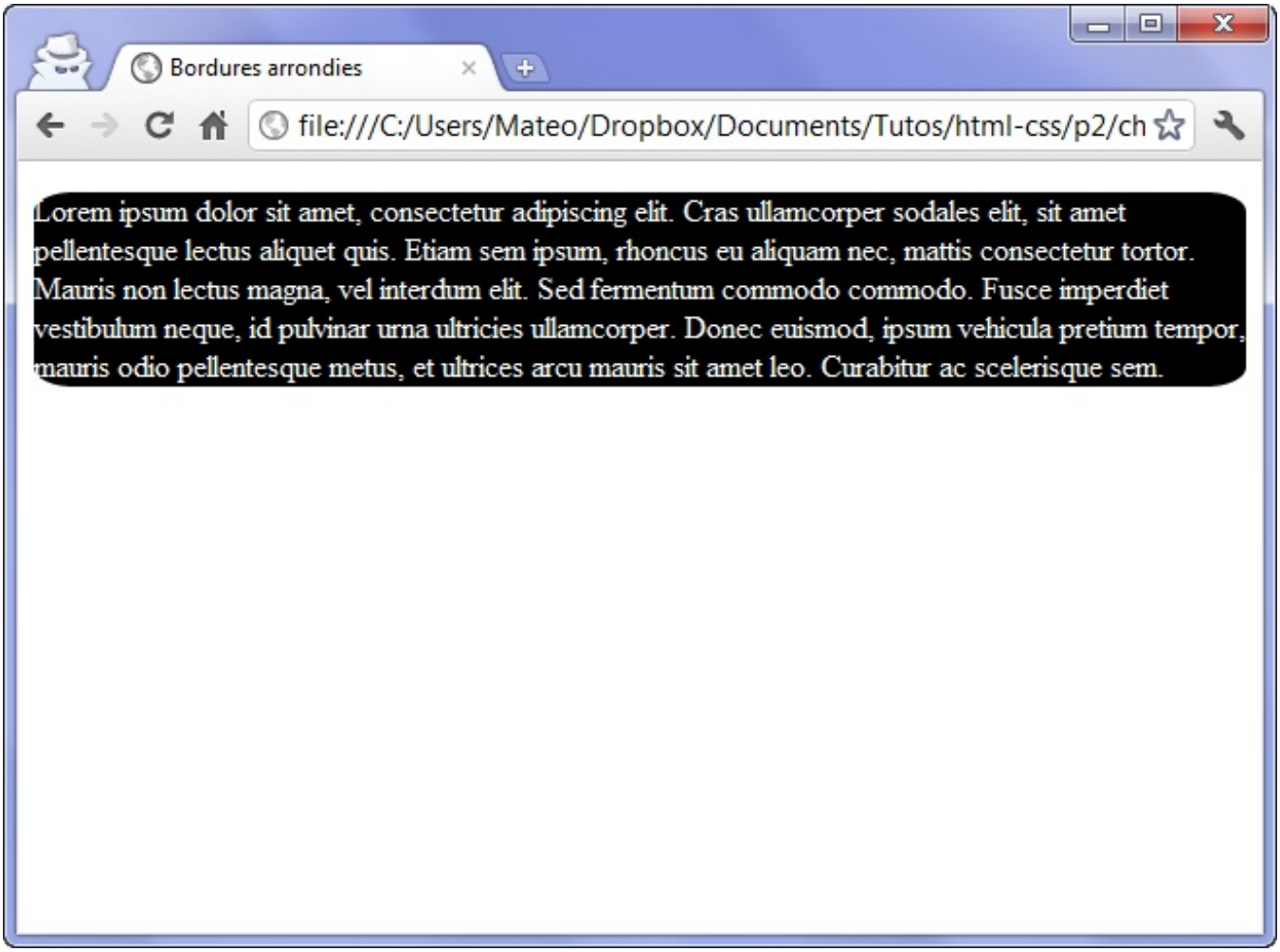
1. En haut à gauche
2. En haut à droite
3. En bas à droite
4. En bas à gauche

Enfin, il est possible d'affiner l'arrondi de nos angles en créant des courbes elliptiques. Il faut indiquer 2 valeurs séparées par un slash (/). Le mieux est certainement de tester pour voir l'effet :

Code : CSS

```
p
{
    border-radius: 20px / 10px;
```

```
}
```



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

Les ombres

Les ombres font partie des nouveautés récentes offertes par CSS3. Aujourd'hui, ajouter des ombres à ses pages se fait en une seule ligne de CSS !

Nous allons ici découvrir deux types d'ombres :

- Les ombres des boîtes
- Les ombres du texte

box-shadow : les ombres des boîtes

La propriété `box-shadow` s'applique à tout le bloc et prend 4 valeurs dans cet ordre :

1. Le décalage horizontal de l'ombre
2. Le décalage vertical de l'ombre

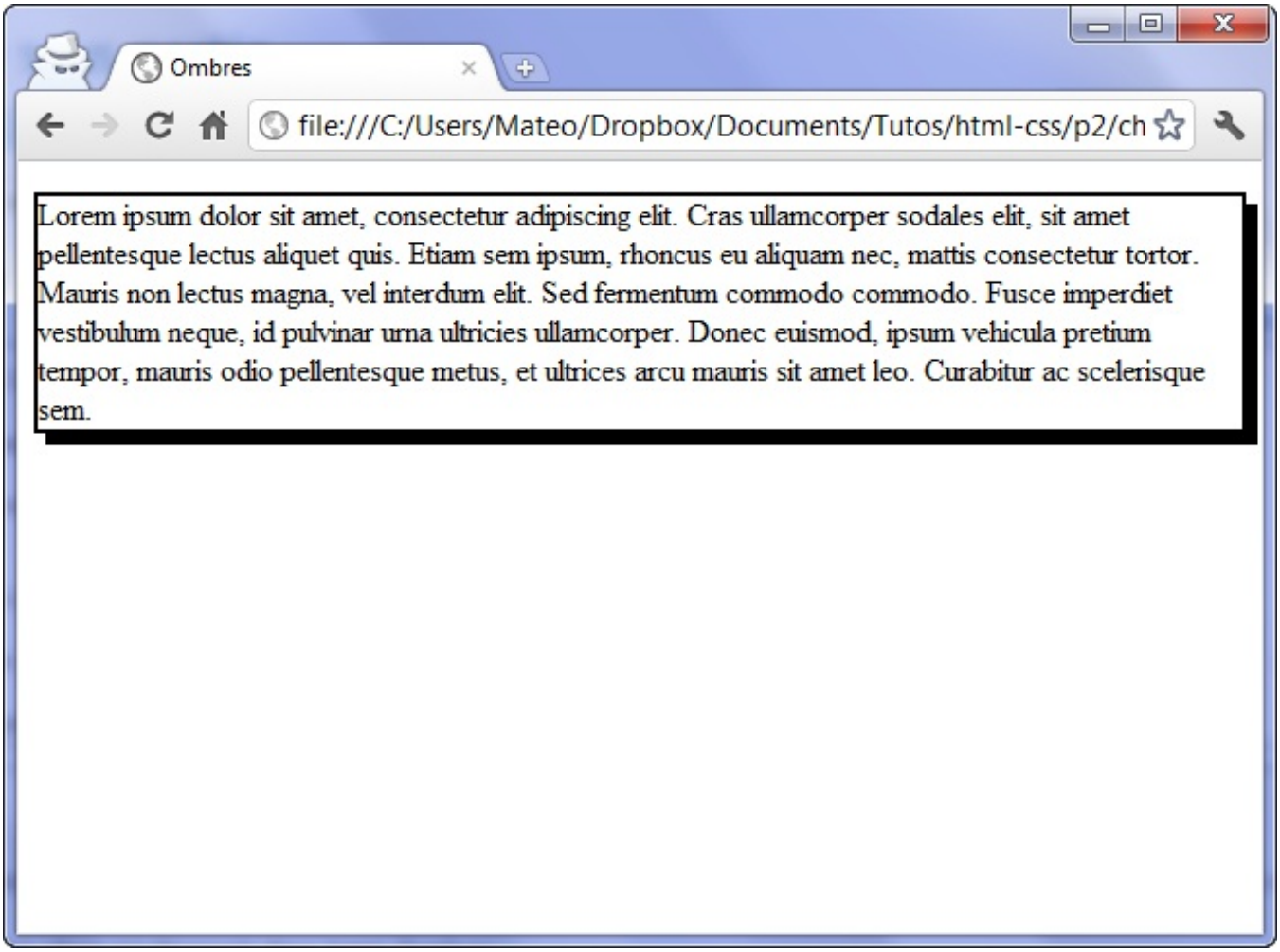
3. L'adoucissement du dégradé
4. La couleur de l'ombre

Par exemple, pour une ombre noire de 6 pixels, sans adoucissement, on écrira :

Code : CSS

```
p
{
    box-shadow: 6px 6px 0px black;
}
```

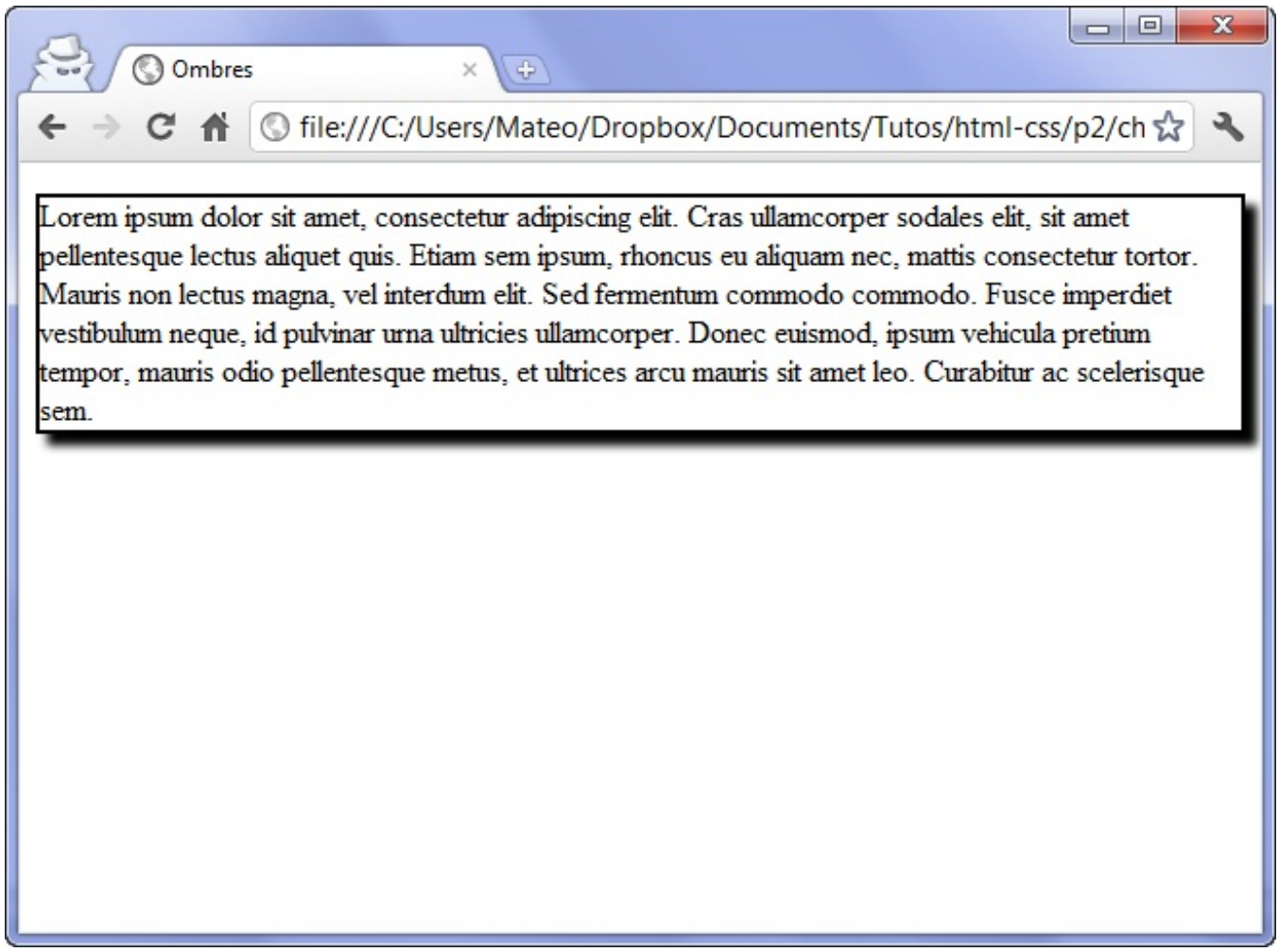
Ce qui donne (j'ai ajouté une bordure au paragraphe pour qu'on voit mieux l'effet) :



Ajoutons un adoucissement avec le troisième paramètre. L'adoucissement peut être faible (inférieur au décalage), normal (égal au décalage) ou élevé (supérieur au décalage). Essayons un décalage normal :

Code : CSS

```
p
{
    box-shadow: 6px 6px 6px black;
}
```



On peut aussi rajouter une quatrième valeur facultative : `inset`. Dans ce cas, l'ombre sera placée à l'intérieur du bloc, pour donner un effet enfoncé :

Code : CSS

```
p
{
    box-shadow: 6px 6px 6px black inset;
}
```

Je vous laisse essayer de voir le résultat.

La propriété **box-shadow** marche sur tous les navigateurs récents, IE9 inclus. Pour certains navigateurs, en particulier les navigateurs mobiles, il faut encore rajouter un préfixe. Ainsi, il faudra écrire une version **-webkit-box-shadow** pour que cela fonctionne sur les navigateurs Android et iOS.

text-shadow : l'ombre du texte

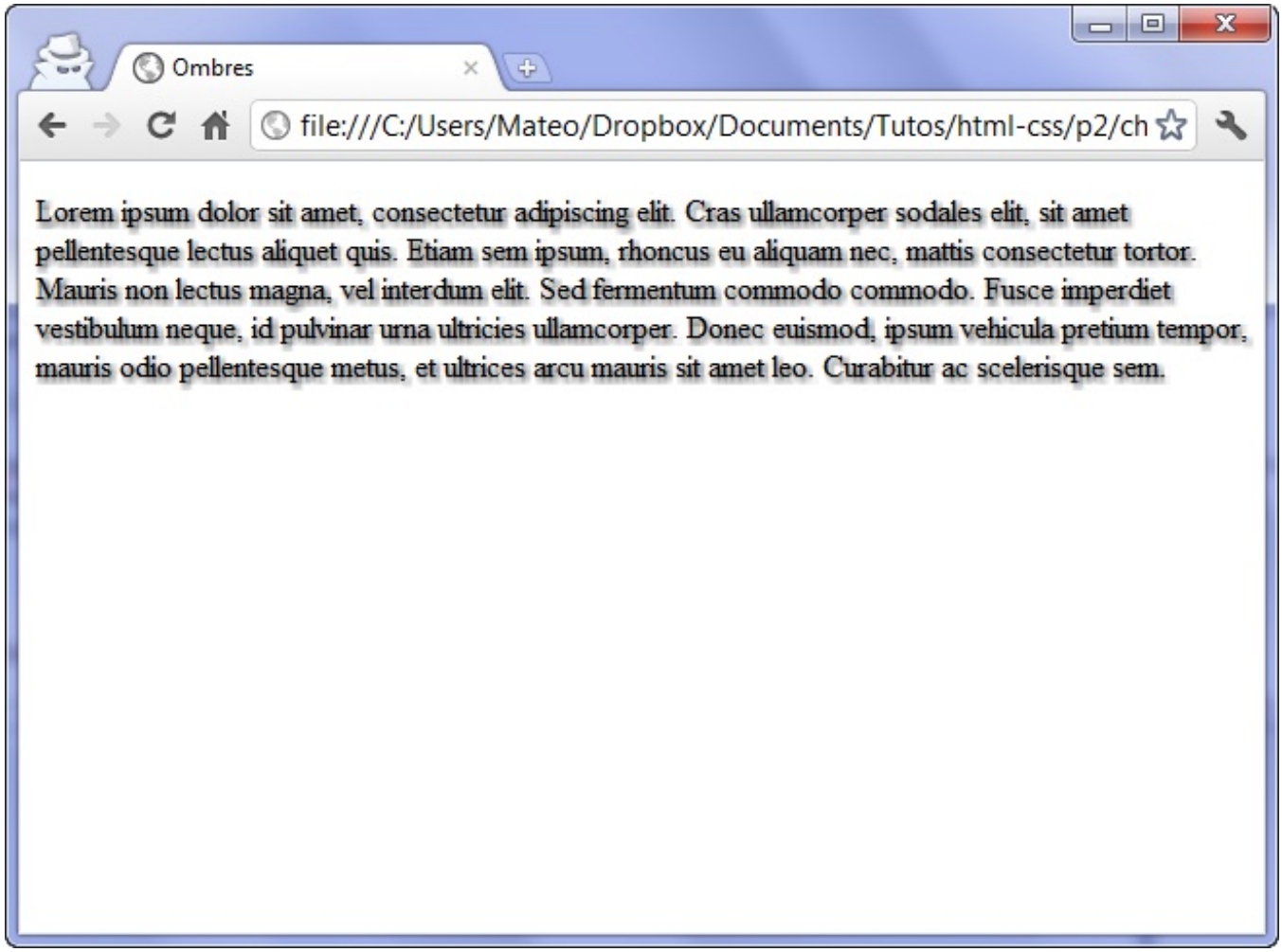
Avec **text-shadow**, vous pouvez ajouter une ombre directement sur les lettres de votre texte ! Les valeurs fonctionnent exactement de la même façon que **box-shadow** : décalage, adoucissement et couleur.

Code : CSS

```
p
{
```

```
        text-shadow: 2px 2px 4px black;
    }
```

Voilà le résultat :



Cette propriété est reconnue par tous les navigateurs récents, sauf Internet Explorer, qui ne la reconnaît qu'à partir de la version IE10.

Comme vous avez pu le constater, on peut aujourd'hui réaliser de nombreux effets sur nos sites web ! Il existe d'ailleurs un certain nombre d'autres effets, que je ne vous présenterai pas ici car un peu complexes et pas toujours très bien supportés (je pense par exemple à **`border-image`**, qui permet d'utiliser une image en guise de bordure, mais qui est encore un peu délicat à manipuler pour le moment !).

Création d'apparences dynamiques

C'est une de ses forces : le CSS nous permet aussi de modifier l'apparence des éléments de façon dynamique, c'est-à-dire que des éléments peuvent changer de forme une fois que la page a été chargée. Nous allons faire appel à une fonctionnalité puissante du CSS : les pseudo-formats.

Nous verrons dans ce chapitre comment changer l'apparence :

- Au survol
- Lors du clic
- Lors du focus (élément sélectionné)
- Lorsqu'un lien a été visité

Vous allez voir que le langage CSS n'a pas fini de nous étonner ! 

Au survol

Nous allons découvrir dans ce chapitre plusieurs pseudo-formats CSS. Le premier 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 classe) 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).

À 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 /* Liens par défaut (non survolés) */
{
   text-decoration: none;
   color: red;
   font-style: italic;
}

a:hover /* Apparence au survol des liens */
{
   text-decoration: underline;
   color: green;
}
```

On a défini ici 2 versions des styles pour les liens :

- Pour les liens par défaut (non survolés)
- Pour les liens au survol

Voici ce que ça donne :

Essayer !

Sympa, n'est-ce pas ?

Même si on l'utilise souvent sur les liens, vous pouvez modifier l'apparence de n'importe quel élément. Par exemple, vous pouvez modifier l'apparence des paragraphes lorsqu'on pointe dessus :

Code : CSS

```
p:hover /* Quand on pointe sur un paragraphe */
{

}
```

Au clic et lors de la sélection

Vous pouvez interagir encore plus finement en CSS. Nous allons voir ici que nous pouvons changer l'apparence des éléments lorsqu'on clique dessus et lorsqu'ils sont sélectionnés !

:active : au moment du clic

Le pseudo-format `:active` permet d'appliquer un style particulier *au moment du clic*. En pratique, il n'est utilisé que sur les liens.

Le lien gardera cette apparence très peu de temps : en fait, le changement apparaît lorsque le bouton de la souris est enfoncé. En clair, ce n'est pas forcément toujours bien visible.

On peut par exemple changer la couleur de fond du lien lorsqu'on clique dessus :

Code : CSS

```
a:active /* Quand le visiteur clique sur le lien */
{
    background-color: #FFCC66;
}
```

Essayer !

:focus : lorsque l'élément est sélectionné

Là, c'est un peu différent. Le pseudo-format `:focus` applique un style *lorsque l'élément est sélectionné*.

C'est-à-dire ?

Une fois que vous avez cliqué, le lien reste "sélectionné" (il y a une petite bordure en pointillés autour). C'est ça la sélection.

Ce pseudo-format pourra être appliqué à d'autres balises HTML que nous n'avons pas encore vues, comme les éléments de formulaires.

Essayons pour l'instant sur les liens :

Code : CSS

```
a:focus /* Quand le visiteur sélectionne le lien */
{
    background-color: #FFCC66;
}
```

Essayer !

Sous Google Chrome et Safari, l'effet ne se voit que si on appuie sur la touche Tab.

Lorsque le lien a déjà été visité

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 assez laid (de mon point de vue du moins !).

Vous pouvez changer cette apparence avec `:visited` (qui signifie "visité"). En pratique, on ne peut pas changer beaucoup de choses à part la couleur sur les liens visités.

Code : CSS

```
a:visited /* Quand le visiteur a déjà vu la page concernée */
{
    color: #AAA; /* Appliquer une couleur grise */
}
```

Essayer !

Si vous ne souhaitez pas que les liens déjà visités soient colorés d'une façon différente, il vous faudra leur appliquer la même couleur qu'aux liens normaux. De nombreux sites web font cela (le Site du Zéro y compris !). Une exception notable : Google... ce qui est plutôt pratique, puisqu'on peut voir dans les résultats de ses recherches si on a déjà visité ou non les sites que Google nous présente.

La plupart du temps, les pseudo-formats que nous venons de voir sont utilisés sur les liens. On peut même jouer avec ces fonctionnalités pour créer des menus déroulants uniquement en CSS !

Partie 3 : Mise en page du site

Nous avons appris à construire des pages basiques en HTML, à modifier la mise en forme avec CSS... Intéressons-nous maintenant à la mise en page de notre site ! A la fin de cette partie, nous aboutirons à notre premier site complet, agencé comme nous le voulons !

Structurer sa page

Nous approchons de plus en plus du but. Si nos pages web ne ressemblent pas encore tout à fait aux sites web que nous connaissons, c'est qu'il nous manque les connaissances nécessaires pour faire la mise en page.

En général, une page web est constituée d'un en-tête (tout en haut), de menus de navigation (en haut ou sur les côtés), de différentes sections au centre... et d'un pied de page (tout en bas).

Nous allons nous intéresser dans ce chapitre aux nouvelles balises HTML dédiées à la structuration du site. Ces balises ont été introduites par HTML5 (elles n'existaient pas avant) et vont nous permettre de dire : "Ceci est mon en-tête", "Ceci est mon menu de navigation", etc.

Nous n'allons pas faire de mise en page pour le moment. Nous allons en fait préparer notre document HTML pour pouvoir découvrir la mise en page dans les prochains chapitres.

Les balises structurantes de HTML5

Je vais vous présenter ici les nouvelles balises introduites par HTML5 pour structurer nos pages. Vous allez voir, cela ne va pas beaucoup changer l'apparence de notre site pour le moment, mais il sera bien construit et prêt à être mis en forme ensuite !

<header> : l'en-tête

La plupart des sites web possèdent en général un en-tête, appelé *header* en anglais. On y trouve le plus souvent un logo, une bannière, le slogan de votre site...

Vous devrez placer ces informations à l'intérieur de la balise **<header>** :

Code : HTML

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

Sur le site web Amazon par exemple, l'en-tête correspond à ceci :

L'en-tête peut contenir tout ce que vous voulez : images, liens, textes...

Il peut y avoir plusieurs en-têtes dans votre page. Si celle-ci est découpée en plusieurs sections, chaque section peut en effet avoir son propre **<header>**.

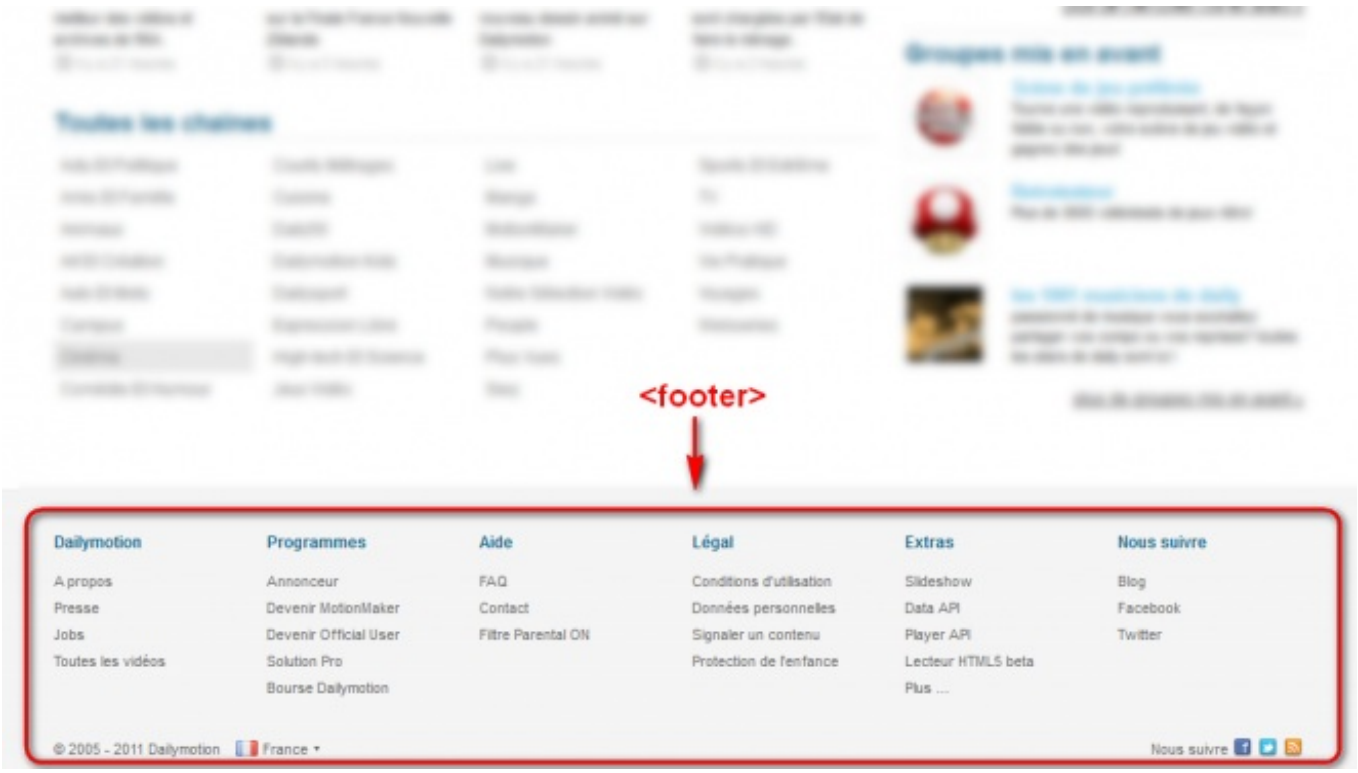
<footer> : le pied de page

A l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document. On y trouve des informations comme des liens de contact, le nom de l'auteur, les mentions légales, etc.

Code : HTML

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

Voici à quoi ressemble le pied de page du site Dailymotion :



<nav> : principaux liens de navigation

La balise **<nav>** doit regrouper tous les principaux liens de navigation du site. Vous y placerez par exemple le menu principal de votre site.

Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise **<nav>** :

Code : HTML

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

Voici le menu principal de navigation du site d'Amazon qui pourrait utiliser la balise **<nav>** :

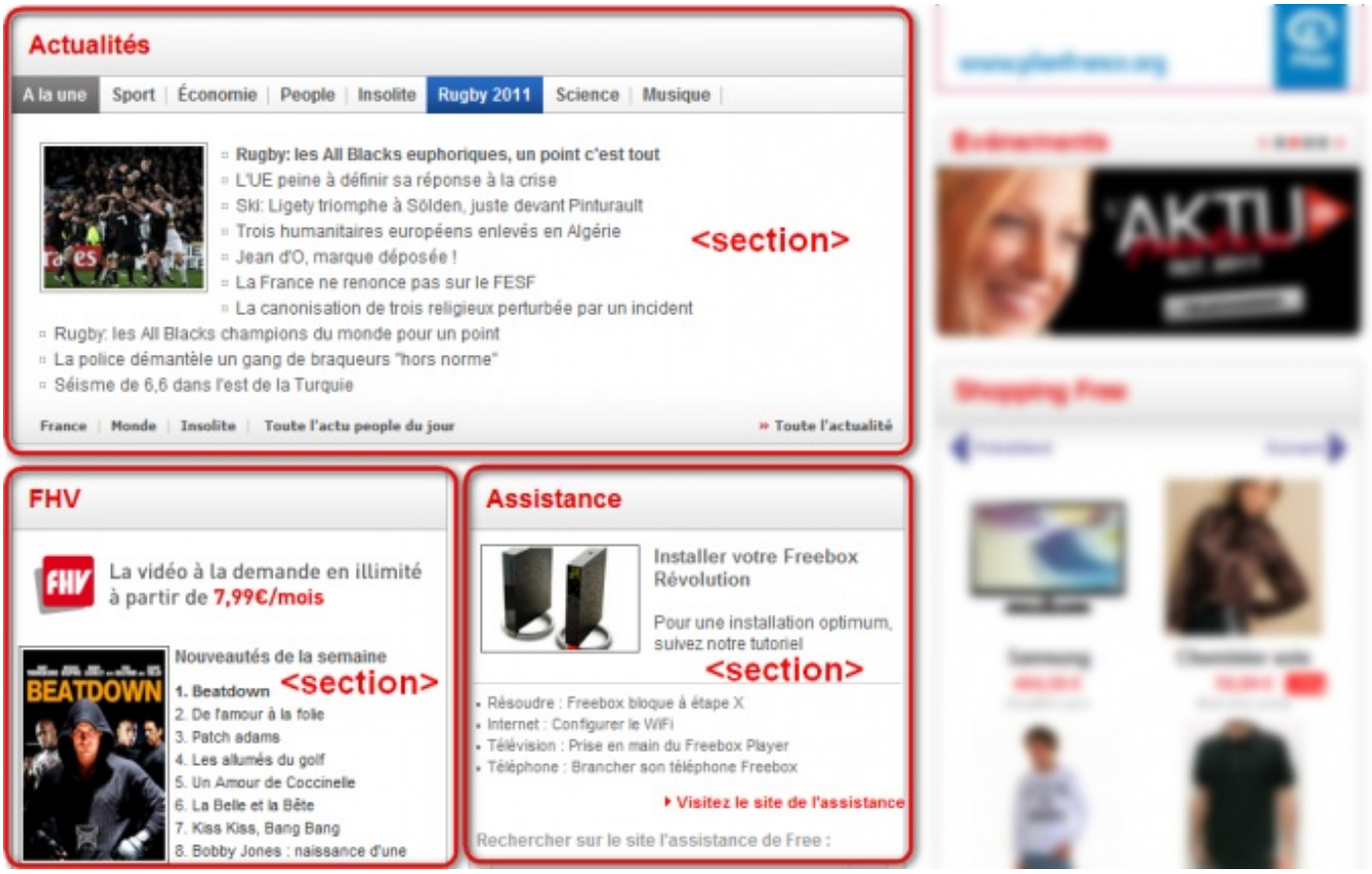
<section> : une section de page

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

Code : HTML

```
<section>
    <h1>Ma section de page</h1>
    <p>Bla bla bla bla</p>
</section>
```

Sur la page d'accueil du portail Free.fr, on trouve plusieurs blocs qui pourraient être considérés comme des sections de page :



Chaque section peut avoir son titre de niveau 1 (**<h1>**), de même que l'en-tête peut contenir un titre **<h1>** lui aussi. Chacun de ces blocs étant indépendant des autres, il n'est pas illogique de retrouver plusieurs titres **<h1>** dans le code de sa page web. On a ainsi "Le titre **<h1>** du **<header>**", "Le titre **<h1>** de cette **<section>**", etc.

<aside> : informations complémentaires

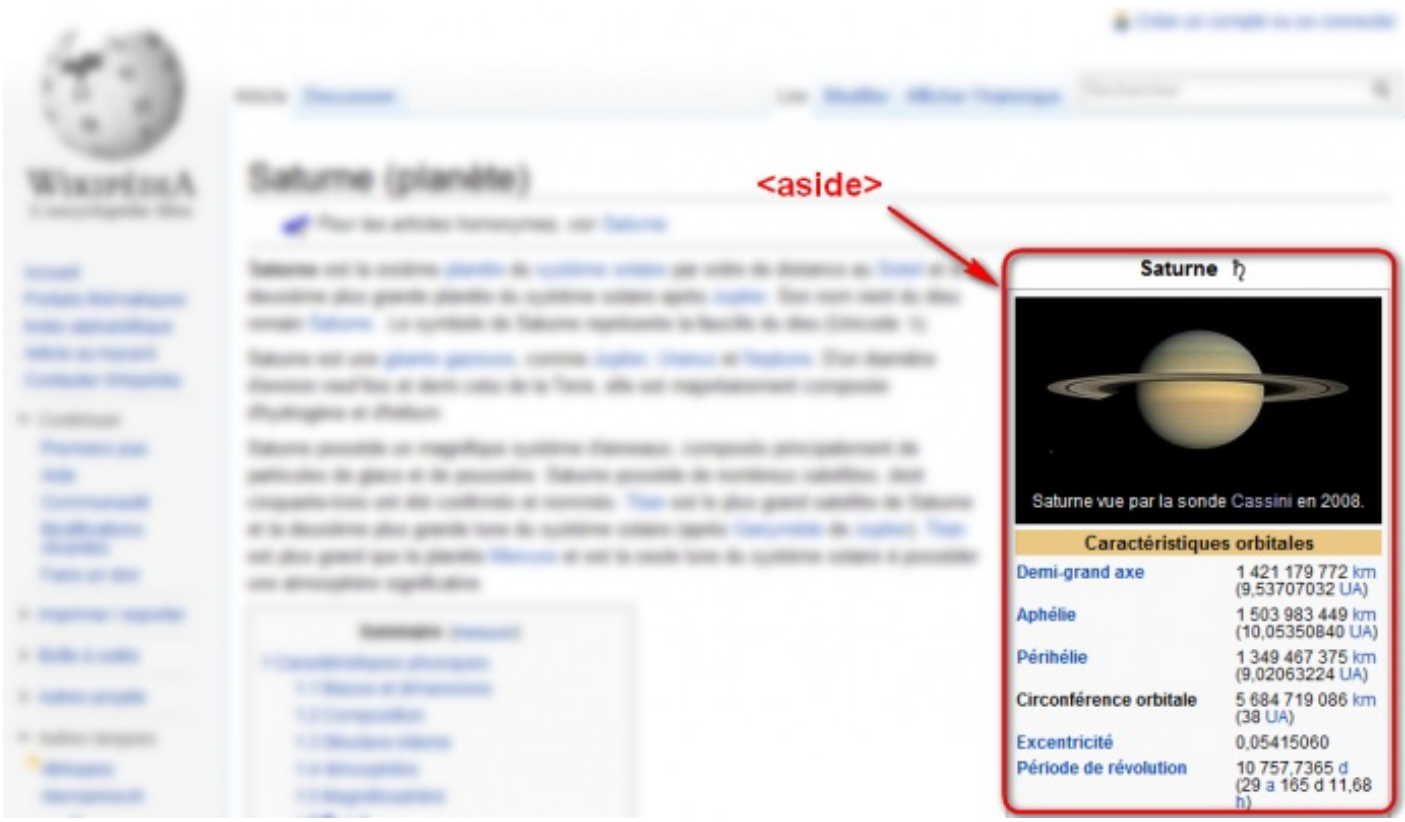
La balise **<aside>** est conçue pour contenir des informations complémentaires au document que l'on visualise. Ces informations sont généralement placées sur le côté (bien que ça ne soit pas une obligation).

Code : HTML

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

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

Sur Wikipédia par exemple, il est courant de voir à droite un bloc d'informations complémentaires sur l'article que l'on visualise. Par exemple, sur la page présentant la planète Saturne, on trouve ses caractéristiques (dimensions, masse...) dans ce bloc :



<article> : un article indépendant

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

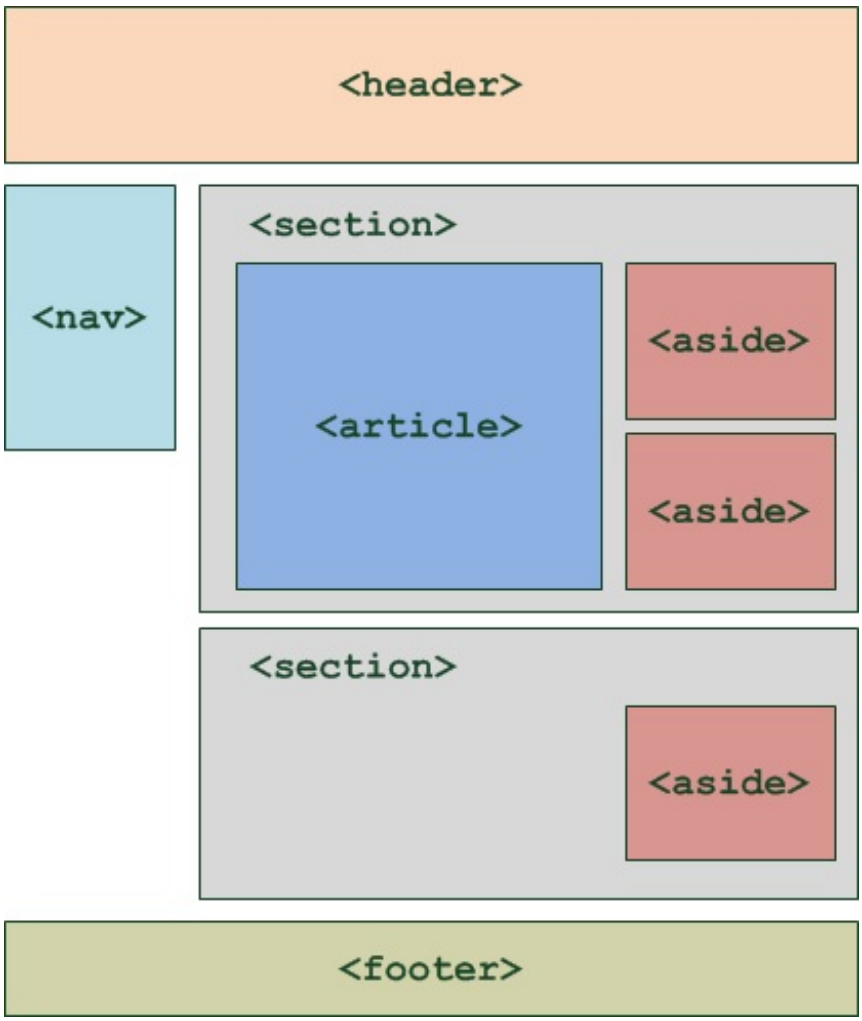
Code : HTML

```
<article>
    <h1>Mon article</h1>
    <p>Bla bla bla bla</p>
</article>
```

Par exemple, voici un article sur le journal Le Monde :

Résumé

Ouf, ça fait beaucoup de nouvelles balises à retenir.
Heureusement que je vous ai fait un petit schéma pour vous aider à retenir leur rôle !



Ne vous y laissez pas tromper : ce schéma propose un *exemple* d'organisation de la page. Rien ne vous empêche de décider que votre menu de navigation doit être à droite, ou tout en haut, que vos balises **<aside>** sont au-dessus, etc.
On peut même imaginer une seconde balise **<header>**, placée cette fois à l'intérieur d'une **<section>**. Dans ce cas-là, elle sera considérée comme étant l'en-tête de la section.
Enfin, une section ne doit pas forcément contenir un **<article>** et des **<aside>**. Utilisez ces balises uniquement si vous en avez besoin. Rien ne vous interdit de créer des sections contenant seulement des paragraphes par exemple.

Exemple concret d'utilisation des balises

Essayons d'utiliser les balises que nous venons de découvrir pour structurer notre page web. Le code ci-dessous reprend toutes les balises que nous venons de voir au sein d'une page web complète :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
```

```
                <li><a href="#">CV</a></li>
            </ul>
        </nav>

        <section>
            <aside>
                <h1>A propos de l'auteur</h1>
                <p>C'est moi, Zozor ! Je suis né un 23 novembre
2005.</p>
            </aside>
            <article>
                <h1>Je suis un grand voyageur</h1>
                <p>Bla bla bla bla (texte de l'article)</p>
            </article>
        </section>

        <footer>
            <p>Copyright Zozor - Tous droits réservés<br />
            <a href="#">Me contacter !</a></p>
        </footer>

    </body>
</html>
```

Ce code peut vous aider à comprendre comment sont supposées être agencées les balises. Vous y reconnaissez un en-tête, un menu de navigation, un pied de page... et une section au centre, avec un article et un bloc **<aside>** qui donne des informations sur l'auteur de l'article.

A quoi ressemble la page que nous venons de créer ?

A rien !

Si vous testez le résultat, vous verrez juste du texte noir sur fond blanc. C'est normal, il n'y a pas de CSS ! Par contre, la page est bien structurée ce qui va nous être utile pour la suite :

Essayer !

Les liens sont volontairement factices (d'où la présence d'un simple #), ils n'amènent donc nulle part (eh, c'est juste une page de démo) !

Je ne comprends pas l'intérêt de ces balises. On peut très bien obtenir le même résultat sans les utiliser !

C'est vrai. En fait, ces balises sont seulement là pour expliquer à l'ordinateur "Ceci est l'en-tête", "Ceci est mon pied de page"... Elles n'indiquent pas, contrairement à ce qu'on pourrait penser, où doit être placé le contenu. C'est le rôle du CSS, comme nous le verrons dans peu de temps maintenant.

A l'heure actuelle, ces balises ont encore assez peu d'utilité pour tout vous dire. On pourrait très bien utiliser des balises génériques `<div>` à la place pour englober les différentes portions de notre contenu. D'ailleurs, c'est comme cela qu'on faisait avant l'arrivée de ces nouvelles balises HTML5.

Néanmoins, il est assez probable que dans un futur proche les ordinateurs commencent à tirer parti intelligemment de ces nouvelles balises. On peut imaginer par exemple un navigateur qui choisit d'afficher les liens de navigation `<nav>` de manière toujours visible par exemple ! Quand l'ordinateur "comprend" la structure de la page, tout devient possible.

Assurer la compatibilité avec IE

Les nouvelles balises que nous venons de voir ne sont reconnues par Internet Explorer que depuis la version 9 (IE9). Cela va poser un problème car, quand les anciennes versions d'IE ne connaissent pas une balise... elles ne traitent pas correctement la page (impossible de modifier leur css par exemple) !

Cela peut heureusement se régler assez facilement à l'aide d'un script Javascript. Les scripts sont des petits morceaux de code qui permettent de manipuler la page web et d'effectuer certaines actions. Nous n'allons pas nous intéresser ici au Javascript (ce n'est pas le sujet de ce cours), mais il faut savoir qu'ils sont appelés depuis les pages HTML, un peu de la même façon que les fichiers

CSS.

Les fichiers Javascript sont généralement des fichiers ayant l'extension `.js`. Dans notre code HTML, on les place en général dans la balise **<head>** avec cette balise :

Code : HTML

```
<script src="monscript.js"></script>
```

Si je vous présente (brièvement) Javascript ici, c'est parce qu'un petit script répondant au doux nom de HTML5shiv permet de faire en sorte que les balises que nous venons de voir (**<header>**, **<footer>**, **<section>**...) s'affichent correctement sur les anciennes versions d'Internet Explorer (IE6, IE7, IE8). Concrètement, il vous suffit d'ajouter la ligne suivante dans votre code :

Code : HTML

```
<!--[if lt IE 9]>
<script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

Placez ce code dans la balise **<head>** comme ceci :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
<!--[if lt IE 9]>
<script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
        <title>Inclusion de HTML5shiv</title>
    </head>

    <body>

    </body>
</html>
```

Et voilà, votre site s'affichera désormais sans problème sous Internet Explorer !

Sans rentrer dans le détail, sachez que `<!--[if lt IE 9]>` est un *commentaire conditionnel*. C'est un commentaire spécial qui n'est lu que par Internet Explorer. Il permet de faire en sorte que le script ne s'exécute que sur les versions d'Internet Explorer inférieures à IE9 (les autres navigateurs n'en ont pas besoin et ignoreront le commentaire). Plus d'informations sur les commentaires conditionnels.

Notre page web commence à très sérieusement se structurer, il ne reste plus qu'à la mettre en forme. Nous allons commencer à faire cela dès le prochain chapitre !

Le modèle de boîte

Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle "blocs". La plupart des éléments vus au chapitre précédent sont des blocs : **<header>**, **<article>**, **<nav>**... Mais nous connaissions déjà d'autres blocs : les paragraphes ** **, les titres **<h1>**...

Dans ce chapitre, nous allons apprendre à manipuler ces blocs comme de véritables boîtes. Nous allons leur donner des dimensions, les agencer en jouant sur leurs marges, mais aussi apprendre à gérer leur contenu... pour éviter que le texte ne dépasse de ces blocs !

Ce sont des notions fondamentales dont nous allons avoir besoin pour mettre en page notre site web... Soyez attentifs !

Les balises de type block et inline

En HTML, la plupart des balises peuvent se ranger dans deux catégories :

- Les balises *inline* : c'est le cas par exemple des liens **<a>**.
- Les balises *block* : c'est le cas par exemple des paragraphes ** **.

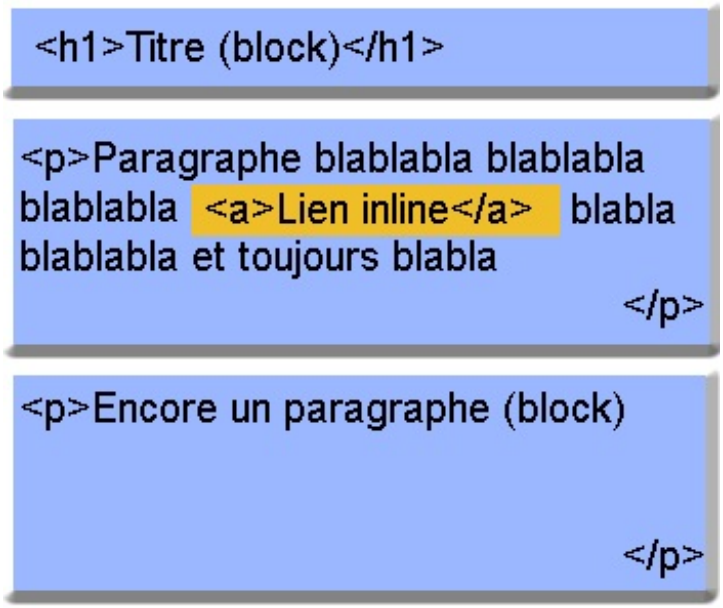
Il existe en fait plusieurs autres catégories très spécifiques, par exemples pour les cellules de tableau (type `table-cell`) ou les puces (type `list-item`). Nous n'allons pas nous y intéresser pour le moment, car ces balises sont minoritaires.

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 blocs à la suite les uns des autres. Mais vous verrez qu'en plus, il est possible de mettre un bloc 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").

Pour bien visualiser le concept, voici un petit schéma que je vous ai concocté :



- 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 blocs sont les uns en-dessous des autres. On peut aussi les imbriquer les uns à l'intérieur des autres (souvenez-vous, nos blocs **<section>** contiennent par exemple des blocs **<aside>** !).
La balise inline **<a>**, elle, se trouve à l'intérieur d'une balise block 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
<p>	
<footer>	
<h1>	<mark>
<h2>	<a>
<article>	<img />
...	...

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 des balises HTML.

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", **** "important", etc.).
Le principal intérêt de ces balises c'est de pouvoir leur 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 :

- **** (*inline*)
- ** ** (*block*)

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 ** ** et des **** trop souvent, et oublient que d'autres balises plus adaptées existent.

Voici 2 exemples :

- **Exemple d'un span inutile** : **<span** `class="important"`**>**. Je ne devrais jamais voir ceci dans un de vos codes alors qu'il existe la balise **** qui sert à indiquer l'importance !
- **Exemple d'un div inutile** : **<div** `class="titre"`**>**. Ceci est complètement absurde puisqu'il existe des balises faites spécialement pour les titres (**<h1>**, **<h2>**...)

Oui, vous allez me dire qu'au final le résultat (visuel) est le même. Je suis tout à fait d'accord. Mais les balises génériques n'apportent aucun sens à la page et ne peuvent pas être comprises par l'ordinateur. Utilisez toujours d'autres balises plus adaptées quand c'est possible. Google lui-même le conseille pour vous aider à améliorer la position de vos pages au sein de ses

résultats de recherche !

Les dimensions

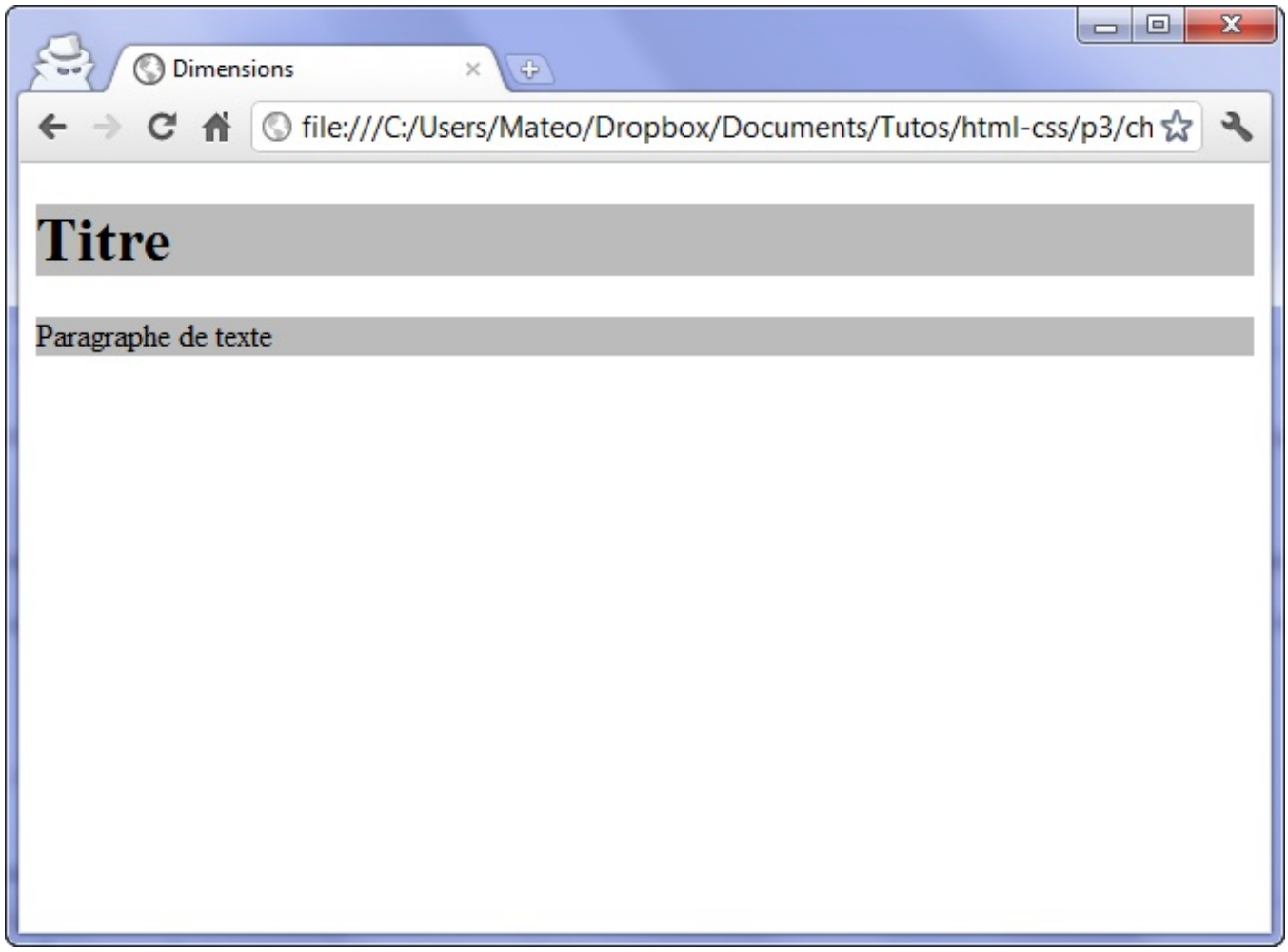
Nous allons ici travailler uniquement sur des balises de type "block".

Pour commencer, intéressons-nous à la taille des blocs. Contrairement à un inline, un bloc 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 bloc. 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 (%).

Pour être exact, **`width`** et **`height`** représentent la largeur et la hauteur du *contenu* des blocs. Si le bloc a des marges (on va découvrir ce principe un peu plus loin), celles-ci s'ajouteront à la largeur et la hauteur.

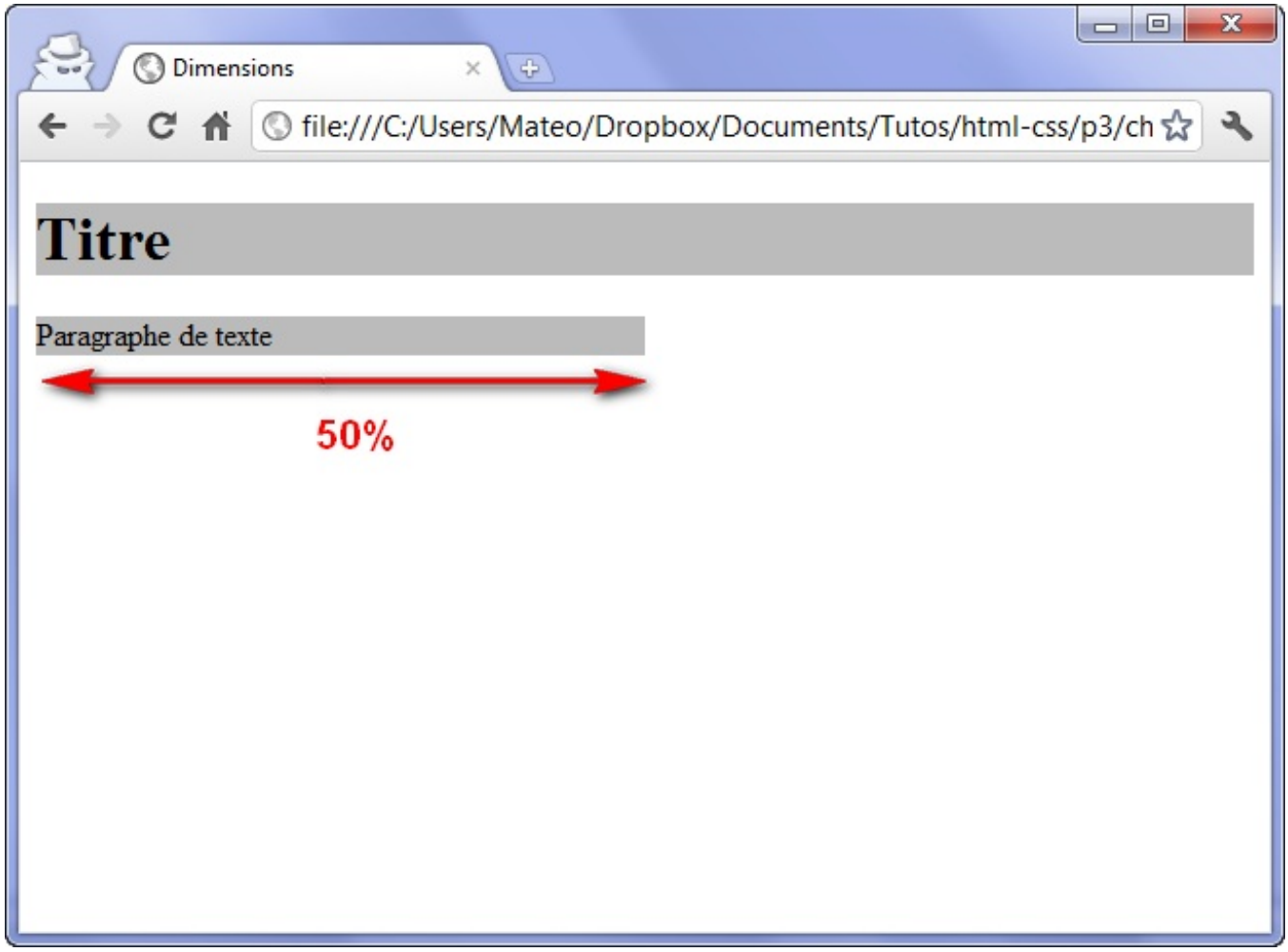
Par défaut, un bloc prend 100% de la largeur disponible. On peut le vérifier en ajoutant des bordures ou une couleur de fond sur nos blocs :



Maintenant, pimentons 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%;
}
```



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 blocs ayant une dimension précise en pixels :

Code : CSS

```
p
{
    width: 250px;
}
```

Minimum et maximum

On peut demander à ce qu'un bloc ait des dimensions minimales et maximales. C'est très pratique, car cela nous permet de définir des dimensions "limites" pour que notre site s'adapte aux différentes résolutions d'écran de nos visiteurs.

- **min-width** : largeur minimale
- **min-height** : hauteur minimale
- **max-width** : largeur maximale
- **max-height** : hauteur maximale

Par exemple, on peut demander à ce que les paragraphes occupent 50% de la largeur *et* exiger qu'il fassent au moins 400 pixels de large dans tous les cas :

Code : CSS

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

Essayez le résultat en modifiant la largeur de la fenêtre de votre navigateur. Vous allez voir que, si celle-ci est trop petite, le paragraphe se force à occuper au moins 400 pixels de largeur :

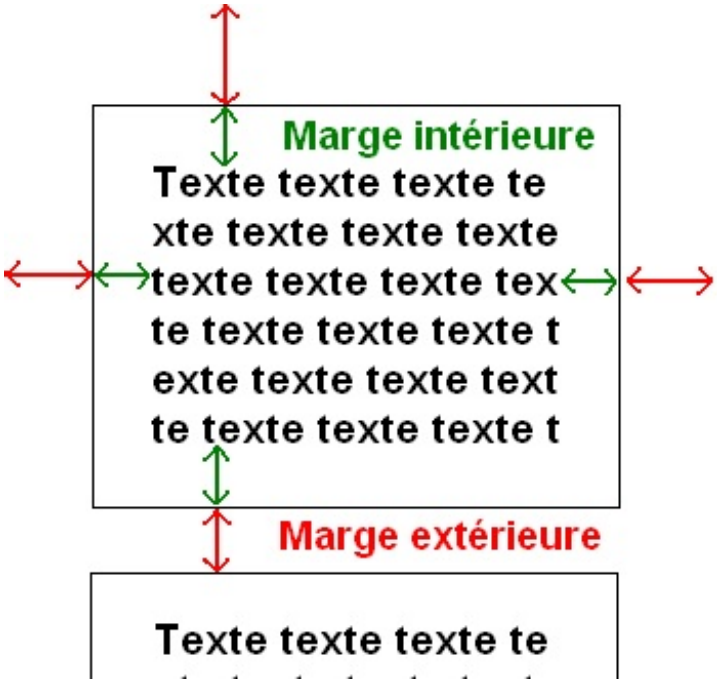
Essayer !

Les marges

Il faut savoir que tous les blocs possèdent des marges. Il existe **2 types de marges** :

- Les marges intérieures
- Les marges extérieures

Regardez bien ce schéma :



Sur ce bloc, 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 bloc 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.

Les balises de type inline possèdent aussi des marges. Vous pouvez donc aussi essayer ces manipulations sur ce type de balise.

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;
}
```

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 de type 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 :

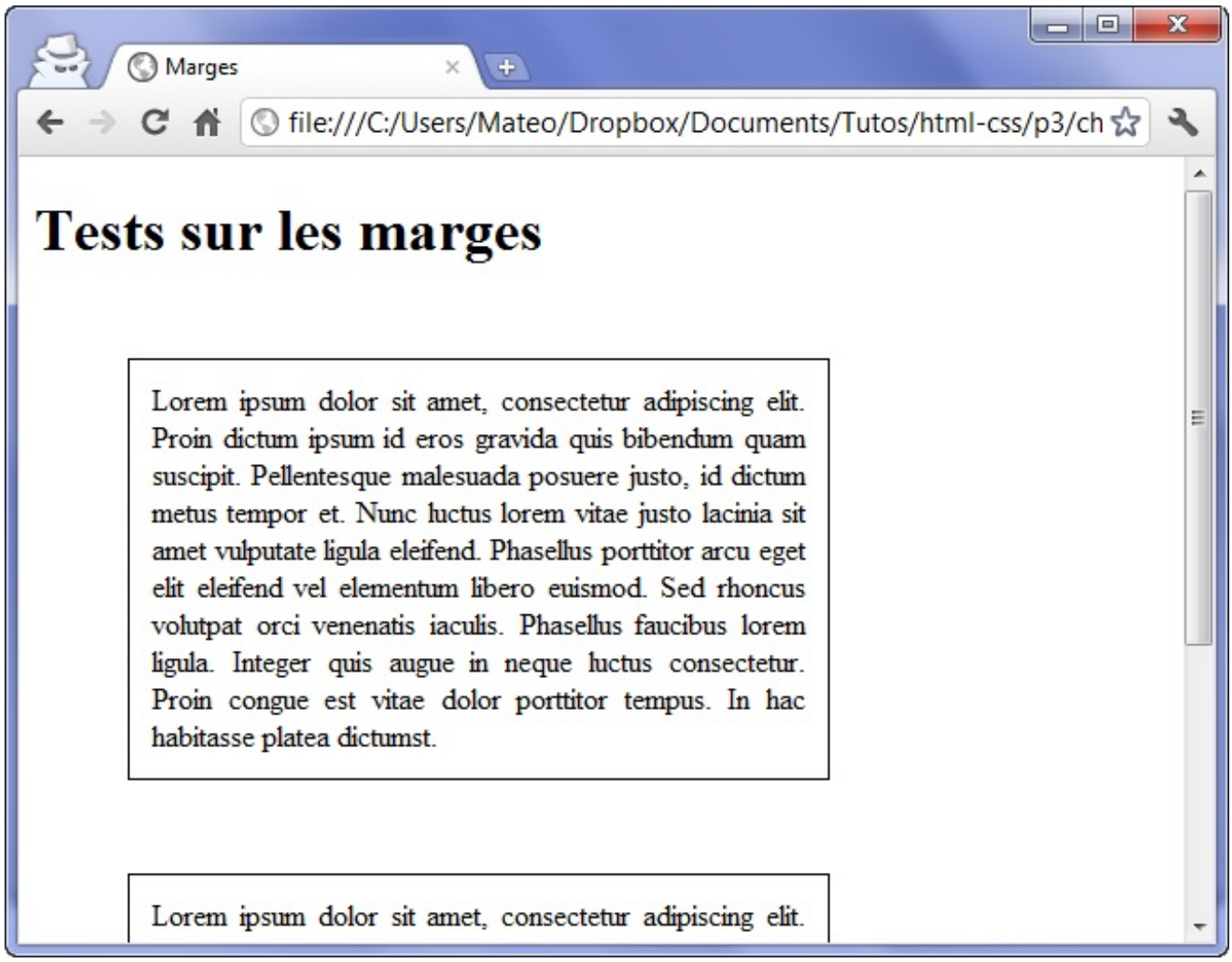
Code : CSS

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
padding: 12px; /* Marge intérieure de 12px */
}
```

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 rajoutée à gauche aussi !

Eh oui, **margin** (comme **padding** d'ailleurs) s'applique aux 4 côtés du bloc.
Si vous voulez indiquer une marge en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises... Le principe est le même que pour la propriété **border**, vous allez voir !

En haut, à droite, à gauche, en bas... Et on recommence !

L'idéal 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.

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

Centrer des blocs

Il est tout à fait possible de centrer des blocs. C'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.

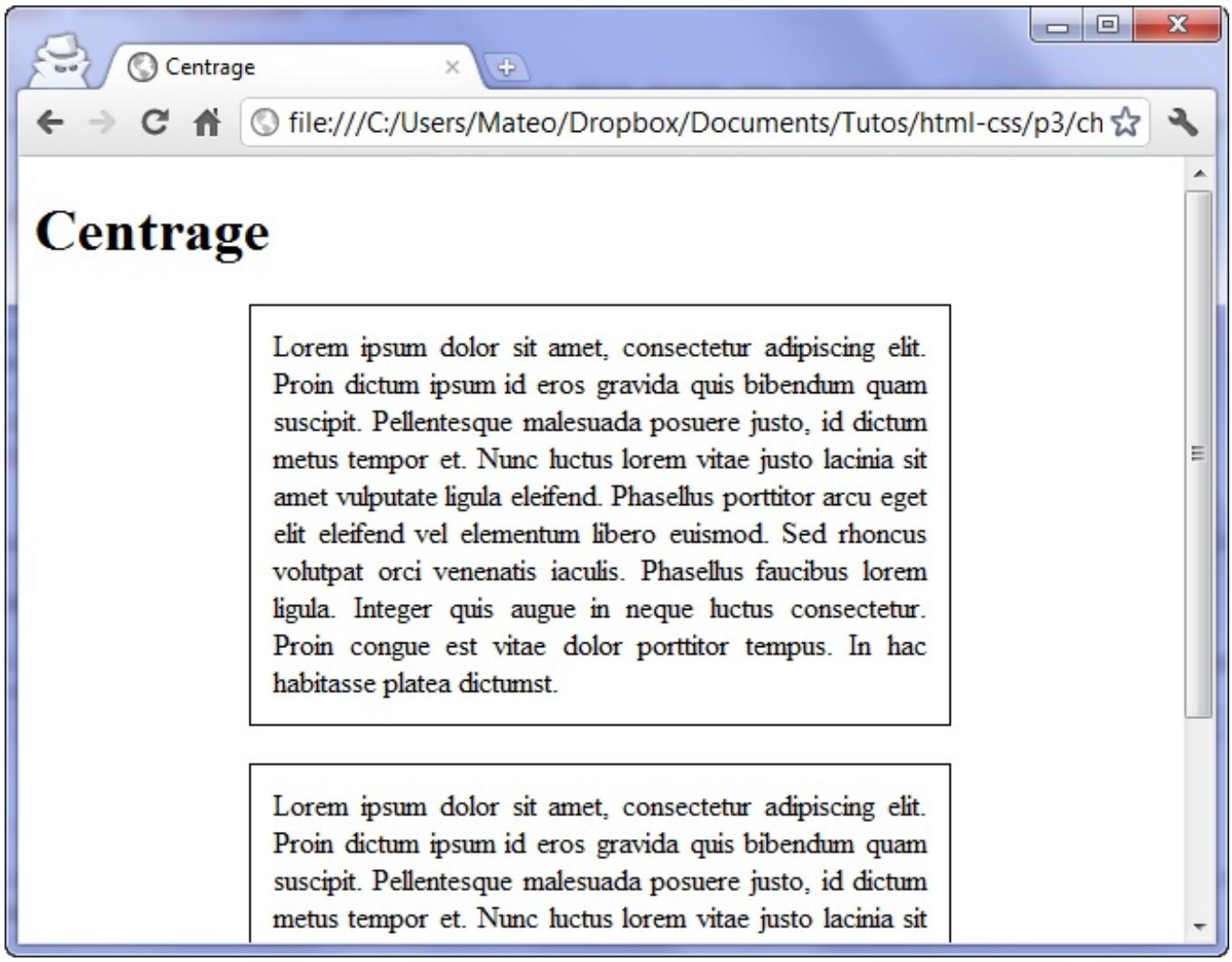
Pour centrer, il faut respecter les règles suivantes :

- Donnez une largeur au bloc (avec la propriété **width**)
- Indiquez que vous voulez des marges extérieures automatiques, comme ceci : **margin: auto;**

Essayons cette technique sur nos petits paragraphes :

Code : CSS

```
p
{
width: 350px; /* On a indiqué une largeur (obligatoire) */
margin: auto; /* On peut donc demander à ce que le bloc soit centré
avec "auto" */
    border: 1px solid black;
    text-align: justify;
    padding: 12px;
    margin-bottom: 20px;
}
```



Essayer !

Ainsi, le navigateur centre automatiquement nos paragraphes !

Il n'est cependant pas possible de centrer verticalement un bloc avec cette technique. Seul le centrage horizontal est permis.

Quand ça dépasse...

Lorsqu'on commence à définir des dimensions précises à nos blocs, comme on vient de le faire, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent.

Les propriétés CSS que nous allons voir ici ont justement été créées pour contrôler les dépassements... et décider quoi faire si jamais cela devait arriver.

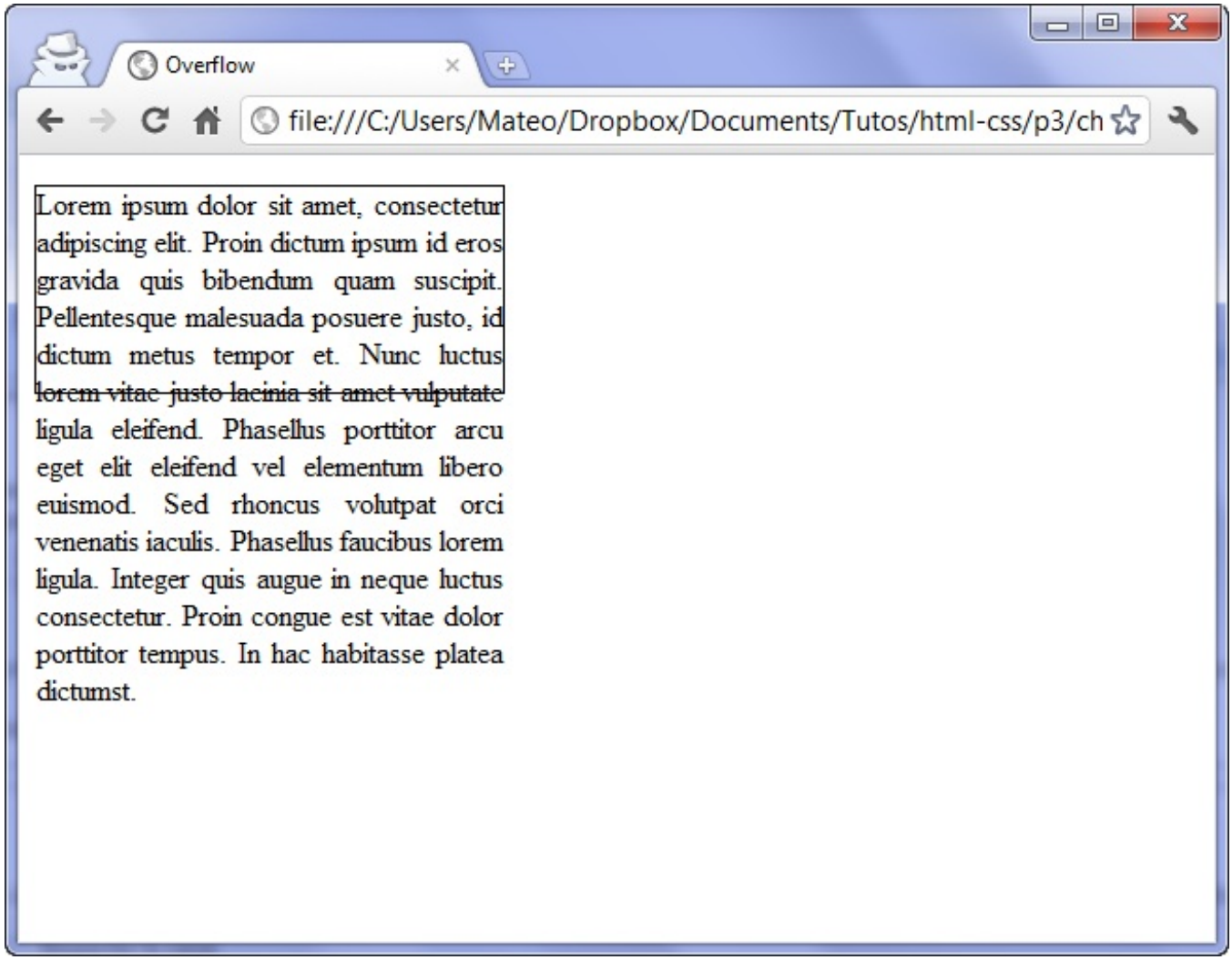
overflow : couper un bloc

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 110px de haut. Ajoutons-lui une bordure et remplissons-le de texte... à ras-bord :

Code : CSS

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
```

```
}
```



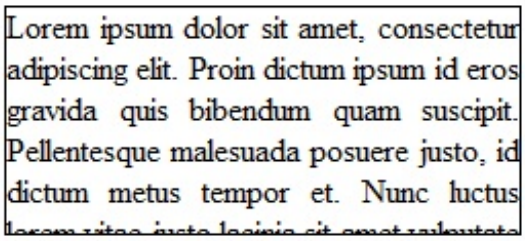
Horreur ! Le texte dépasse des limites du paragraphe !

Eh oui ! Vous avez demandé des dimensions précises, vous les avez eues ! Mais... le texte ne tient pas à l'intérieur d'un si petit bloc.

Si vous voulez que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété **`overflow`**. Voici les valeurs qu'elle peut accepter :

- `visible` (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
- `hidden` : si le texte dépasse les limites, il sera tout simplement coupé. 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` : c'est le mode "pilote automatique". En gros, c'est le navigateur qui décide ou pas de mettre des barres de défilement (il n'en mettra que si c'est nécessaire). C'est la valeur que je conseille d'utiliser le plus souvent.

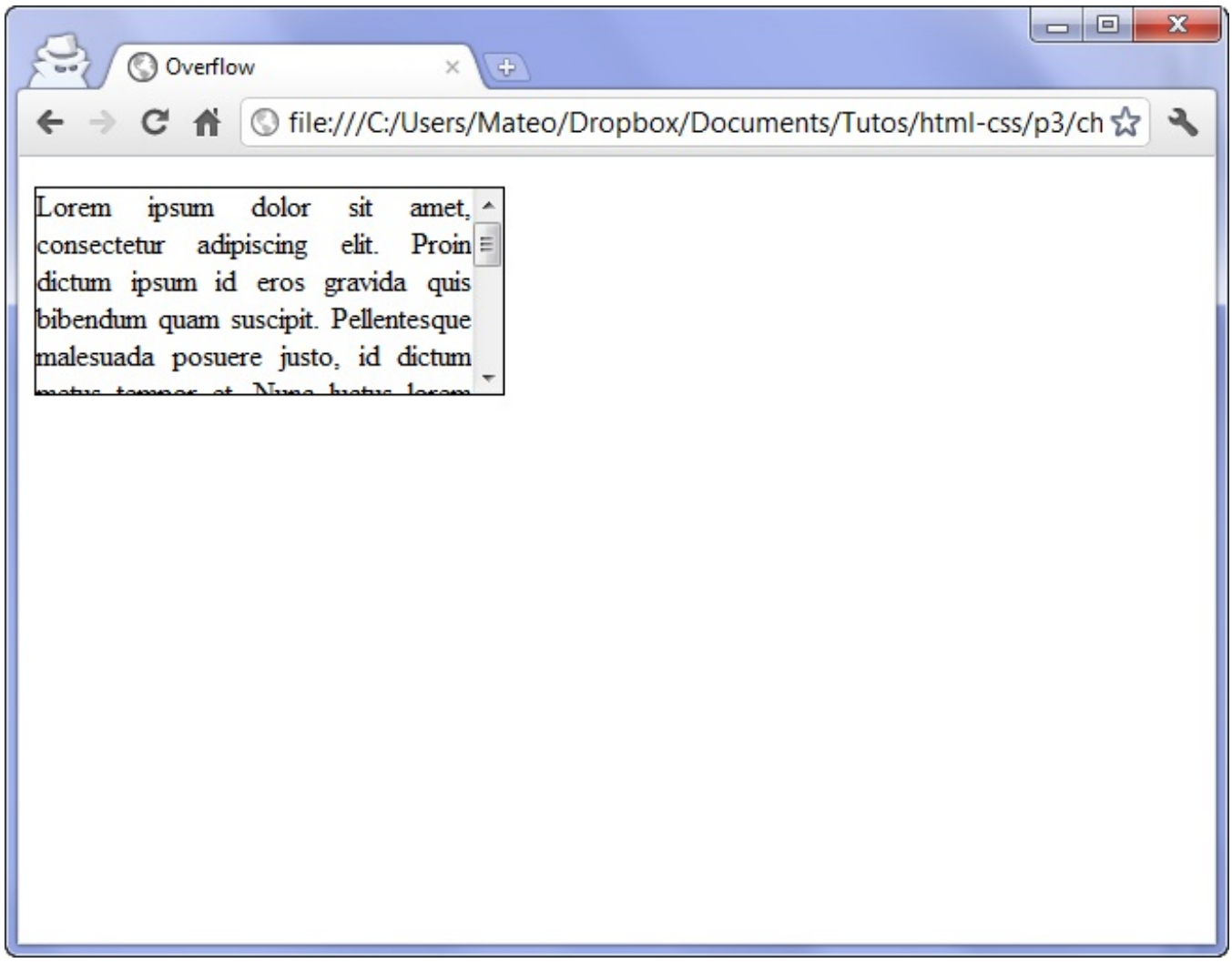
Avec **`overflow: hidden;`** le texte est donc coupé (on ne peut pas voir la suite) :



Essayons maintenant `overflow: auto;` avec le code CSS suivant :

Code : CSS

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
    overflow: auto;
}
```



Essayer !

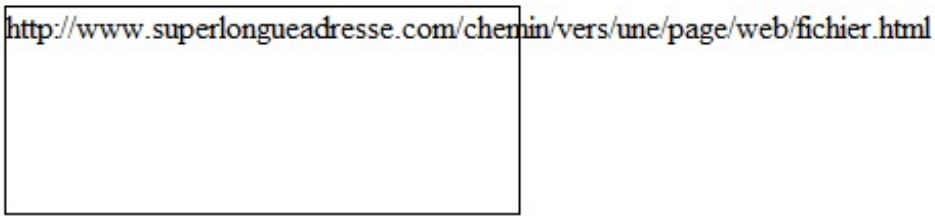
Eurêka ! Des barres de défilement nous permettent maintenant de consulter le contenu qui n'était pas visible.

Il existe une balise HTML, **<iframe>**, qui donne à peu près le même résultat. Elle permet de charger tout le contenu d'une autre page HTML au sein de votre page. Je vous conseille d'éviter d'en abuser, car il peut être lourd de charger une page qui charge elle-même d'autres pages web.

word-wrap : couper les textes trop larges

Si vous devez placer un mot très long dans un bloc, qui ne tient pas dans la largeur, vous allez adorer **word-wrap**. Cette propriété permet de forcer la césure des très longs mots (généralement des adresses un peu longues).

Voici par exemple un problème que l'on peut avoir quand on écrit une URL un peu longue dans un bloc :

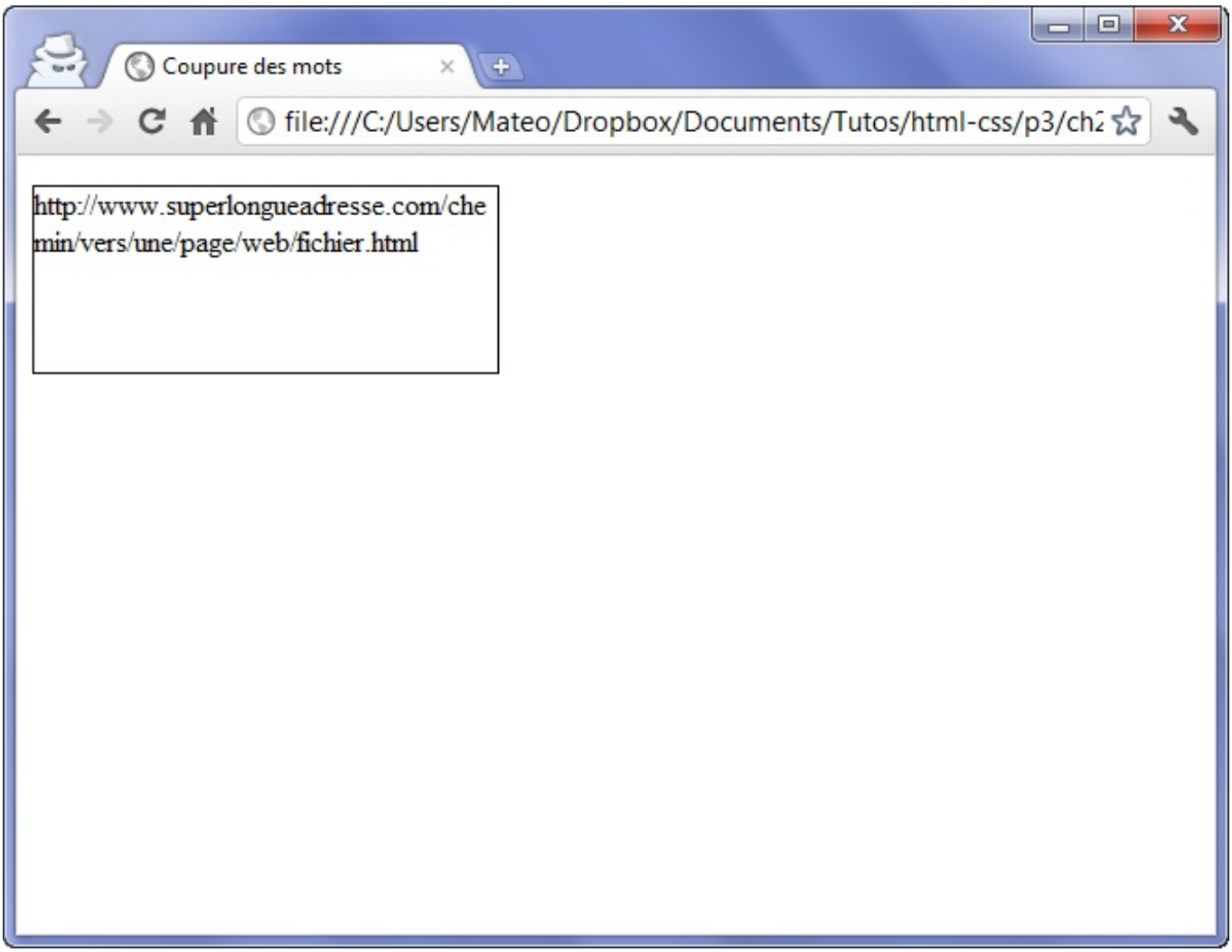


L'ordinateur ne sait pas "couper" l'adresse car il n'y a ni espace, ni tiret. Il ne sait pas faire la césure.

Avec le code suivant, la césure sera forcée dès que le texte risque de dépasser :

Code : CSS

```
p
{
    word-wrap: break-word;
}
```



Je conseille d'utiliser cette fonctionnalité dès qu'un bloc de texte est susceptible de contenir du texte saisi par des utilisateurs (par exemple sur les forums de votre futur site). Sans cette astuce, on peut "casser" facilement le design d'un site (en écrivant une longue suite de "aaaaaaaaaaa" par exemple).

Vous connaissez maintenant un peu mieux le fonctionnement des blocs qui constituent les pages web. Et si nous apprenions à les agencer entre eux pour construire notre site web ?

Vous allez voir, notre site web va très bientôt prendre forme !

Le positionnement en CSS

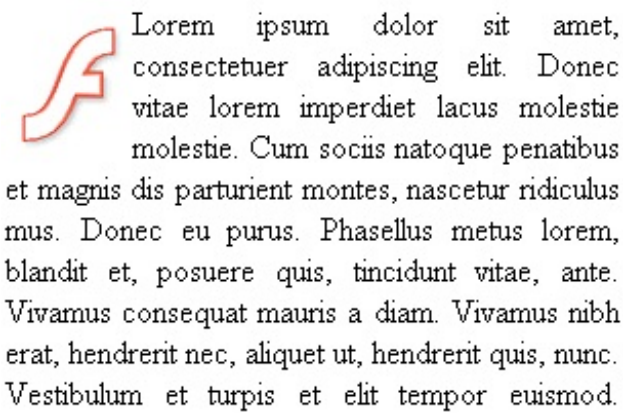
Voici venu le moment tant attendu : nous allons apprendre à modifier la position des éléments sur notre page. La théorie que nous allons voir ici nous sera indispensable dans le prochain chapitre, dans lequel nous réaliserons le design de notre premier site ensemble pas à pas !

Vous allez voir, il existe plusieurs techniques permettant d'effectuer la mise en page de son site. Chacune a ses avantages et ses défauts, ce sera à vous de sélectionner celle qui vous semble la meilleure selon votre cas.

Le positionnement flottant

La technique présentée ici l'est à titre d'information. Elle est utilisée par la majorité des sites à l'heure actuelle mais comporte un certain nombre de défauts. Une meilleure technique sera présentée un peu plus loin, le positionnement `inline-block`, que je vous encourage à utiliser autant que possible.

Vous vous souvenez de la propriété **`float`** ? Nous l'avons utilisée pour faire flotter une image autour du texte :



Il se trouve que cette propriété est aujourd'hui utilisée par la majorité des sites web pour... faire la mise en page ! En effet, si on veut placer son menu à gauche et le contenu de sa page à droite, c'est a priori un bon moyen. Je dis bien a priori, car cette propriété n'a pas été conçue pour faire la mise en page à la base et nous allons voir qu'elle a quelques petits défauts.

Reprenons le code HTML structuré que nous avons réalisé il y a quelques chapitres :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
                <li><a href="#">CV</a></li>
            </ul>
        </nav>

        <section>
            <aside>
                <h1>A propos de l'auteur</h1>
                <p>C'est moi, Zozor ! Je suis né un 23 novembre
```

```
2005.</p>
            </aside>
            <article>
                <h1>Je suis un grand voyageur</h1>
                <p>Bla bla bla bla (texte de l'article)</p>
            </article>
        </section>

        <footer>
            <p>Copyright Zozor - Tous droits réservés

            <a href="#">Me contacter !</a></p>
        </footer>

    </body>
</html>
```

Je rappelle que, sans CSS, la mise en page ressemble à ceci :

Nous allons essayer de placer le menu à gauche, et le reste du texte à droite. Pour cela, nous allons faire flotter le menu à gauche, et laisser le reste du texte se placer à sa droite.

Nous voulons que le menu occupe 150 pixels de large. Nous allons aussi rajouter une bordure noire au menu et une bordure bleue au corps (à la `<section>`) pour bien les distinguer :

Code : CSS

```
nav
{
```

```
        float: left;
        width: 150px;
        border: 1px solid black;
    }

    section
    {
        border: 1px solid blue;
    }
```

Voici le résultat :

Ca n'est pas encore tout à fait ça.

Il y a deux défauts (mis à part le fait que c'est encore bien moche) :

- Le texte du corps de la page touche la bordure du menu. Il manque une petite marge...
- Plus embêtant encore : la suite du texte passe... sous le menu !

On veut bien que le pied de page ("Copyright Zozor") soit placé en bas sous le menu, mais par contre on aimerait que tout le corps de page soit constitué comme un seul bloc à droite.

Pour résoudre ces deux problèmes d'un seul coup, il faut ajouter une marge extérieure à gauche de notre **<section>** *supérieure* à la largeur du menu. Si notre menu fait 150px, nous allons par exemple donner une marge extérieure gauche de 170px à notre section de page.

Code : CSS

```
nav
{
    float: left;
    width: 150px;
    border: 1px solid black;
}

section
{
margin-left: 170px;
    border: 1px solid blue;
}
```

Voilà, le contenu de la page est maintenant correctement aligné.

A l'inverse, il se peut que vous souhaitiez qu'un élément se place obligatoirement sous le menu. Dans ce cas, il faudra utiliser... `clear: both;`, que nous avions déjà découvert, qui oblige la suite du texte à se positionner sous l'élément flottant.

Transformez vos éléments avec display

Je vais vous apprendre ici ~~à repousser les lois de la physique~~ à modifier les lois du CSS (brrr...). Accrochez-vous !

Il existe en CSS une propriété très puissante : `display`. Elle est capable de **transformer** n'importe quel élément de votre page d'un type vers un autre. Avec cette propriété magique, je peux par exemple transformer mes liens (originellement de type inline) sous forme de blocs :

Code : CSS

```
a
{
    display: block;
}
```

A ce moment-là, les liens vont se positionner les uns en-dessous des autres (comme des blocs normaux) et il devient possible de modifier leurs dimensions !

Voici quelques-unes des principales valeurs que peut prendre la propriété `display` en CSS (il y en a encore d'autres) :

Valeur	Exemples	Description
`inline`	`<a>`, `<em>`, `<span>`...	Eléments d'une ligne. Se placent les uns à côté des autres.
`block`	`<p>`, `<div>`, `<section>`...	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
`inline-block`	`<select>`, `<input>`	Eléments positionnés les uns à côté des autres (comme les inline) mais qui peuvent être redimensionnés (comme les blocs).
`none`	`<head>`	Eléments non affichés.

On peut donc décider de masquer complètement un élément de la page avec cette propriété. Par exemple, si je veux masquer les éléments qui ont la classe "secret", je vais écrire :

Code : CSS

```
.secret
{
    display: none;
}
```

Et quel est ce nouveau type bizarre, `inline-block` ? C'est un mélange ? 

Oui, ce type d'élément est en fait une combinaison des inline et des blocks. C'est un peu le meilleur des deux mondes : ils s'affichent côte à côte et peuvent être redimensionnés.

Peu de balises sont affichées comme cela par défaut, c'est surtout le cas des éléments de formulaire (comme `<input>`) que nous découvrirons un peu plus tard. Par contre, avec la propriété `display`, nous allons pouvoir transformer d'autres balises en `inline-block`, ce qui va nous aider pour réaliser notre design. 

Le positionnement inline-block

Les manipulations que demande le positionnement flottant se révèlent parfois un peu délicates sur des sites complexes. Dès qu'il y a un peu plus qu'un simple menu à mettre en page, on risque d'avoir à recourir à des `clear: both;` qui complexifient rapidement le code de la page.

Si le positionnement flottant reste, de loin, le mode de positionnement le plus utilisé sur le Web à l'heure actuelle, d'autres techniques existent et bien peu de webmasters le savent. L'une d'elles, étonnamment puissante, est passée sous les yeux des concepteurs de sites web alors qu'elle existe depuis CSS 2.1, c'est-à-dire depuis plus de 10 ans ! Elle consiste à transformer vos éléments en `inline-block` avec la propriété `display`.

Quelques petits rappels sur les éléments de type inline-block :

- Ils se positionnent les uns à côté des autres (exactement ce qu'on veut pour placer notre menu et le corps de notre page !).
- On peut leur donner des dimensions précises (là encore, exactement ce qu'on veut !).

Nous allons transformer en inline-block les deux éléments que nous voulons placer côte à côte : le menu de navigation et la section du centre de la page.

Code : CSS

```
nav
{
    display: inline-block;
    width: 150px;
    border: 1px solid black;
}

section
{
    display: inline-block;
    border: 1px solid blue;
}
```

Argh !
Ca n'est pas tout à fait ce qu'on voulait là encore. Et en fait, c'est normal : les éléments inline-block se positionnent sur une même ligne de base (appelée *baseline*), en bas.

Heureusement, le fait d'avoir transformé les éléments en inline-block nous permet d'utiliser une nouvelle propriété, normalement réservée aux tableaux : **vertical-align**. Cette propriété permet de modifier l'alignement vertical des éléments. Voici quelques-unes des valeurs possibles pour cette propriété :

- `baseline` : aligne la base de l'élément avec celle de l'élément parent (par défaut)
- `top` : aligne en haut
- `middle` : aligne au milieu
- `bottom` : aligne en bas
- (valeur en px ou %) : aligne à une certaine distance de la ligne de base (baseline)

Il ne nous reste plus qu'à aligner nos éléments en haut, et le tour est joué !

Code : CSS

```
nav
{
    display: inline-block;
    width: 150px;
    border: 1px solid black;
vertical-align: top;
}

section
{
    display: inline-block;
    border: 1px solid blue;
vertical-align: top;
}
```

Vous noterez que le corps (la **<section>**) ne prend pas toute la largeur. En effet, ce n'est plus un bloc ! La section occupe seulement la place dont elle a besoin.
Si cela ne vous convient pas pour votre design, modifiez la taille de la section avec **width**.

Et voilà ! Pas besoin de s'embêter avec les marges, aucun risque que le texte passe sous le menu... Bref, c'est parfait !

... Quoi ? Pardon, on me signale dans l'oreillette qu'un certain navigateur vient jouer les trouble-fêtes...

inline-block et compatibilité Internet Explorer

Le positionnement `inline-block` est parfaitement compris par Internet Explorer à partir de IE8.

Pour les anciennes versions, IE6 et IE7 en particulier, le positionnement inline-block fonctionne... mais uniquement sur les éléments qui étaient des inline à la base !

On peut heureusement régler ce problème avec une petite "bidouille" : si le navigateur est IE6 ou IE7, on transforme l'élément en inline et on change son comportement en lui donnant le *layout* (en lui affectant la propriété **zoom : 1 ;**). L'élément pourra alors être redimensionné.

Le *layout* est un mode d'affichage un peu particulier d'Internet Explorer, qui peut être la source de nombreux bugs. Comme ça devient rapidement très technique, je passe volontairement là-dessus et je vous invite à consulter cet article sur HasLayout sur le site Alsacréations si le sujet vous intéresse.

Pour commencer, il va falloir créer une seconde feuille de style CSS spéciale pour Internet Explorer 6 et 7. On va utiliser pour cela un *commentaire conditionnel*, que nous avions déjà aperçu auparavant, qui ne sera lu que par IE6 et IE7 :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
<!--[if lte IE 7]>
<link rel="stylesheet" href="style_ie.css" />
<![endif]-->

        <title>Zozor - Le Site Web</title>
    </head>
```

Si le navigateur est IE6 ou IE7, une seconde feuille de style CSS sera chargée et viendra s'ajouter à la première : `style_ie.css`. Dedans, nous ferons toutes nos "bidouilles" pour régler les problèmes des vieilles versions d'Internet Explorer, afin que notre site s'affiche correctement chez tout le monde, même chez ceux qui vivent encore à la préhistoire (pardon).

Dans `style_ie.css`, rajoutez simplement le code suivant :

Code : CSS

```
nav, section
{
    display: inline;
    zoom: 1;
}
```

Voilà, c'est tout ! Dans votre feuille de style classique (`style.css`), continuez à utiliser `inline-block` comme avant. Ceux qui ont IE6 ou IE7 liront *en plus* ce code un peu spécial, qui leur permet d'obtenir le même résultat visuel ! 

Les positionnements absolu, fixe et relatif

Il existe d'autres techniques un peu particulières permettant de positionner avec précision des éléments sur la page :

- **Le positionnement absolu** : il nous permet de placer un élément n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre, etc..)
- **Le positionnement fixe** : identique au positionnement absolu, mais cette fois l'élément reste toujours visible, même si on descend plus bas dans la page. C'est un peu comme le même principe que `background-attachment: fixed;` (si vous vous en souvenez encore ).
- **Le positionnement relatif** : permet de décaler l'élément par rapport à sa position normale.

Comme pour les flottants, le positionnement absolu, fixé et relatif fonctionne aussi sur des balises de type inline. 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

Le positionnement absolu permet de placer un élément (réellement) n'importe où sur la page. Pour effectuer un positionnement absolu, on doit écrire :

Code : CSS

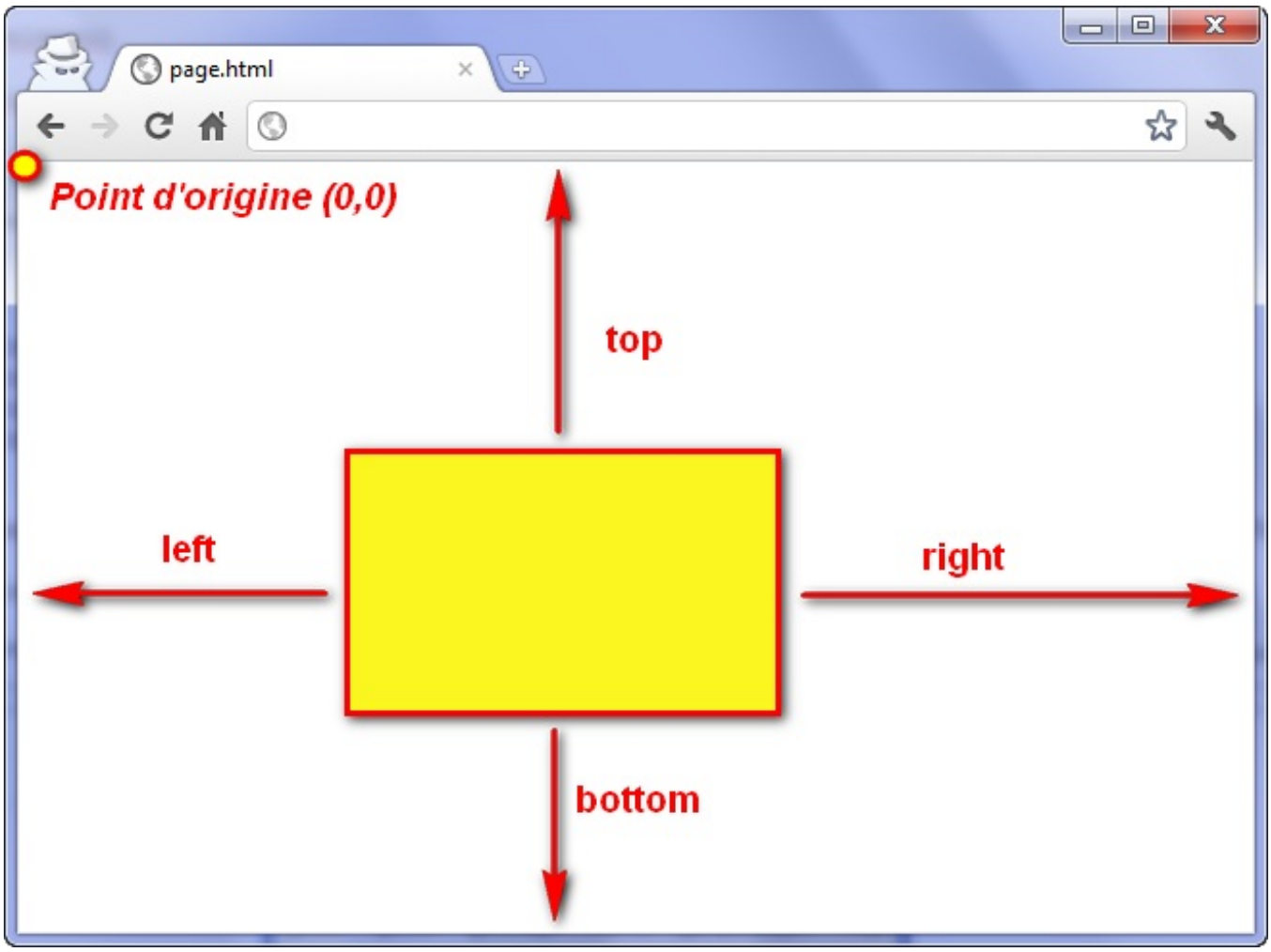
```
element
{
    position: absolute;
}
```

Mais ça ne suffit pas ! On a dit qu'on voulait un positionnement absolu, mais encore faut-il dire *où on veut que le bloc soit positionné sur la page*.
Pour ce faire, on va utiliser 4 propriétés CSS :

- `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 bloc.

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 :

Code : CSS

```
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

... cela signifie que le bloc 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).

Si on essaye de placer notre bloc **<nav>** en bas à droite de la page, on obtient ce résultat :

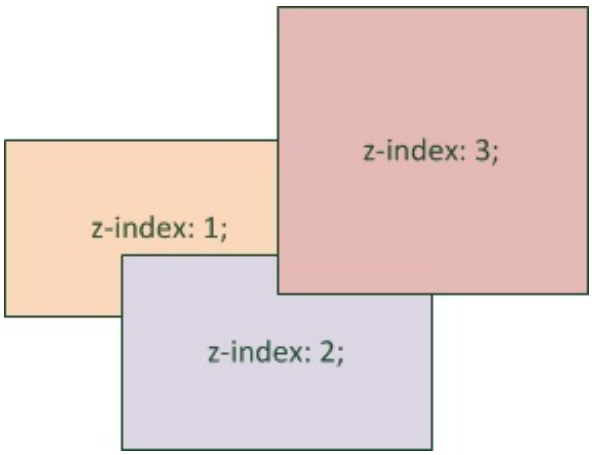
On peut bien entendu ajouter une marge intérieure (padding) au menu pour qu'il soit moins collé à sa bordure.

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

Code : CSS

```
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
z-index: 1;
}
element2
{
    position: absolute;
    right: 30px;
    bottom: 30px;
z-index: 2;
}
```

L'élément ayant la valeur de z-index la plus élevée sera placé par-dessus les autres, comme le montre ce schéma :



Une petite précision technique qui a son importance : le positionnement absolu ne se fait pas toujours forcément par rapport au coin en haut à gauche de la fenêtre ! Si vous positionnez en absolu un bloc A qui se trouve dans un autre bloc B lui-même positionné en absolu (ou fixe ou relatif), alors votre bloc A se positionnera par rapport au coin en haut à gauche du bloc B. Faites le test, vous verrez !

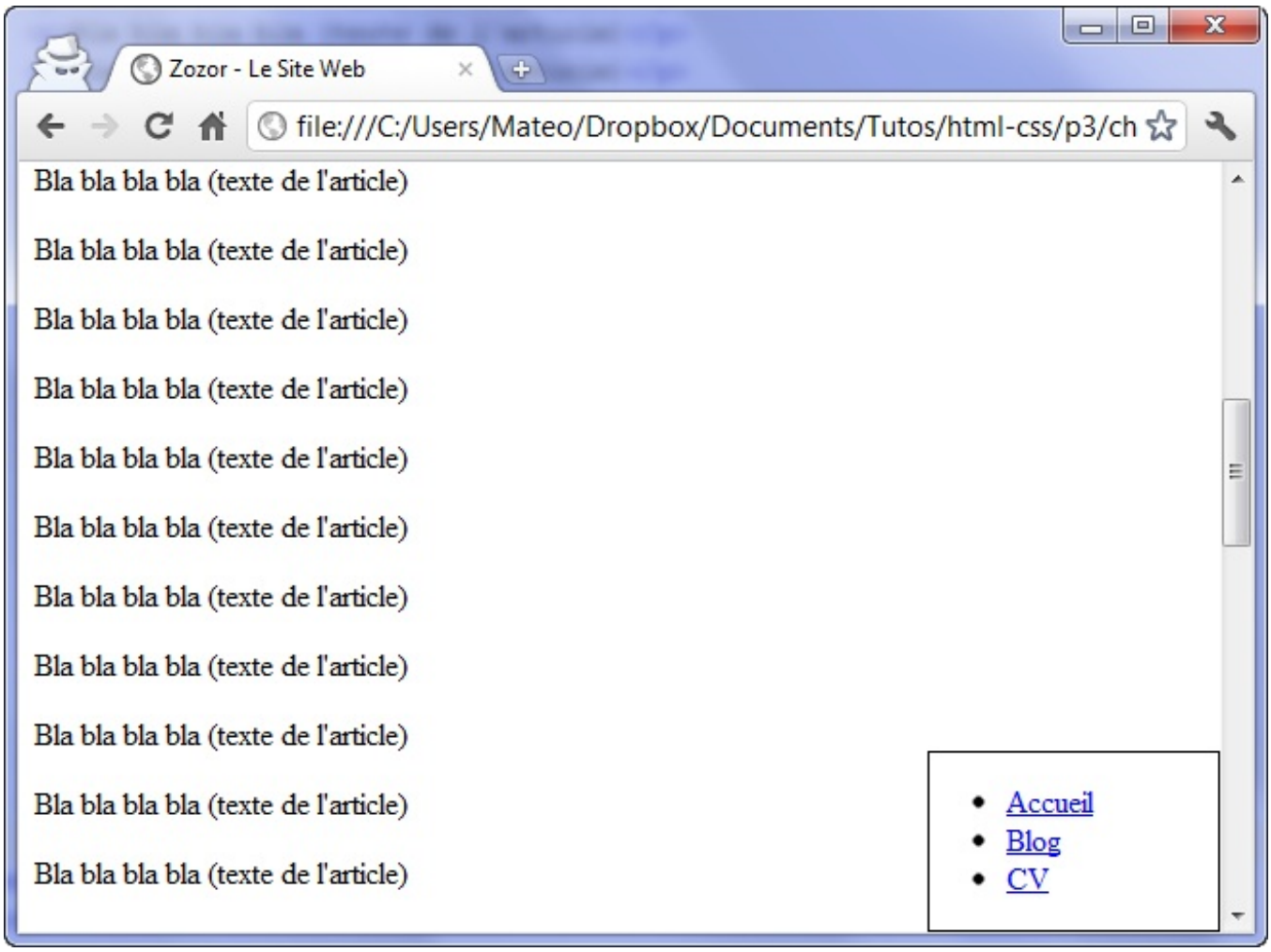
Le positionnement fixe

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

Code : CSS

```
element
{
    position: fixed;
    right: 0px;
    bottom: 0px;
}
```

Essayez le résultat, vous verrez que le menu reste dans le cas présent affiché en bas à droite même si on descend plus bas dans la page.



Essayez !

Le positionnement relatif

Le positionnement relatif est un petit peu plus délicat à utiliser. Ce positionnement permet d'effectuer des "ajustements" : l'élément est décalé par rapport à sa position initiale.

Prenons par exemple un texte important, situé entre 2 balises ****. 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 */
}
```

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 ? C'est le principe de la position relative. Ce schéma devrait vous aider à comprendre où se trouve l'origine des points :

Donc, si vous faites un **position: relative;** et que vous appliquez une des propriétés **top**, **left**, **right** ou **bottom**, le texte sur fond rouge va se déplacer par rapport à la position où il se trouve.

Prenons 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;
}
```

Le texte s'est alors décalé par rapport à sa position initiale, comme ceci :

Avec ces connaissances, vous êtes maintenant 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 va être l'occasion de pratiquer : je vais vous montrer comment on crée le design d'un site de A à Z avec tout ce qu'on a appris !

Eh oui, vous savez maintenant tout ce qu'il faut, encore faut-il maintenant savoir s'y prendre... On y va quand vous voulez !

TP : création d'un site pas à pas

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.

Vous avez lu beaucoup de théorie jusqu'ici, mais vous vous demandez sûrement comment font les webmasters pour créer d'aussi beaux sites. Vous vous dites que vous êtes encore loin d'avoir les connaissances nécessaires pour construire tout un site... eh bien vous vous trompez !

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.

Maquettage du design

Je vois le tableau d'ici. Vous vous dites "*Chouette, on va créer un site complet*", vous ouvrez votre éditeur de texte (Notepad++ par exemple), et vous me regardez en me disant "*Bon, par quelle ligne de code on commence ?* ".

Et là, je dois justement vous arrêter. Prenez un crayon et un papier et arrêtez votre ordinateur (bon il faut quand même que vous puissiez continuer à me lire quand même !).

Il faut d'abord réfléchir à ce que vous voulez créer comme site. De quoi va-t-il parler ? Avez-vous un thème, un objectif ?
Je sais, par expérience, que la plupart d'entre vous "cherchent juste à apprendre" pour le moment. Vous n'avez donc peut-être pas encore d'idée précise en tête. Dans ce cas, je vous suggère de créer un site pour vous présenter, pour assurer votre présence sur le Web : ce site parlera de vous, il y aura votre CV, vos futures réalisations et pourquoi pas votre blog.

Zozor !

En ce qui me concerne, dans ce TP je vais réaliser le site web de notre mascotte Zozor, le célèbre âne du Site du Zéro. Zozor a décidé de partir en voyage à travers le monde, et sa première étape sera... San Francisco ! Il veut donc créer un site web pour qu'on le connaisse et pour qu'on suive son périple à travers le monde.

La première étape consiste à **maquetter le design**, pour avoir un objectif du site web à réaliser. A partir de là, deux possibilités :

- Soit vous êtes un graphiste (ou vous en connaissez un) qui a l'habitude d'imaginer des designs, avec des logiciels comme Photoshop
- Soit vous n'êtes pas très créatif, vous manquez d'inspiration, et dans ce cas vous allez chercher votre inspiration sur des sites web comme freehtml5templates.com, qui vous proposent des idées de design et qui peuvent même vous donner le code HTML / CSS tout prêt !

Pour ma part, j'ai fait appel à un graphiste, qui m'a proposé ce design (qui me plaît beaucoup !) :

Conception de la maquette : Fan Jiyong

Cette maquette est en fait une simple image du résultat qu'on veut obtenir. Je demande au graphiste de me fournir les éléments qui vont m'aider à construire le design, c'est-à-dire les codes couleur utilisés, les images découpées et les polices dont j'aurai besoin.

Télécharger les images et les polices

Quelques-unes des images "découpées" utilisées dans le design

Il ne nous reste plus qu'à réaliser ce site web ! Nous allons procéder en deux temps :

1. Nous allons construire le squelette **HTML** de la page
2. Puis nous allons le mettre en forme et le mettre en page avec **CSS**

Allez, au boulot ! 

Organiser le contenu en HTML

La première chose à faire est de distinguer les principaux blocs sur la maquette. Ces blocs vont constituer le squelette de notre page.

Pour créer ce squelette, nous allons utiliser différentes balises HTML :

- Les balises structurantes de HTML5, que nous connaissons : `<header>`, `<section>`, `<nav>`...
- La balise universelle `<div>` quand aucune balise structurante ne convient.

 Comment je sais quelle balise utiliser moi ? 

C'est à vous de décider. De préférence, utilisez une balise qui a du sens (comme les balises structurantes `<header>`, `<section>`, `<nav>`), mais si aucune balise ne vous semble mieux convenir, optez pour la balise générique `<div>`.

Voici comment je propose de faire le découpage :

On peut imaginer d'autres façons de faire le découpage, retenez bien que ma proposition n'est pas forcément la seule et unique solution !

Toutes les balises que l'on va utiliser n'apparaissent pas sur cette maquette, mais cela vous permet d'avoir un ordre d'idée de l'imbrication des éléments que je propose.

Le HTML n'est pas vraiment la partie complexe de la réalisation du site web. En fait, si vous avez bien compris comment imbriquer des balises, vous ne devriez pas avoir de mal à réaliser un code approchant du mien :

Code : HTML

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

```
        <title>Zozor - Carnets de voyage</title>
    </head>

    <body>
        <div id="bloc_page">
            <header>
                <div id="titre_principal">
                    <img src="images/zozor_logo.png" alt="Logo de
Zozor" id="logo" />
                    <h1>Zozor</h1>
                    <h2>Carnets de voyage</h2>
                </div>

                <nav>
                    <ul>
                        <li><a href="#">Accueil</a></li>
                        <li><a href="#">Blog</a></li>
                        <li><a href="#">CV</a></li>
                        <li><a href="#">Contact</a></li>
                    </ul>
                </nav>
            </header>

            <div id="banniere_image">
                <div id="banniere_description">
                    Retour sur mes vacances aux États-Unis...
                    <a href="#" class="bouton_rouge">Voir l'article
<img src="images/flecheblanchedroite.png" alt="" /></a>
                </div>
            </div>

            <section>
                <article>
                    <h1><img src="images/ico_epingle.png"
alt="Catégorie voyage" class="ico_categorie" />Je suis un grand
voyageur</h1>
                    <p>Lorem ipsum dolor sit amet...</p>
                    <p>Vivamus sed libero nec mauris pulvinar
facilisis ut non sem...</p>
                    <p>Phasellus ligula massa, congue ac vulputate
non, dignissim at augue...</p>
                </article>
                <aside>
                    <h1>À propos de l'auteur</h1>
                    <img src="images/bulle.png" alt=""
id="fleche_bulle" />
                    <p id="photo_zozor"><img
src="images/zozor_classe.png" alt="Photo de Zozor" /></p>
                    <p>Laisse-moi le temps de me présenter : je
m'appelle Zozor, je suis né un 23 novembre 2005.</p>
                    <p>Bien maigre, n'est-ce pas ? C'est pourquoi,
aujourd'hui, j'ai décidé d'écrire ma biographie (ou zBiographie,
comme vous voulez !) afin que les zéros sachent qui je suis
réellement.</p>
                    <p><img src="images/facebook.png" alt="Facebook"
/><img src="images/twitter.png" alt="Twitter" /><img
src="images/vimeo.png" alt="Vimeo" /><img src="images/flickr.png"
alt="Flickr" /><img src="images/rss.png" alt="RSS" /></p>
                </aside>
            </section>

            <footer>
                <div id="tweet">
                    <h1>Mon dernier tweet</h1>
                    <p>Hii haaaaaan !</p>
                    <p>le 12 mai à 23h12</p>
                </div>
                <div id="mes_photos">
                    <h1>Mes photos</h1>
                    <p><img src="images/photo1.jpg"
```

```
        alt="Photographie" /><img src="images/photo2.jpg" alt="Photographie"
        /><img src="images/photo3.jpg" alt="Photographie" /><img
        src="images/photo4.jpg" alt="Photographie" /></p>
                    </div>
                    <div id="mes_amis">
                        <h1>Mes amis</h1>
                        <ul>
                            <li><a href="#">Pupi le lapin</a></li>
                            <li><a href="#">Mr Baobab</a></li>
                            <li><a href="#">Kaiwaii</a></li>
                            <li><a href="#">Perceval.eu</a></li>
                        </ul>
                        <ul>
                            <li><a href="#">Belette</a></li>
                            <li><a href="#">Le concombre masqué</a></li>
                            <li><a href="#">Ptit prince</a></li>
                            <li><a href="#">Mr Fan</a></li>
                        </ul>
                    </div>
                </footer>
            </div>
        </body>
    </html>
```

Petite particularité : comme vous le voyez, tout le contenu de la page est placé dans une grande balise ** ** ayant pour id `bloc_page` (on l'appelle aussi parfois `main_wrapper` en anglais). Cette balise englobe tout le contenu, ce qui va nous permettre de fixer facilement les dimensions de la page et de centrer notre site à l'écran.

Pour le reste, aucune grosse difficulté à signaler. Notez que je n'ai pas forcément pensé à toutes les balises du premier coup : en réalisant le design en CSS parfois il m'est apparu qu'il était nécessaire d'englober une partie des balises d'un bloc ** ** pour m'aider dans la réalisation du design.

Pour le moment, comme vous vous en doutez, le site web n'est pas bien beau (et encore, je suis gentil) :

C'est en CSS que la magie va maintenant opérer. 

Mettre en forme en CSS

Les choses se compliquent un peu plus lorsqu'on arrive au CSS. En effet, il faut du travail (et parfois un peu d'astuce) pour obtenir un résultat se rapprochant de la maquette. Je dis bien "se rapprochant", car vous ne pourrez jamais obtenir un résultat identique au pixel près.

Mettez-vous bien cela dans la tête : le but est d'obtenir le rendu **le plus proche possibl**e, sans chercher la perfection. Même si vous obtenez selon vous "la perfection" sur un navigateur, vous pouvez être sûr qu'il y aura des différences sur un autre navigateur (plus ancien) ou sur une autre machine que la vôtre. Nous allons donc faire au mieux, et ce sera déjà du travail vous verrez. 

Pour mettre en forme le design, je vais procéder en plusieurs temps. Je vais m'occuper des éléments suivants dans cet ordre :

1. Polices personnalisées
2. Définition des styles principaux de la page (largeur du site, fond, couleur par défaut du texte)
3. En-tête et liens de navigation
4. Bannière (représentant le pont de San Francisco)
5. Section principale du corps de page, au centre
6. Pied de page (footer)

Les polices personnalisées

Pour les besoins du design, mon graphiste a utilisé 3 polices sur sa maquette :

- Trebuchet MS (police courante)
- BallparkWeiner (police exotique, trouvée sur dafont.com)
- Day Roman (police exotique, trouvée sur dafont.com)

La plupart des ordinateurs sont équipés de Trebuchet MS (quoique pas nécessairement tous, on pourrait la faire télécharger). Par contre, les deux autres polices sont un peu originales et ne sont sûrement pas présentes sur les ordinateurs de vos visiteurs. Nous allons les leur faire télécharger.

Comme vous le savez, il faut proposer plusieurs versions de ces polices pour les différents navigateurs. Dafont ne propose de télécharger que le `.ttf`. Par contre, FontSquirrel propose un générateur de polices à utiliser en CSS3 avec @font-face : vous lui envoyez un `.ttf`, l'outil transforme le fichier dans tous les autres formats nécessaires et vous fournit même le code CSS prêt à l'emploi !

Je m'en suis servi pour générer les différentes versions des 2 polices exotiques que je vais utiliser. Ensuite, dans mon fichier CSS, je rajoute ce code qui m'a été fourni par FontSquirrel pour déclarer les nouvelles polices :

Code : CSS

```
/* Définition des polices personnalisées */

@font-face
{
    font-family: 'BallparkWeiner';
    src: url('polices/ballpark.eot');
    src: url('polices/ballpark.eot?#iefix') format('embedded-
opentype'),
         url('polices/ballpark.woff') format('woff'),
         url('polices/ballpark.ttf') format('truetype'),
         url('polices/ballpark.svg#BallparkWeiner') format('svg');
    font-weight: normal;
    font-style: normal;
}

@font-face
{
    font-family: 'Dayrom';
    src: url('polices/dayrom.eot');
    src: url('polices/dayrom.eot?#iefix') format('embedded-
opentype'),
         url('polices/dayrom.woff') format('woff'),
         url('polices/dayrom.ttf') format('truetype'),
         url('polices/dayrom.svg#Dayrom') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

En plus de ça, il faut bien entendu placer les fichiers de police. Comme vous le voyez, j'ai créé un sous-dossier `polices` dans lequel j'ai mis les différentes versions de mes polices.

Définition des styles principaux

On peut maintenant s'attaquer à définir quelques styles globaux pour tout le design de notre page. On va définir une image de fond, une police et une couleur de texte par défaut, et surtout on va dimensionner notre page et la centrer à l'écran.

Code : CSS

```
/* Eléments principaux de la page */

body
{
    background: url('images/fond_jaune.png');
    font-family: 'Trebuchet MS', Arial, sans-serif;
```

```
        color: #181818;
    }

    #bloc_page
    {
        width: 900px;
        margin: auto;
    }

    section h1, footer h1, nav a
    {
        font-family: Dayrom, serif;
        font-weight: normal;
        text-transform: uppercase;
    }
```

Avec `#bloc_page`, le bloc qui englobe toute la page, j'ai défini les limites à 900 pixels de large. Avec les marges automatiques, le design sera centré.

Si on souhaite créer un design qui s'adapte aux dimensions de l'écran du visiteur, définissez une largeur en pourcentage plutôt qu'en pixels.

J'ai utilisé la propriété CSS `text-transform: uppercase;` (que nous n'avons pas vue auparavant) pour faire en sorte que mes titres soient toujours écrits en majuscules. Cette propriété transforme en effet le texte en majuscules (elle peut aussi faire l'inverse). Notez qu'on aurait pu aussi écrire les titres directement en majuscules dans le code HTML.

Voici ce qu'on obtient avec le code CSS pour le moment :

On est encore loin du résultat final, mais on se sent déjà un petit peu plus "chez soi".

En-tête et liens de navigation

D'après la structure que j'ai proposée, l'en-tête contient aussi les liens de navigation. Commençons par définir l'en-tête, et en particulier le logo en haut à gauche, et nous verrons ensuite comment mettre en forme les liens de navigation.

L'en-tête

Code : CSS

```
/* Header */

header
{
    background: url('images/separateur.png') repeat-x bottom;
}

#titre_principal
{
    display: inline-block;
}

header h1
{
```

```
    font-family: 'BallparkWeiner', serif;
    font-size: 2.5em;
    font-weight: normal;
}

#logo, header h1
{
    display: inline-block;
    margin-bottom: 0px;
}

header h2
{
    font-family: Dayrom, serif;
    font-size: 1.1em;
    margin-top: 0px;
    font-weight: normal;
}
```

Nous créons une séparation entre l'en-tête et le corps de page sous forme d'image de fond. Les éléments sont positionnés en `inline-block` et nous personnalisons les polices et les dimensions. Rien d'extraordinaire pour le moment.

Les liens de navigation

La mise en forme des liens de navigation est un petit peu plus intéressante. Vous l'avez vu, j'ai créé une liste à puces pour les liens... mais une telle liste s'affiche habituellement en hauteur, et non en largeur. Heureusement, cela se change facilement vous allez voir :

Code : CSS

```
/* Navigation */

nav
{
    display: inline-block;
    width: 740px;
    text-align: right;
}

nav ul
{
    list-style-type: none;
}

nav li
{
    display: inline-block;
    margin-right: 15px;
}

nav a
{
    font-size: 1.3em;
    color: #181818;
    padding-bottom: 3px;
    text-decoration: none;
}

nav a:hover
{
    color: #760001;
    border-bottom: 3px solid #760001;
}
```

La principale nouveauté est la définition CSS `list-style-type: none;`, qui permet de retirer l'image ronde des puces. Chaque élément de la liste (`<li>`) est positionné en `inline-block`, ce qui nous permet de placer les liens côte à côte comme nous le souhaitions.

Le reste des définitions ne contient rien d'extraordinaire : des dimensions, des couleurs, des bordures... Autant de choses que vous connaissez déjà. Notez que je ne trouve pas les bonnes valeurs forcément du premier coup, il me faut parfois tâtonner un peu pour retrouver une apparence proche de la maquette d'origine.

Voici le résultat que nous obtenons avec les derniers ajouts de CSS :

La bannière

Bien, passons maintenant à un exercice un peu plus difficile mais très intéressant : la bannière ! Notre maquette comporte une jolie bannière représentant le pont de San Francisco. Cette bannière, sur votre site, peut être amenée à évoluer. Ici, elle peut servir à illustrer par exemple le dernier billet de blog de notre ami Zozor, qui vient de visiter San Francisco.

La bannière est intéressante à plus d'un titre :

- Elle comporte des angles arrondis
- La description est écrite sur un fond légèrement transparent
- Le bouton "Voir l'article" est réalisé en CSS, avec des angles arrondis
- Une ombre vient donner du volume à la bannière

Voici le code que j'ai utilisé pour réaliser toute la bannière :

Code : CSS

```
/* Bannière */
```

```
#banniere_image
{
    margin-top: 15px;
    height: 200px;
    border-radius: 5px;
    background: url('images/sanfrancisco.jpg') no-repeat;
    position: relative;
    box-shadow: 0px 4px 4px #1c1a19;
    margin-bottom: 25px;
}

#banniere_description
{
    position: absolute;
    bottom: 0;
    border-radius: 0px 0px 5px 5px;
    width: 99.5%;
    height: 33px;
    padding-top: 15px;
    padding-left: 4px;
    background-color: rgb(24,24,24); /* Pour les anciens navigateurs
*/
    background-color: rgba(24,24,24,0.8);
    color: white;
    font-size: 0.8em;

}

.bouton_rouge
{
    display: inline-block;
    height: 25px;
    position: absolute;
    right: 5px;
    bottom: 5px;
    background: url('images/fond_degraderouge.png') repeat-x;
    border: 1px solid #760001;
    border-radius: 5px;
    font-size: 1.2em;
    text-align: center;
    padding: 3px 8px 0px 8px;
    color: white;
    text-decoration: none;
}

.bouton_rouge img
{
    border: 0;
}
```

Ce code est assez technique et riche en fonctionnalités CSS. C'est peut-être la partie la plus délicate de la page à réaliser.

Vous pouvez constater que j'ai choisi d'afficher l'image du pont sous forme d'image de fond dans le bloc ** ** de la bannière.

J'ai aussi donné une position relative à la bannière, sans utiliser de propriétés pour en modifier le décalage... Pourquoi ? A priori, une position relative sans décalage ne sert à rien... Et pourtant, cela m'a été particulièrement utile pour placer le bouton "Voir l'article" en bas à droite de la bannière. En effet, j'ai placé le bouton en absolu à l'intérieur.

Le bouton ne devrait pas se placer en bas à droite de la page ?

Non, souvenez-vous ce que je vous avais dit : si un bloc est positionné en absolu dans un autre bloc lui-même positionné en absolu, fixe ou relatif, alors il se positionne *à l'intérieur* de ce bloc.
Notre bannière est positionnée en relatif (sans décalage). Comme le bouton est positionné en absolu à l'intérieur, il se place donc

en bas à droite de la bannière !

C'est une technique particulièrement utile et puissante dans la réalisation d'un design, souvenez-vous en !

Dernier détail : pour la légende de la bannière, j'ai choisi d'utiliser la transparence avec la notation RGBa plutôt que la propriété **opacity**. En effet, **opacity** aurait rendu tout le contenu du bloc transparent, y compris le bouton "Voir l'article" à l'intérieur. J'ai trouvé préférable de rendre seulement la couleur de fond transparente plutôt que tout le bloc.

Le résultat est plutôt sympathique.

Ca en jette, vous ne trouvez pas ?

Pour réaliser le dégradé du bouton "Voir l'article", j'ai utilisé une image de fond représentant le dégradé que j'ai répétée horizontalement. Sachez qu'il existe une propriété CSS3 **linear-gradient** qui permet de réaliser des dégradés sans avoir à recourir à une image de fond. Son usage étant un peu complexe actuellement, j'ai choisi de ne pas l'utiliser dans cet exemple, mais vous pouvez vous documenter à son sujet si vous le souhaitez !

Le corps

Le corps, au centre de la page, est dans notre cas constitué d'une unique balise **<section>** (mais il pourrait y en avoir plusieurs bien sûr).

Pas beaucoup de difficultés pour le corps, le positionnement du bloc "A propos de l'auteur" se fait en inline-block. On joue avec les angles arrondis et les ombres, on ajuste un peu les marges et les dimensions du texte, et nous y voilà !

Code : CSS

```
/* Corps */

article, aside
{
    display: inline-block;
```

```
        vertical-align: top;
        text-align: justify;
    }

    article
    {
        width: 625px;
        margin-right: 15px;
    }

    .ico_categorie
    {
        vertical-align: middle;
        margin-right: 8px;
    }

    article p
    {
        font-size: 0.8em;
    }

    aside
    {
        position: relative;
        width: 235px;
        background-color: #706b64;
        box-shadow: 0px 2px 5px #1c1a19;
        border-radius: 5px;
        padding: 10px;
        color: white;
        font-size: 0.9em;
    }

    #fleche_bulle
    {
        position: absolute;
        top: 100px;
        left: -12px;
    }

    #photo_zozor
    {
        text-align: center;
    }

    #photo_zozor img
    {
        border: 1px solid #181818;
    }

    aside img
    {
        margin-right: 5px;
    }
```

La petite difficulté ici était de réussir à placer la flèche à gauche du bloc **<aside>** "A propos de l'auteur" pour donner l'effet d'une bulle. Là encore, notre meilleur ami est le positionnement absolu. La technique est la même : je positionne le bloc **<aside>** en relatif (sans effectuer de décalage), ce qui me permet ensuite de positionner l'image de la flèche en absolu par rapport au bloc **<aside>** (et non par rapport à la page entière). En jouant sur le décalage de l'image, je peux la placer avec précision où je veux, au pixel près ! 

Le pied de page

Il ne nous reste plus que le pied de page à mettre en forme. Il est constitué de 3 sous-blocs que j'ai matérialisés par des ** ** auxquels j'ai donné des id pour mieux les repérer. Ces blocs sont positionnés en `inline-block` les uns à côté des autres.

Code : CSS

```
/* Footer */

footer
{
    background:  url('images/ico_top.png') no-repeat top center,
url('images/separateur.png') repeat-x top, url('images/ombre.png')
repeat-x top;
    padding-top: 25px;
}

footer p, footer ul
{
    font-size: 0.8em;
}

footer h1
{
    font-size: 1.1em;
}

#tweet, #mes_photos, #mes_amis
{
    display: inline-block;
    vertical-align: top;
}
```

```
#tweet
{
    width: 28%;
}

#mes_photos
{
    width: 35%;
}

#mes_amis
{
    width: 31%;
}

#mes_photos img
{
    border: 1px solid #181818;
    margin-right: 2px;
}

#mes_amis ul
{
    display: inline-block;
    vertical-align: top;
    margin-top: 0;
    width: 48%;
    list-style-image: url('images/ico_liensexterne.png');
    padding-left: 2px;
}

#mes_amis a
{
    text-decoration: none;
    color: #760001;
}
```

Deux petites particularités à signaler sur le pied de page :

- J'ai utilisé la fonctionnalité des images de fond multiples de CSS3, ce qui m'a permis de réaliser le séparateur entre le corps et le pied de page. Il est constitué de 3 images : le séparateur, la petite flèche vers le haut et un léger dégradé.
- J'ai modifié la puce de la liste "Mes amis" en bas à droite avec la propriété **`list-style-image`**, ce qui m'a permis d'utiliser une image personnalisée pour les puces. Il existe de nombreuses propriétés CSS spécifiques comme celle-ci et nous ne pouvons pas toutes les voir une par une dans ce cours, mais maintenant que vous êtes des habitués du CSS vous n'aurez aucun mal à apprendre à les utiliser simplement en lisant l'annexe listant les principales propriétés CSS.

Et voilà, notre design est terminé ! 

Ah, vous pensez en avoir fini ? Il reste hélas encore un peu de travail : il faut tester notre site sur différents navigateurs. Idéalement, il vaut mieux le faire au fur et à mesure de la mise en place du design. En particulier, les anciennes versions d'Internet Explorer (IE6 à IE8) méritent qu'on s'y attarde, car le résultat n'est pas forcément celui auquel on s'attendait...

Assurer la compatibilité avec IE

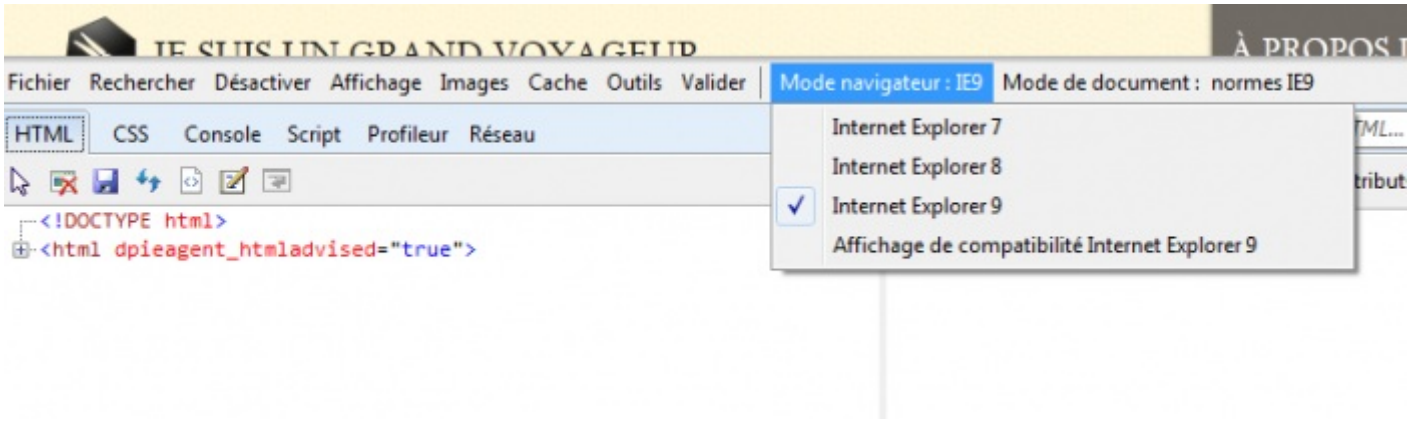
Depuis Internet Explorer 9 (IE9), nous n'avons plus vraiment de raisons de nous plaindre du légendaire retard d'Internet Explorer dans la gestion du CSS. Voyez vous-mêmes le résultat, il est très bon sur ce navigateur sans aucune adaptation nécessaire :

Par contre, vous risquez d'avoir quelques cheveux blancs en regardent le résultat sur les anciennes versions d'Internet Explorer.

Comment je fais pour voir le résultat sous IE6 à IE8, si je suis équipé d'IE9 ?

Je vous avais parlé d'IETester, un outil pratique mais instable (il plante souvent). Vous pouvez l'essayer pour tester votre site sur les anciennes versions d'IE.

Il y a cependant une solution plus stable et plus rapide : appuyez sur F12 dans IE9, et une barre dédiée aux développeurs web apparaît. Là, un menu vous permet de changer le comportement d'IE, qui se comportera alors comme sur les anciennes versions (à partir de IE7) :



Là, en général, il arrive qu'on prenne très peur.
Voici ce qu'on obtient en "mode IE7" :

Avant de préparer votre valise pour aller vivre en Patagonie loin de ce monde cruel, laissez-moi vous redonner le moral avec cette phrase rassurante : *tout problème a une solution* (répétez-le autant de fois que nécessaire).

En fait, notre site ne rencontre que 2 principaux problèmes sur les anciennes versions d'IE :

- Le positionnement `inline-block` n'est pas bien géré sous IE6 et IE7, ce qui fait que la plupart de nos positionnements ne fonctionnent pas pour le moment... Mais nous avions vu qu'une astuce permettait de régler le problème sans trop d'efforts !
- Les balises structurantes de HTML5 (**`<header>`**, **`<nav>`**, **`<aside>`**...) ne sont pas gérées sous IE6, IE7 et IE8, ce qui pose de gros problèmes d'affichage... mais là encore, un petit script rajouté au début de votre code HTML permet de régler le problème !

Par contre, il faudra faire une croix sur certaines fonctionnalités plus récentes de CSS3 qui ne sont pas gérées sur ces vieilles versions :

- Les coins arrondis
- Les images de fond multiples
- La transparence
- Les ombres

Etant donné que ce sont des fonctionnalités liées à l'apparence, nous ne chercherons pas à les faire fonctionner sur les anciennes versions d'IE. Si toutefois vous y tenez, sachez que là encore des scripts existent et permettent d'émuler la plupart de ces fonctionnalités manquantes, mais cela vous demandera pas mal de travail supplémentaire et votre site risque d'être plus lent sur ces navigateurs. Du temps que le site reste lisible sur les anciennes versions d'IE, je vous recommande de ne pas vous préoccuper trop à ce sujet.

Lorsqu'on accepte que son site soit "un peu moins beau" sur les anciennes versions des navigateurs, on dit qu'on fait

une *dégradation gracieuse*. Cela veut dire qu'on ne cherche pas à obtenir tous les effets, mais on s'efforce d'avoir un site qui reste quand même lisible sur les vieux navigateurs.

Faire fonctionner les balises structurantes de HTML5

Nous l'avons vu, il suffit de rajouter un simple bout de code Javascript dans l'en-tête de son site, et le tour est joué :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
<!--[if lt IE 9]>
<script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
        <title>Zozor - Carnets de voyage</title>
    </head>
```

Le fichier Javascript sera téléchargé depuis les serveurs de Google : c'est une technique rapide qui vous évite d'avoir à gérer vous-même le fichier. Si vous le souhaitez, vous pouvez aussi en récupérer une copie et la placer à côté des fichiers de votre site.

Régler le positionnement inline-block

Pour gérer le positionnement `inline-block`, nous avons vu qu'il était nécessaire de créer une feuille de style spéciale pour les anciennes versions d'Internet Explorer. Il faut utiliser un CSS un peu différent pour que les vieilles versions d'IE "comprennent" ce qu'il faut faire.

En créant une feuille de style spéciale pour les vieilles versions d'IE (qu'on pourrait appeler `style_ie.css`), et en utilisant la technique ci-dessous, on peut reproduire le comportement des `inline-block` :

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <!--[if lte IE 7]>
<link rel="stylesheet" href="style_ie.css" />
<![endif]-->

        <title>Zozor - Le Site Web</title>
    </head>
```

La feuille de style `style_ie.css` contiendra des déclarations comme celle-ci :

Code : CSS

```
element
{
    display: inline;
```

```
            zoom: 1;
        }
```

Cette technique doit être appliquée sur chaque élément positionné en `inline-block`.

D'autres différences existent sur les vieilles versions d'IE : le texte n'est pas toujours à la bonne taille, certains blocs eux aussi sont mal dimensionnés, etc. Ces différences doivent être réglées au cas par cas dans la feuille `style_ie.css`.

Vérifier la validité

Le W3C propose sur son site web un outil appelé le "Validateur" ("Validator" en anglais).
Le validateur 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.

Souvenez-vous : le W3C a établi des normes. Il est nécessaire de les respecter, pour qu'on soit sûr que tous les sites web parlent la même "langue".

Il existe un validateur pour HTML et un validateur pour CSS. Celui pour CSS comportant quelques bugs (il signale des feuilles CSS invalides alors qu'elles sont valides), nous ne nous y attarderons pas. Par contre, le validateur HTML va être très intéressant pour nous.

Voici l'adresse du validateur HTML (à mettre dans vos favoris !) :

http://validator.w3.org

Vous pouvez valider votre page web de 3 façons différentes, c'est pour cela qu'il y a 3 onglets :

- Adresse (URL)
- Envoi du fichier .html
- Copier-coller du code HTML

Pour le moment, notre site web n'est pas encore disponible sur le Web, ce qui fait qu'il n'a pas d'adresse URL. Le mieux est donc d'envoyer le fichier `.html` que l'on a fait, ou encore de copier-coller directement le code HTML.

Si on envoie notre code HTML et que tout se passe bien, le validateur va nous répondre avec ce message :

Dans ce cas, cela signifie que tout va bien et que vous avez bien construit votre page !

Malheureusement, il arrivera souvent que vous ayez des erreurs. Dans ce cas, évitez de paniquer comme ça :

Au SEEECOUUUUUUUURS !!!
Ma page web n'est pas valide, je vais pas m'en sortir je suis cerné par les erreurs, faites quelque chose aidez-mmmoiiiii !

Vous aviez une belle page web, elle s'affichait bien, elle était jolie, mais pourtant le validateur vous répond avec un message rouge inquiétant, en vous disant que votre page web n'est pas bien construite.

Tout d'abord, mettez-vous bien ceci 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 beaucoup d'erreurs.

Quel intérêt de les corriger alors ?

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. Cela simplifie le travail des programmes qui lisent les pages web.
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 !

Voici une liste de conseils qui peuvent vous aider à résoudre les erreurs que l'on risque de vous signaler tôt ou tard :

- Tous vos textes doivent en général 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 />`, qui doivent être à l'intérieur de paragraphes.
  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 />
```

- **Toutes vos images doivent comporter un attribut "alt"** qui indique ce que contient l'image :
  `<img src="photo.jpg" alt="Une photo de moi" />`
  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 :
  `<img src="deco.png" alt="" />`
- 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 `&`, vous devez taper le code `&` 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 `&` :
  `<a href="http://www.site.com/?jour=15&mois=10&an=2000">`
- Vérifiez enfin que vous n'avez pas utilisé d'anciennes balises, désormais obsolètes en HTML5 (comme le vieux `<frame>`, la balise `<marquee>`....
  Le validateur vous dira "Element *XXX* undefined" (balise inconnue) ou encore "There is no attribute *XXX*" (attribut inconnu).

Tout le monde fait des erreurs, alors ne paniquez pas. Corrigez pas à pas votre page web jusqu'à ce que le validateur vous affiche un beau résultat en vert. 

Le code final

Je mets à disposition le code final de la page web que nous avons réalisée. Vous pouvez visualiser le résultat en ligne :

Essayer !

Vous pouvez aussi télécharger un fichier ZIP contenant tous les fichiers du site pour pouvoir le tester chez vous :

Télécharger les fichiers du site (500 Ko)

Pour ajuster au mieux le site sur les anciennes versions d'Internet Explorer, vous verrez je n'ai pas créé dans ce code final de feuille `style_ie.css`. A la place, j'ai utilisé une autre technique, qui consiste à donner une classe spéciale (comme `.ie7`) à la balise `<body>` uniquement sur les anciennes versions d'IE, ce qui me permet de savoir quelle est la version du navigateur dans le fichier CSS (`.ie7 footer` permet par exemple de modifier le style du footer sur IE7).

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.

Prenez bien le temps de lire et relire ce TP pour comprendre comment je procède. Je suis confiant : à force de pratiquer, vous allez devenir des as de la création de sites web !

Partie 4 : Fonctionnalités évoluées

A ce stade, vous êtes déjà capables de réaliser sans problème votre site web.
Nous allons découvrir dans cette partie de nouvelles fonctionnalités de HTML et CSS, que l'on peut considérer un peu plus évoluées (et donc un peu plus complexes).

Les tableaux

Indispensables pour organiser nos informations, les tableaux sont un petit peu délicats à construire en HTML, ce qui explique que je ne vous les présente que maintenant.

Vous voyez à quoi ressemble un tableau, je ne vous fais pas l'affront de vous en dessiner un ?... 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

C'est un tableau assez basique. Bien entendu, nous apprendrons à faire des tableaux plus complexes afin que vous puissiez réaliser tout ce dont vous avez besoin.
Nous découvrirons aussi les propriétés CSS liées aux tableaux, qui nous permettront de personnaliser leur apparence.

Un tableau simple

La première balise à connaître est **<table> </table>**. C'est cette balise qui permet d'indiquer le début et la fin d'un tableau.

Cette balise est de type block, il faut donc la placer en dehors d'un paragraphe. Exemple :

Code : HTML

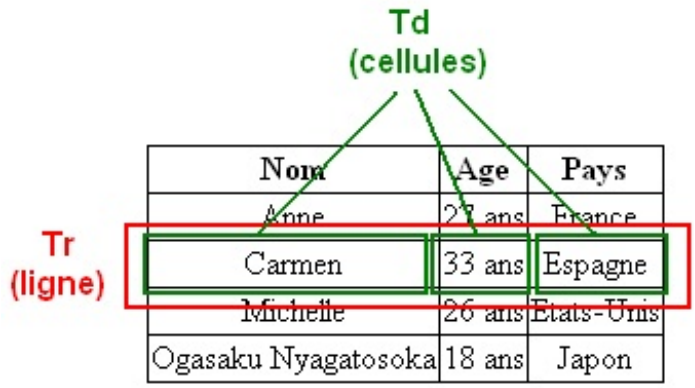
```
<p>Ceci est un paragraphe avant le tableau.</p>

<table>
   <!-- Ici, on écrira le contenu du tableau -->
</table>

<p>Ceci est un paragraphe après le tableau.</p>
```

Bon, et qu'écrit-on à l'intérieur du tableau ?

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 HTML, 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 :



On a une balise de ligne (**<tr>**) qui englobe chacune des cellules (**<td>**)
Par exemple, si je veux faire un tableau à deux lignes, avec 3 cellules par ligne (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>
```

Le résultat est un peu déprimant :

Carmen 33 ans Espagne
Michelle 26 ans Etats-Unis

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 paraît bien vide. Alors justement, rajouter des bordures est très simple, vous connaissez déjà le code CSS correspondant !

Code : CSS

```
td /* Toutes les cellules des tableaux... */
{
    border: 1px solid black; /* ... auront une bordure de 1px */
}
```

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

Hum, 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;
}
```

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

Voilà qui est mieux ! 

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).

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;
}
```

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

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.

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 perdu.

Heureusement, il y a **<caption>** !
Cette balise se place 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>
```

Passagers du vol 377

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

C'est quand même plus clair ! 

Sachez que vous pouvez changer la position du titre avec la propriété CSS **caption-side**, qui peut prendre quatre valeurs :

- `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.

Un tableau structuré

Nous avons appris à construire des petits tableaux simples. Ces petits tableaux suffisent dans la plupart des cas, mais il arrivera que vous ayez besoin de réaliser des tableaux plus... complexes.

Nous allons découvrir 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.

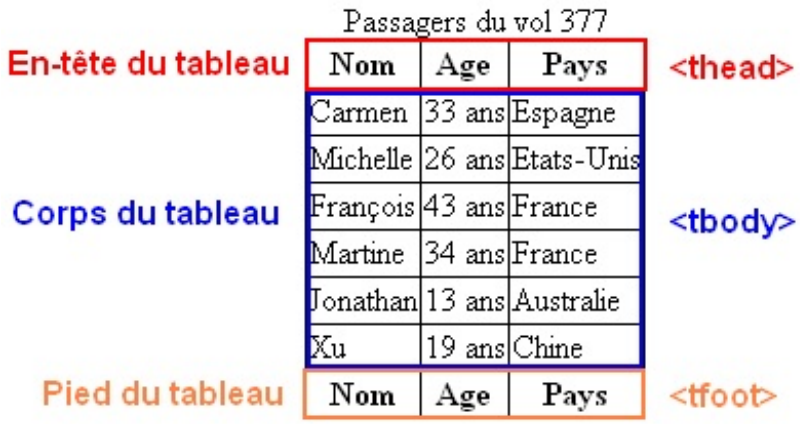
Diviser un gros tableau

Si votre tableau est assez gros, vous aurez tout intérêt à le découper en plusieurs parties. Pour cela, il existe des balises HTML 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 y recopiez les cellules d'en-tête. Ca 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 :



C'est un peu déroutant, mais il est conseillé d'écrire 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 construire 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>
```

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 sans problème l'organisation plus simple qu'on a vue au début.

3, 2, 1... Fusioooon !

Sur certains tableaux complexes, vous aurez besoin de "fusionner" des cellules entre elles.
Un exemple de fusion ? Regardez, ce tableau qui liste des films et qui 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`). Il faut indiquer 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 :

Code : HTML

```
<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 HTML qui me permet de réaliser la fusion correspondant au tableau précédent :

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>
```

Une 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.

Et pour la fusion verticale avec `rowspan`, on fait comment ?

Ca se 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 placer en haut.
C'est une autre façon de voir le tableau : au lieu de le construire en hauteur, on peut le construire en longueur.

Dans ce cas, le `colspan` n'est plus adapté, c'est un `rowspan` qu'il faut utiliser :

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>
```

Résultat : les cellules sont fusionnées verticalement !

Titre du film	Massacre à la tronçonneuse	Les bisounours font du ski	Lucky Luke, seul contre tous
Pour enfants ?	Non, trop violent	Oui, adapté	Pour toute la famille !
Pour adolescents ?	Oui	Pas assez violent...	

Notez qu'on peut modifier l'alignement vertical du texte des cellules de tableaux, avec la propriété **`vertical-align`** que nous avons découverte dans le chapitre sur la mise en page.

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... ), vous avez toutes les chances de vous y habituer en peu de temps.

Je vous conseille surtout de 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 n'est... pas très esthétique. Profitez-en donc pour mettre à l'épreuve vos connaissances en CSS, c'est l'occasion idéale !

Les formulaires

Toute page HTML peut être enrichie de formulaires interactifs, qui invitent vos visiteurs à renseigner des informations : saisir du texte, sélectionner des options, valider avec un bouton... tout est possible !

Nous arrivons cependant aux limites du langage HTML, car il faut ensuite pouvoir analyser les informations que le visiteur a saisies... et cela ne peut pas se faire en langage HTML. Le traitement des résultats doit s'effectuer dans un autre langage comme nous le verrons, tel que le PHP.

En attendant, nous avons un grand nombre de nouvelles balises HTML à découvrir. 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 HTML, vous devez pour commencer écrire 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 souhaitez écrire du texte à l'intérieur.

Voilà pour la structure de base. Maintenant soyez attentifs, parce que ce que j'ai à vous dire n'est pas évident étant donné qu'on est à la limite du HTML.

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 un message qu'il aimerait poster sur vos forums. Ce message doit être *envoyé* pour que vous puissiez le recevoir (logique non ?) et l'afficher à vos autres visiteurs.

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 ?

Vous devez ajouter 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 (réponse au **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 envoyer un grand nombre d'informations grâce à elle. Les données saisies dans le formulaire ne transitent pas par la barre d'adresse.
- `action` : c'est l'adresse de la page ou du programme qui va *traiter* les informations (réponse au **problème n°2**). Cette page se chargera de vous envoyer un e-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 HTML et CSS, on utilisera en général un autre langage dont vous avez peut-être entendu parler : PHP.

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 fictive 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.

Notre priorité pour le moment est de découvrir en HTML / CSS comment faire pour insérer des zones de texte, des boutons et des cases à cocher dans votre page web. C'est ce que nous allons voir maintenant.

Les zones de saisie basiques

Bien, retour au concret.

Nous allons passer en revue les différentes balises HTML permettant de rentrer du texte dans un formulaire.
Il faut savoir qu'il y a 2 zones de texte différentes :

- **La zone de texte monoligne** : comme son nom l'indique, on ne peut écrire qu'une seule ligne à l'intérieur. Elle sert à 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 : "Rédigez une dissertation sur l'utilité du HTML dans le développement des pays d'Asie du Sud-Est"

Zone de texte monoligne

Voici à quoi ressemble une zone de texte monoligne :

Pour insérer une zone de texte à une ligne, on va utiliser la balise **<input />**.

On retrouvera cette balise plusieurs fois par la suite dans ce chapitre. A chaque fois, c'est la valeur de son attribut `type` qui va changer.

Pour créer une zone de texte à une ligne, on doit écrire :

Code : HTML

```
<input type="text" />
```

Ce n'est pas encore suffisant : il faut donner un nom à votre zone de texte. Ce nom n'apparaît pas sur la page, mais il vous sera indispensable par la suite. En effet, cela vous permettra (en PHP par exemple) de reconnaître d'où viennent les informations :

vous saurez que tel texte est le pseudo du visiteur, tel texte est le mot de passe qu'il a choisi, etc.

Pour donner un nom à un élément de formulaire, on utilise l'attribut `name`. Ici, on va supposer qu'on demande au visiteur de rentrer son pseudo :

Code : HTML

```
<input type="text" name="pseudo" />
```

Essayons donc de créer un formulaire très basique avec notre champ de texte :

Code : HTML

```
<form method="post" action="traitement.php">
   <p><input type="text" name="pseudo" /></p>
</form>
```

Essayer !

Les libellés

Cette zone de texte est bien jolie, mais si votre visiteur tombe dessus il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise **<label>** :

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... Le mieux est de le voir sur un exemple :

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.

Quelques attributs supplémentaires

On peut placer un certain nombre d'autres attributs sur la balise `<input />` pour personnaliser son fonctionnement :

- On peut agrandir le champ avec `size`.
- On peut limiter le nombre de caractères que l'on peut saisir avec `maxlength`.
- On peut pré-remplir le champ avec une valeur par défaut avec `value`.
- On peut donner une indication sur le contenu du champ avec `placeholder`. Cette indication disparaîtra dès que le visiteur aura cliqué à l'intérieur du champ.

Dans l'exemple suivant, la zone de texte contient une indication permettant de comprendre ce qu'il faut saisir, elle fait 30 caractères de long mais on ne peut écrire 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" placeholder="Ex
: Zozor" size="30" maxlength="10" />
    </p>
</form>
```

Testez vous-mêmes le résultat pour voir le comportement du champ :

Votre pseudo : Ex : Zozor

Essayer !

Zone de mot de passe

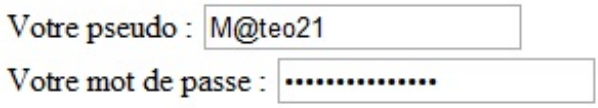
Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une zone "mot de passe", c'est-à-dire une zone où on ne voit pas à l'écran les caractères saisis. Pour créer ce type de zone de saisie, utilisez l'attribut `type="password"`.

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" />

        <br />
        <label for="pass">Votre mot de passe :</label>
        <input type="password" name="pass" id="pass" />

   </p>
</form>
```

Testez la zone de mot de passe : vous verrez que les caractères ne s'affichent pas à l'écran.



Essayer !

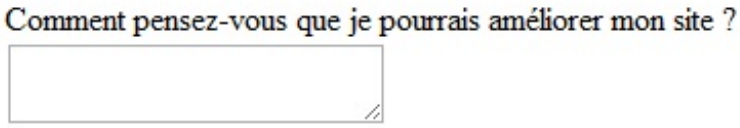
Zone de texte multiligne

Pour créer une zone de texte multiligne, on change de balise : nous allons utiliser **<textarea></textarea>**.

Comme pour 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>
```



C'est un peu petit comme vous pouvez le constater ! Heureusement, 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 de texte qui peuvent être affichées simultanément, et le second le nombre de colonnes.

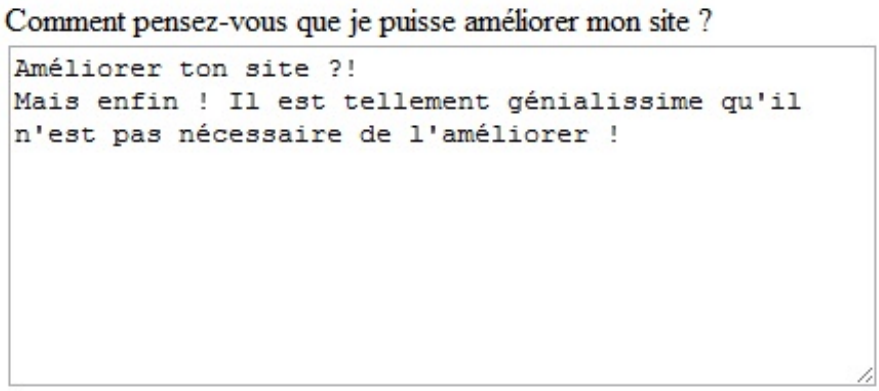
Pourquoi ouvre-t-on la balise **<textarea>** pour la fermer juste après ?

Vous pouvez pré-remplir le **<textarea>** avec une valeur par défaut. Dans ce cas, on n'utilise pas l'attribut `value` : on écrit tout simplement le texte par défaut entre la balise ouvrante et la balise fermante !

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'est pas
```

```
nécessaire de l'améliorer !</textarea>
   </p>
</form>
```



Essayer !

Les zones de saisie enrichies

HTML5 apporte de nombreuses nouvelles fonctionnalités aux formulaires. De nouveaux types de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut `type` de la balise `<input />` l'une des nouvelles valeurs disponibles. Faisons un petit tour d'horizon !

Tous les navigateurs ne connaissent pas encore ces zones de saisie enrichies. Les anciennes versions des navigateurs afficheront une simple zone de saisie monoligne à la place (comme si on avait écrit `type="text"`). Entre nous, c'est parfait : les nouveaux navigateurs peuvent profiter des nouvelles fonctionnalités, tandis que les anciens affichent une zone de texte de remplacement qui convient tout aussi bien.
Vous avez donc tout intérêt à utiliser ces nouvelles zones de saisie dès aujourd'hui ! Au mieux, vos visiteurs profiteront des nouvelles fonctionnalités, au pire, ils ne verront aucun problème.

E-mail

Vous pouvez demander à saisir une adresse e-mail :

Code : HTML

```
<input type="email" />
```

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

Sachez que certains navigateurs, comme les navigateurs mobiles des iPhone et Android, affichent un clavier adapté à la saisie d'e-mail dans ce cas de figure :

Une URL

Avec le type `url`, on peut demander à saisir une adresse absolue (commençant généralement par http://) :

Code : HTML

```
<input type="url" />
```

Même principe : si le champ ne vous semble pas différent sur votre ordinateur, sachez que votre ordinateur comprend bel et bien que le visiteur est censé entrer une adresse. Les navigateurs mobiles affichent par exemple un clavier adapté à la saisie d'adresses.

Numéro de téléphone

Ce champ est dédié à la saisie de numéros de téléphone :

Code : HTML

```
<input type="tel" />
```

Sur iPhone par exemple, un clavier adapté s'affiche lorsqu'on doit remplir le champ :

Nombre

Ce champ permet de saisir un nombre entier :

Code : HTML

```
<input type="number" />
```

Le champ s'affichera en général avec des petites flèches pour changer la valeur :

Essayer !

Vous pouvez personnaliser le fonctionnement du champ avec les attributs suivants :

- `min` : valeur minimale autorisée.
- `max` : valeur maximale autorisée.
- `step` : c'est le "pas" de déplacement. Si vous indiquez un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6...).

Un curseur

Le type `range` permet de sélectionner un nombre avec un curseur (aussi appelé *slider*) :

Code : HTML

```
<input type="range" />
```

Essayer !

Vous pouvez utiliser là aussi les attributs `min`, `max` et `step` pour définir les limites qui peuvent être sélectionnées.

Couleur

Ce champ permet de saisir une couleur :

Code : HTML

```
<input type="color" />
```

En pratique, il reste assez peu implémenté par les navigateurs à l'heure actuelle. Ne vous étonnez pas si vous voyez seulement un champ de texte classique.

Date

Différents types de champs de sélection de date existent :

- `date` : pour la date (05/08/1985 par exemple)
- `time` : pour l'heure (13:37 par exemple)
- `week` : pour la semaine
- `month` : pour le mois
- `datetime` : pour la date et l'heure (avec gestion du décalage horaire)
- `datetime-local` pour la date et l'heure (sans gestion du décalage horaire)

Exemple :

Code : HTML

```
<input type="date" />
```

Comme vous le voyez, il y a le choix !
A l'heure actuelle, peu de navigateurs gèrent ce type de champ à part Opera.

Recherche

On peut créer un champ de recherche comme ceci :

Code : HTML

```
<input type="search" />
```

Le navigateur décide ensuite comment afficher le champ de recherche. Il peut ajouter une petite loupe au champ pour signifier que c'est un champ de recherche par exemple, et éventuellement mémoriser les dernières recherches effectuées par le visiteur.

Les éléments d'options

HTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire. Ce sont des éléments qui demandent au visiteur de faire un choix parmi une liste de possibilités. Nous allons passer en revue :

- Les cases à cocher
- Les zones d'options
- Les listes déroulantes

Les cases à cocher

Créer une case à cocher ? Rien de plus simple ! Nous allons réutiliser la balise **<input />**, en spécifiant cette fois le type `checkbox` :

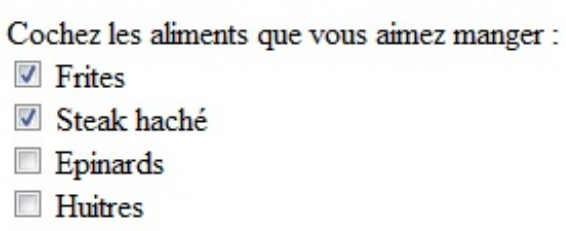
Code : HTML

```
<input type="checkbox" name="choix" />
```

Rajoutez un **<label>** bien placé, et le tour est 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 !

N'oubliez pas de donner un nom différent à chaque case à cocher, cela vous permettra d'identifier plus tard quelles cases le visiteur a coché.

Enfin, sachez que vous pouvez faire en sorte qu'une case soit cochée par défaut avec l'attribut `checked` :

Code : HTML

```
<input type="checkbox" name="choix" checked />
```

Normalement, tout attribut possède une valeur. Dans le cas présent en revanche, ajouter une valeur n'est pas obligatoire : la présence de l'attribut suffit à faire comprendre à l'ordinateur que la case doit être cochée.
Si cela vous perturbe, sachez que vous pouvez donner n'importe quelle valeur à l'attribut (certains webmasters écrivent parfois `checked="checked"`, mais c'est un peu redondant !). Dans tous les cas, la case sera cochée.

Les zones d'options

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités. Elles ressemblent un peu aux cases à cocher, mais il y a une petite difficulté supplémentaire : elles doivent être organisées en groupes. Un même groupe d'options a le même nom (`name`), mais chaque option doit avoir une valeur (`value`) différente.

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

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>
```

Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :
- ○ Moins de 15 ans
- ○ 15-25 ans
- ◉ 25-40 ans
- ○ Encore plus vieux que ça ?!

Essayer !

Pourquoi avoir mis le même nom pour chaque option ? Tout simplement pour que le navigateur sache de quel "groupe" le bouton fait partie.
Essayez d'enlever les attributs `name`, vous verrez qu'il devient possible de sélectionner tous les éléments d'option. 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>
```

L'attribut `checked` est là aussi disponible pour sélectionner une valeur par défaut.

Les listes déroulantes

Les listes déroulantes sont un autre moyen élégant de faire un choix parmi plusieurs possibilités. Le fonctionnement est un peu différent. 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.

Puis, à l'intérieur du **<select></select>**, nous allons placer plusieurs balises **<option></option>** (une par choix possible). On ajoute à chacune d'elles un attribut `value` pour pouvoir identifier ce que le visiteur a choisi.

Voici un exemple d'utilisation :

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 !

Si vous voulez qu'une option soit sélectionnée par défaut, utilisez cette fois l'attribut `selected` :

Code : HTML

```
<option value="canada" selected>Canada</option>
```

Vous pouvez aussi grouper vos options avec la balise **<optgroup></optgroup>**. 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 !

Les groupes ne peuvent pas être sélectionnés. Dans notre exemple, on ne peut pas choisir "Europe" par exemple, seuls les noms de pays sont sélectionnables.

Finaliser et envoyer le formulaire

Nous y sommes presque. Il ne nous reste plus qu'à agrémenter notre formulaire de quelques dernières fonctionnalités (comme la validation), puis nous pourrons ajouter le bouton d'envoi du formulaire.

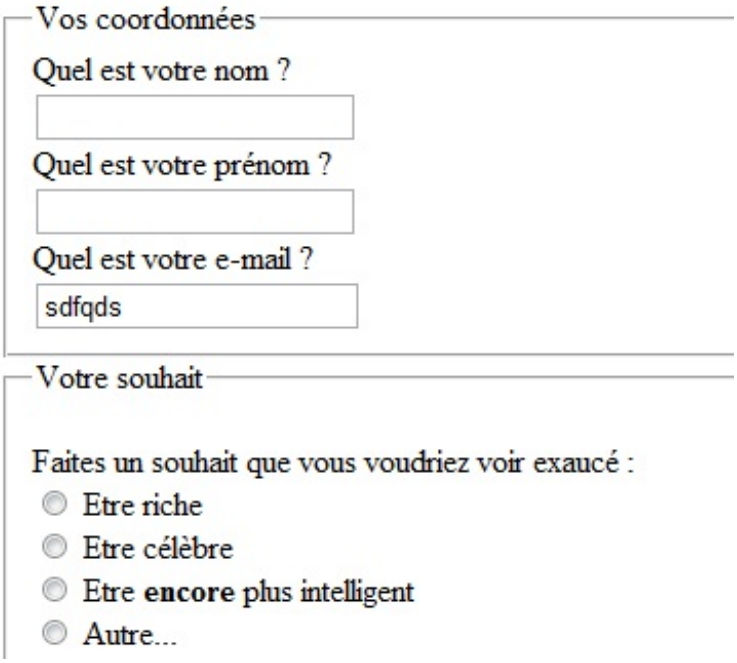
Regrouper les champs

Si votre formulaire grossit et comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises **<fieldset>**. Chaque **<fieldset>** peut contenir une légende avec la balise **<legend>**. Regardez cet exemple :

Code : HTML

```
<form method="post" action="traitement.php">

   <fieldset>
       <legend>Vos coordonnées</legend> <!-- Titre du fieldset -->

       <label for="nom">Quel est votre nom ?</label>
       <input type="text" name="nom" id="nom" />

       <label for="prenom">Quel est votre prénom ?</label>
       <input type="text" name="prenom" id="prenom" />

       <label for="email">Quel est votre e-mail ?</label>
       <input type="email" name="email" id="email" />

   </fieldset>

   <fieldset>
       <legend>Votre souhait</legend> <!-- Titre du fieldset -->

       <p>
           Faites un souhait que vous voudriez voir exaucé :

           <input type="radio" name="souhait" value="riche"
id="riche" /> <label for="riche">Etre riche</label>
           <input type="radio" name="souhait" value="celebre"
id="celebre" /> <label for="celebre">Etre célèbre</label>
           <input type="radio" name="souhait" value="intelligent"
id="intelligent" /> <label for="intelligent">Etre
<strong>encore</strong> plus intelligent</label>
           <input type="radio" name="souhait" value="autre"
```

```
id="autre" /> <label for="autre">Autre...</label>
        </p>

        <p>
            <label for="precisions">Si "Autre", veuillez préciser
:</label>
            <textarea name="precisions" id="precisions" cols="40"
rows="4"></textarea>
        </p>
    </fieldset>
</form>
```



Essayer !

Sélectionner automatiquement un champ

Vous pouvez placer automatiquement le curseur dans l'un des champs de votre formulaire avec l'attribut `autofocus`. Dès que le visiteur chargera la page, le curseur se placera dans ce champ.

Par exemple, pour que le curseur soit dans le champ "Prénom" par défaut :

Code : HTML

```
<input type="text" name="prenom" id="prenom" autofocus />
```

Rendre un champ obligatoire

Vous pouvez faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut `required`.

Code : HTML

```
<input type="text" name="prenom" id="prenom" required />
```

Le navigateur indiquera alors au visiteur qu'il doit remplir le champ si celui-ci est vide au moment de l'envoi.

Les anciens navigateurs, qui ne reconnaissent pas cet attribut, enverront le formulaire sans vérification. Il sera nécessaire pour eux de compléter les tests avec des scripts Javascript par exemple.

On dispose de pseudo-formats en CSS pour styliser les éléments requis (`:required`) et invalides (`:invalid`). Pour ajouter un fond rouge aux éléments requis qui n'auraient pas été saisis, on écrira donc :

Code : CSS

```
:required
{
    background-color: red;
}
```

N'oubliez pas aussi que le pseudo-format `:focus` est disponible pour changer l'apparence d'un champ lorsque le curseur se trouve à l'intérieur.

Le bouton d'envoi

Il ne nous reste plus qu'à créer le bouton d'envoi. Là encore, la balise **`<input />`** vient à notre secours. Elle existe en 4 versions :

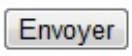
- `type="submit"` : le principal bouton d'envoi de formulaire. C'est celui que vous utiliserez le plus souvent. Le visiteur sera envoyé à la page indiquée à l'attribut `action` du formulaire.
- `type="reset"` : remise à zéro du formulaire.
- `type="image"` : équivalent du bouton "submit", présenté cette fois sous forme d'image. Rajoutez l'attribut `src` pour indiquer l'URL de l'image.
- `type="button"` : bouton générique, qui n'aura (par défaut) aucun effet. En général, ce bouton est géré en Javascript pour exécuter des actions sur la page. Nous ne l'utiliserons pas ici.

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

Pour créer un bouton d'envoi, on écrira donc par exemple :

Code : HTML

```
<input type="submit" value="Envoyer" />
```



Lorsque vous cliquez sur le bouton "Envoyer", le formulaire vous amène alors à la page indiquée à l'attribut `action`. Nous avions imaginé une page fictive (`traitement.php`), souvenez-vous.

Le problème, c'est que vous ne pouvez pas créer cette page seulement en HTML. Il est nécessaire d'apprendre un nouveau langage, comme le PHP, pour pouvoir "récupérer" les informations saisies et décider quoi en faire. Ca tombe bien, j'ai aussi rédigé un cours sur le langage PHP pour ceux que ça intéresse !

Certains sites comme swisstools.net proposent des services appelés "MailForm" qui se chargent de vous envoyer un email lorsqu'un de vos visiteurs a rempli le formulaire. Cela vous dispense d'apprendre un nouveau langage, mais ce n'est pas très pratique : vous devrez soit payer, soit voir de la publicité... et vous ne pouvez pas beaucoup personnaliser la façon dont les données sont traitées.

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 tomaaaaates !!! 

Il va être nécessaire d'apprendre un autre langage, comme le PHP, pour effectuer le traitement de vos formulaires. Un autre monde magique rempli de mots tordus d'informaticien vous attend ! 

La vidéo et l'audio

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

Cependant, aucune balise HTML ne permettait jusqu'ici de gérer la vidéo. Il fallait à la place utiliser un *plugin*, comme Flash. Encore aujourd'hui, Flash reste de loin le moyen le plus utilisé pour regarder des vidéos sur Youtube, Dailymotion, Vimeo et ailleurs. Mais utiliser un plugin a de nombreux défauts : on dépend de ceux qui gèrent le plugin (en l'occurence l'entreprise Adobe, qui possède Flash), on ne peut pas toujours contrôler son fonctionnement, il y a parfois des failles de sécurité... Au final, c'est assez lourd.

Imaginez, pour les images, si la balise `<img />` n'existait pas. Si à la place il fallait à chaque fois charger un programme (un plugin) pour afficher une image, ce serait lourd et compliqué !

C'est pour cela que 2 nouvelles balises standard ont été créées en HTML5 : `<video>` et `<audio>` !

Les formats audio et vidéo

Lorsque je vous avais présenté les images et la balise `<img />`, j'avais commencé par un petit tour d'horizon des différents formats d'image (JPEG, PNG, GIF...). Pour la vidéo et l'audio, je vais faire pareil... mais c'est plus compliqué.

En fait, le fonctionnement des vidéos est même tellement complexe qu'on pourrait faire un cours entier à ce sujet ! Etant donné qu'on parle ici de HTML, nous n'allons pas passer toutes nos prochaines nuits à étudier les subtilités de l'encodage vidéo. Je vais donc simplifier les choses et vous expliquer juste ce que vous avez besoin de savoir.

Les formats audio

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

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

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

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

Il n'y a pas de format "idéal" reconnu par tous les navigateurs ?

Eh non ! Heureusement on pourra proposer différents formats aux navigateurs, qui sélectionneront celui qu'ils savent lire.

Les formats vidéo

Le stockage de la vidéo est autrement plus complexe. On a besoin de 3 éléments :

- **Un format conteneur** : c'est un peu comme une boîte qui va servir à contenir les deux éléments ci-dessous. On les reconnaît à l'extension du fichier en général : AVI, MP4, MKV...
- **Un codec audio** : c'est le format du son de la vidéo, généralement compressé. Nous venons de les voir, on utilise les mêmes : MP3, AAC, OGG...
- **Un codec vidéo** : c'est le format qui va compresser les images. C'est là que les choses se corsent, car ces formats sont complexes et on ne peut pas toujours les utiliser gratuitement. Les principaux à connaître pour le Web sont :
  - **H.264** : l'un des plus puissants et des plus utilisés aujourd'hui... mais il n'est pas 100% gratuit. En fait, on peut l'utiliser gratuitement dans certains cas (comme la diffusion de vidéos sur son site web), mais il y a un flou juridique qui fait qu'il est risqué de l'utiliser à tout va.
  - **Ogg Theora** : un codec gratuit et libre de droits, mais moins puissant que H.264. Il est bien reconnu sous Linux, mais sous Windows il faut installer des programmes pour pouvoir le lire.
  - **WebM** : un autre codec gratuit et libre de droits, plus récent. Proposé par Google, c'est le concurrent le plus sérieux à H.264 à l'heure actuelle.

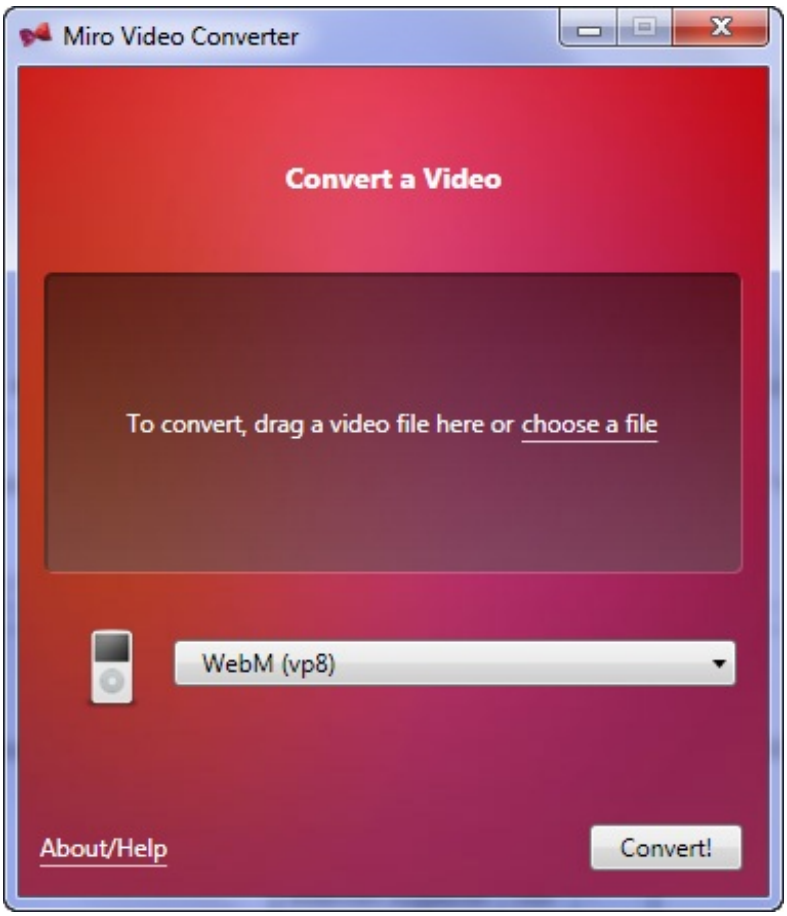
Quelle est la compatibilité des codecs vidéo sur les différents navigateurs ? Là encore, vous allez voir que c'est un joyeux bazar :

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

* WebM pour IE ne fonctionne que si le codec est installé sur l'ordinateur.

Là encore, aucun format ne sort du lot. Il est conseillé de proposer sa vidéo en plusieurs formats pour qu'elle soit lisible sur un maximum de navigateurs.

Pour convertir une vidéo dans ces différents formats, je vous conseille l'excellent logiciel gratuit Miro Video Converter :



Il vous suffit de faire glisser-déposer votre vidéo dans la fenêtre du programme et de sélectionner le format de sortie souhaité. Cela vous permettra de créer plusieurs versions de votre vidéo !

Insertion d'un élément audio

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

En théorie, il suffit d'une simple balise pour jouer un son sur notre page :

Code : HTML

```
<audio src="musique.mp3"></audio>
```

En pratique, c'est un peu plus compliqué que ça.
Si vous testez ce code... vous ne verrez rien ! En effet, le navigateur va seulement télécharger les informations générales sur le fichier (on parle de *métadonnées*), mais il ne se passera rien de particulier.

Vous pouvez compléter la balise des attributs suivants :

- `controls` : pour ajouter les boutons "Lecture", "Pause" et la barre de défilement. Ca peut sembler indispensable, et vous vous demandez peut-être pourquoi ça n'y est pas par défaut, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du Javascript.
- `width` : pour modifier la largeur de l'outil de lecture audio.
- `loop` : la musique sera jouée en boucle.
- `autoplay` : la musique sera jouée dès le chargement de la page. Evitez d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !
- `preload` : indique si la musique peut être préchargée dès le chargement de la page ou non. Elle peut prendre les valeurs :
  - `auto` (par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou

rien du tout.
- `metadata` : charge uniquement les métadonnées (durée, etc.).
- `none` : pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.

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

Ajoutons les contrôles et ça sera déjà mieux !

Code : HTML

```
<audio src="hype_home.mp3" controls></audio>
```

L'apparence du lecteur audio change en fonction du navigateur (ci-dessous l'apparence sous Google Chrome) :

Essayer !

Pourquoi ouvrir la balise pour la refermer immédiatement après ?

Cela vous permet d'afficher un message ou de proposer une solution de secours pour les navigateurs qui ne gèrent pas cette nouvelle balise. Par exemple :

Code : HTML

```
<audio src="hype_home.mp3" controls>Veuillez mettre votre navigateur
à jour !</audio>
```

Ceux qui ont un navigateur récent ne verront pas le message. Les anciens navigateurs, qui ne comprennent pas la balise, afficheront en revanche le texte qui se trouve à l'intérieur.

Je vous conseille de proposer une solution de secours en Flash, comme le Dewplayer. Vous placerez le code correspondant à Flash entre les balises **<audio>** et **</audio>** : ainsi, les anciens navigateurs afficheront le lecteur Flash, et les nouveaux afficheront le lecteur natif.

On a vu que certains navigateurs ne géraient pas le MP3, comment faire ?

Il faut proposer plusieurs versions du fichier audio. Dans ce cas, on va construire notre balise comme ceci :

Code : HTML

```
<audio controls>
    <source src="hype_home.mp3"></source>
    <source src="hype_home.ogg"></source>
</audio>
```

Le navigateur prendra automatiquement le format qu'il reconnaît.

Insertion d'une vidéo

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

Il suffit d'une simple balise **<video >** pour insérer une vidéo sur sa page :

Code : HTML

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

Mais, là encore, vous risquez d'être déçus si vous utilisez seulement ce code. Aucun contrôle ne permet de lancer la vidéo !

Rajoutons quelques attributs (la plupart sont les mêmes que pour la balise audio) :

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

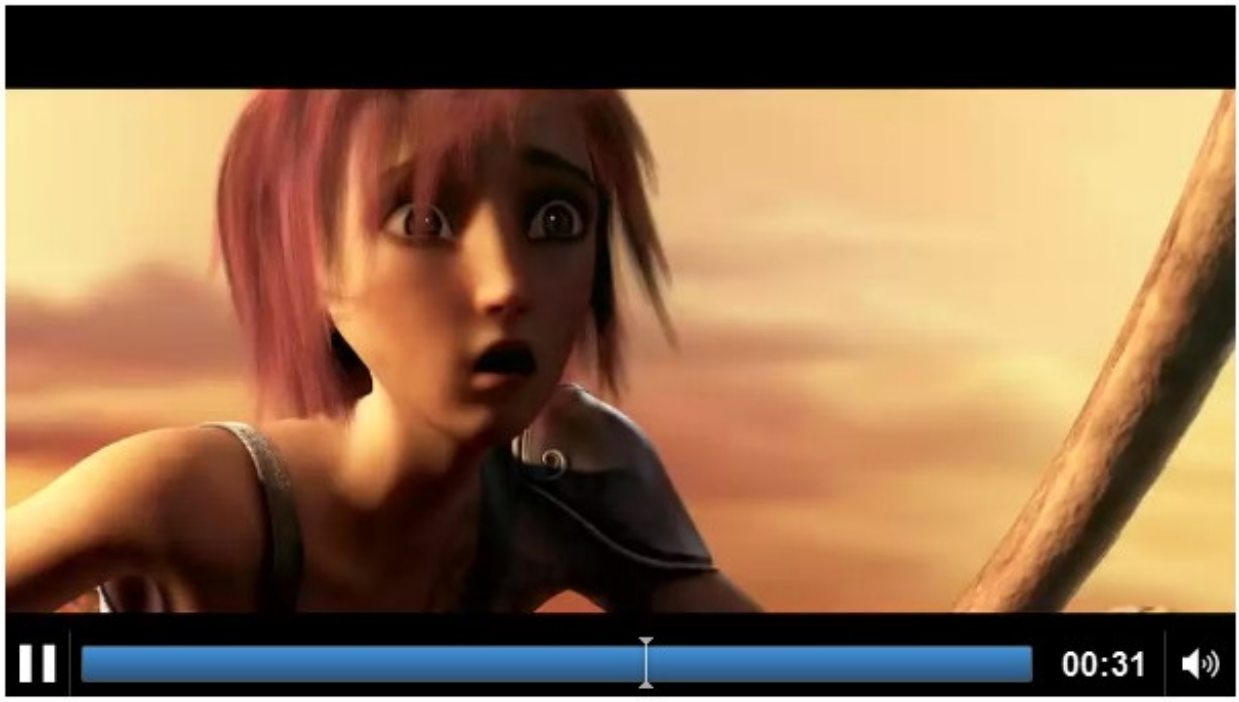
On ne peut pas forcer le préchargement de la vidéo, c'est toujours le navigateur qui décide.

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

Voici un code un peu plus complet :

Code : HTML

```
<video src="sintel.webm" controls poster="sintel.jpg"
width="600"></video>
```



Essayer !

Pourquoi ouvrir et refermer immédiatement après la balise ?

La réponse est la même que pour la balise **<audio>**. Cela vous permet d'afficher un message ou d'utiliser une technique de secours (en Flash) si le navigateur ne reconnaît pas la balise :

Code : HTML

```
<video src="sintel.webm" controls poster="sintel.jpg" width="600">
    Il est temps de mettre à jour votre navigateur !
</video>
```

Comment contenter tous les navigateurs, puisque chacun reconnaît des formats vidéo différents ?

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

Code : HTML

```
<video controls poster="sintel.jpg" width="600">
    <source src="sintel.mp4" />
    <source src="sintel.webm" />
    <source src="sintel.ogv" />
</video>
```

Les iPhone, iPad et iPod ne reconnaissent que le format H.264 à l'heure actuelle (fichier .mp4)... et uniquement si celui-ci est affiché en premier dans la liste ! Je vous recommande donc d'indiquer le format H.264 en premier pour assurer une compatibilité maximale.

Comment afficher la vidéo en plein écran ?

Ce n'est pas possible à l'heure actuelle. En fait, il y a bien un moyen sous Firefox mais il est un peu caché : il faut faire un clic droit sur la vidéo, puis sélectionner "Plein écran".

Il n'y a pas de moyen de forcer le plein écran, même en Javascript. Cela peut se comprendre, car des sites pourraient perturber fortement la navigation des visiteurs en affichant des vidéos en plein écran sans leur demander leur accord !

Comment protéger ma vidéo, je ne veux pas qu'on puisse la copier facilement !

Ce n'est pas possible. Les balises n'ont pas été conçues pour limiter ou empêcher le téléchargement. C'est assez logique quand on y pense : pour que le visiteur puisse voir la vidéo, il faut bien de toute façon qu'il la télécharge d'une manière ou d'une autre ! N'espérez donc pas empêcher le téléchargement de votre vidéo avec cette technique.

Notez que la balise `<audio>` est elle aussi concernée : le contenu peut toujours être récupéré.

Les lecteurs vidéo Flash permettent de "protéger" le contenu des vidéos, mais là encore des solutions de contournement existent. De nombreux plugins permettent de télécharger les vidéos de Youtube par exemple.

Comme vous avez pu le constater, on peut utiliser ces nouvelles balises `<video>` et `<audio>` dès aujourd'hui sur nos sites web... mais les choses sont encore assez compliquées à cause des différents formats, notamment pour la vidéo.

En attendant que ces nouvelles balises se démocratisent et que les navigateurs trouvent un terrain d'entente sur les formats, il reste conseillé de proposer une version Flash de votre vidéo. Mais un jour, fort heureusement, nous ne devrions plus à avoir à utiliser Flash pour nos vidéos.

Aller plus loin

Alors que ce tutoriel touche à sa fin, la tentation est grande de penser que l'on a tout vu.

Tout vu ? Vous n'avez quand même pas cru ça ? Allons bon, il vous reste des centaines de choses à découvrir, que ce soit sur HTML, CSS, ou les technologies qui y sont liées (PHP, Javascript...).

Ce chapitre a pour but de vous donner quelques directions pour compléter votre apprentissage. Alors ne soyez pas tristes, car vous n'avez pas fini de faire des découvertes !

Du site web à l'application web (Javascript, AJAX...)

Javascript est un langage qui existe depuis de nombreuses années maintenant et que l'on utilise fréquemment sur le web en plus de HTML et CSS. C'est probablement l'un des premiers langages que vous voudrez apprendre maintenant que vous avez des connaissances en HTML et CSS.

A quoi Javascript peut-il bien servir ? On ne peut pas tout faire avec HTML et CSS ?

On peut faire déjà beaucoup de choses en HTML et CSS, mais lorsqu'on veut rendre sa page plus interactive, un langage comme Javascript devient indispensable.

Voici quelques exemples de ce à quoi peut servir Javascript :

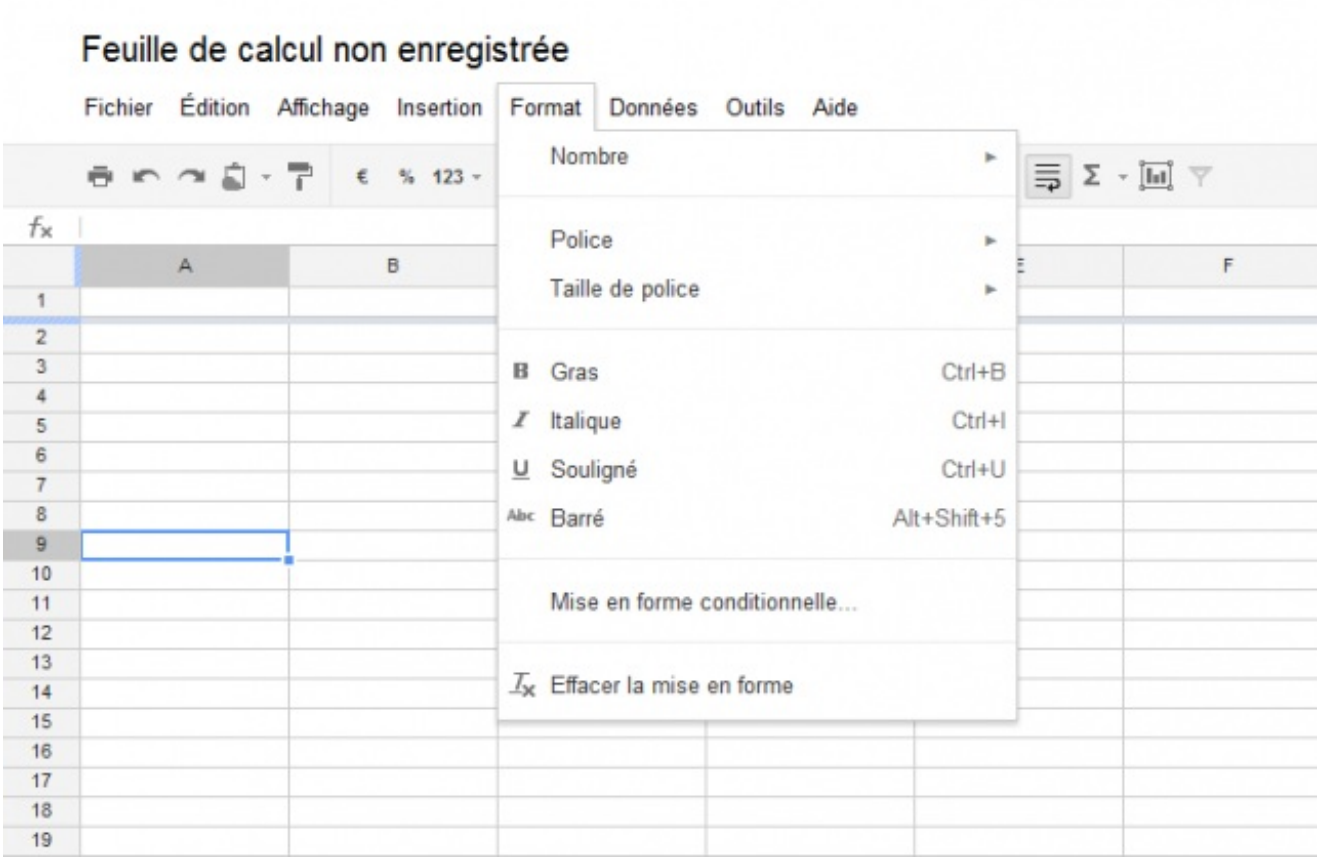
- On l'utilisera le plus souvent pour modifier des propriétés CSS sans avoir à recharger la page. Par exemple, vous pointez sur une image et le fond de votre site change de couleur (c'est pas possible à faire avec un `:hover` car ça concerne 2 balises différentes, c'est bien là une limite du CSS).
- On peut l'utiliser aussi pour modifier la source HTML sans avoir à recharger la page, *pendant* que le visiteur consulte la page.
- Il permet aussi d'afficher des boîtes de dialogue à l'écran du visiteur...
- ... ou encore de modifier la taille de la fenêtre.

Javascript est un langage qui se rapproche des langages de programmation tels que le C, C++, Python, Ruby... A l'inverse, HTML et CSS sont plus des langages de description : ils décrivent comment la page doit apparaître, mais ils ne donnent pas d'ordres directs à l'ordinateur ("fais ceci, fais cela...") contrairement à Javascript.

Javascript n'a *aucun* rapport avec le langage Java. Seuls les noms se ressemblent.

Javascript est régulièrement utilisé aujourd'hui pour faire de l'AJAX (Asynchronous Javascript and XML). Cette technique permet de modifier une partie de la page web que le visiteur consulte en échangeant des données avec le serveur. Cela donne l'impression que les pages sont plus dynamiques et plus réactives. Le visiteur n'a plus besoin de recharger toute la page systématiquement.

Les navigateurs sont de plus en plus efficaces dans leur traitement de Javascript aujourd'hui, ce qui fait que les pages qui utilisent Javascript sont de plus en plus réactives. On peut ainsi arriver aujourd'hui à créer des sites qui deviennent littéralement des applications web, l'équivalent des programmes disponibles sous forme de sites web !
Un exemple célèbre : Google Docs, la suite bureautique de Google disponible sur le Web.



Google Docs (ici le tableur) fait un usage intensif de Javascript

Pour en savoir plus sur Javascript, lisez le tutoriel du Site du Zéro !

Lire le tutoriel Javascript

Technologies liées à HTML5 (Canvas, SVG, Web Sockets...)

Le W3C ne travaille pas que sur les langages HTML et CSS. Ce sont certes les plus connus, mais le W3C cherche aussi à définir d'autres technologies qui viennent compléter HTML et CSS. Elles sont nombreuses et on les confond d'ailleurs souvent avec HTML5.

En fait, HTML5 est devenu un mot très utilisé qui fait référence à d'autres technos que HTML. Quand quelqu'un vous parle de "HTML5" aujourd'hui, il fait peut-être aussi référence à d'autres éléments que simplement le HTML.

Voici une petite liste de ces nouvelles technologies introduites en parallèle de HTML5 (notez que certaines ne sont pas vraiment "nouvelles" mais elles reviennent sur le devant de la scène) :

- **Canvas** : permet de dessiner au sein de la page web, à l'intérieur de la balise HTML `<canvas>`. On peut dessiner des formes (triangles, cercles...) mais aussi ajouter des images, les manipuler, appliquer des filtres graphiques... Au final, cela nous permet de réaliser aujourd'hui de véritables jeux et des applications graphiques directement dans des pages web ! Vous pouvez par exemple consulter ce tutoriel sur Canvas.
- **SVG** : permet de créer des dessins vectoriels au sein des pages web. A la différence de Canvas, ces dessins peuvent être agrandis à l'infini (c'est le principe du vectoriel). Le logiciel Inkscape est connu pour permettre de dessiner des SVG. Lire le tutoriel sur SVG sur le Site du Zéro.
- **Drag & Drop** : permet le glisser / déposer des objets dans la page web, de la même façon qu'on peut faire glisser / déposer des fichiers sur son bureau. Gmail l'utilise pour permettre d'ajouter facilement des pièces jointes à un email.
- **File API** : permet d'accéder aux fichiers stockés sur la machine du visiteur (avec son autorisation). On l'utilisera notamment en combinaison avec le Drag & Drop.
- **Géolocalisation** : pour localiser le visiteur et lui proposer des services liés au lieu où il se trouve (ex. : les horaires des salles de cinéma proches). La localisation n'est pas toujours très précise, mais cela peut permettre de repérer un visiteur à quelques kilomètres près (avec son accord).
- **Web Storage** : permet de stocker un grand nombre d'informations sur la machine du visiteur. C'est une alternative plus puissante que les traditionnels cookies. Les informations sont hiérarchisées, comme dans une base de données.
- **Appcache** : permet de demander au navigateur de mettre en cache certains fichiers, qu'il ne cherchera alors plus à

télécharger systématiquement. Très utile pour créer des applications web qui peuvent fonctionner même en mode "hors ligne" (déconnecté).
- **Web Sockets** : permet des échanges plus rapides en temps réel entre le navigateur du visiteur et le serveur qui gère le site web (c'est une sorte d'AJAX amélioré). C'est un peu l'avenir pour les applications web, qui pourront devenir aussi réactives que les vrais programmes.
- **WebGL** : permet d'introduire de la 3D dans les pages web, en utilisant le standard de la 3D OpenGL. Les scènes 3D sont directement gérées par la carte graphique.

La plupart de ces technologies s'utilisent avec Javascript. Il s'agit donc de nouvelles fonctionnalités que l'on peut utiliser en Javascript.

Démo de WebGL dans le navigateur web

Comme vous le voyez, vous avez de nouveaux mondes à découvrir ! Dès que vous connaîtrez suffisamment Javascript, vous pourrez aller encore plus loin dans la gestion de votre site web... que vous pourrez même transformer en véritable application !

Les sites web dynamiques (PHP, JEE, ASP .NET...)

Les langages dont nous allons parler ici sont eux aussi des langages de programmation. Comme Javascript ? Oui, mais avec une différence importante : Javascript s'exécute sur la machine de vos visiteurs, tandis que les langages que nous allons voir s'exécutent sur le "serveur" qui contient votre site web.

Quelle différence ça fait que le programme tourne sur la machine du visiteur ou sur le serveur ?

Les différences sont importantes. Tout d'abord, en terme de puissance, un serveur sera bien souvent plus rapide que la machine de vos visiteurs, ce qui permet d'effectuer des calculs plus complexes. Vous avez aussi plus de contrôle côté serveur qu'en Javascript... mais le Javascript reste irremplaçable car il y a certaines actions que vous ne pouvez faire que du côté "visiteur".

Les langages serveur permettent de générer la page web lorsque le visiteur arrive sur votre site. Chaque visiteur peut donc obtenir une page web personnalisée à ses besoins !

Les langages ne servent donc pas aux mêmes choses, mais ils se complètent. Si vous combinez HTML + CSS + Javascript + PHP par exemple, vous pouvez faire de l'AJAX (échanges de données entre la page et le serveur), vous pouvez effectuer des calculs, stocker des informations dans des bases de données... bref, faire de vrais sites web dynamiques !

Les langages "côté serveur" sont nombreux. Citons-en quelques-uns :

- **PHP** : l'un des plus connus. Facile à utiliser et puissant, il est utilisé notamment par Facebook... et le Site du Zéro. J'ai d'ailleurs rédigé un tutoriel sur PHP sur le Site du Zéro !
- **JEE** (Java) : très utilisé dans le monde professionnel, il s'agit d'une extension du langage Java qui permet de réaliser des sites web dynamiques, puissants et robustes. Il est un peu plus complexe à prendre en main au début que PHP.
- **ASP .NET** (C#) : assez semblable à JEE, c'est le langage de Microsoft. On l'utilise en combinaison avec d'autres technologies Microsoft (Windows Server...). Il utilise le puissant framework .NET, véritable couteau suisse des développeurs qui offre de nombreuses fonctionnalités.
- **Django** (Python) : une extension du langage Python qui permet de réaliser rapidement et facilement des sites web dynamiques. Il est connu pour générer des interfaces d'administration prêtes à l'emploi.
- **Ruby on Rails** (Ruby) : une extension du langage Ruby, assez similaire à Django, qui permet de réaliser des sites web dynamiques facilement avec une grande souplesse.

Connaître l'un de ces langages est indispensable si vous voulez traiter le résultat des formulaires HTML ! Souvenez-vous de la balise `<form>` : je vous avais expliqué comment créer des formulaires, mais pas comment "récupérer" les informations saisies par vos visiteurs. Il vous faut obligatoirement un langage serveur, comme PHP, pour récupérer et traiter ces données !

Au final, ces langages vous permettent de réaliser vos rêves les plus fous sur votre site web :

- Forums
- Newsletter
- Compteur de visiteurs
- Système de news automatisé
- Système de membres
- Jeux web (jeux de stratégie, élevage d'animaux virtuels...)
- etc.

Il est indispensable de connaître les langages HTML et CSS avant d'apprendre un langage serveur comme PHP !

Bonne découverte !

Lire le tutoriel PHP

Moussaillons, vous avez de nouvelles contrées fascinantes à explorer et de nouveaux langages à dompter !
Bon vent !

Partie 5 : Annexes

Les annexes contiennent d'autres informations qui vous seront utiles lors de la création de votre site web, comme des mémentos (résumés), 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 adaptés 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 **PlanetHoster** qui propose des services d'hébergement de qualité. Ils font d'ailleurs **des réductions pour tous les visiteurs du Site du Zéro** !

Si les offres de PlanetHoster ne vous conviennent pas, vous pouvez regarder chez l'hébergeur **1&1** (un concurrent) ou encore MavenHosting, qui proposent d'autres offres pour les particuliers et entreprises.

La suite de ce chapitre vous détaille la procédure pour héberger votre site chez PlanetHoster, mais sachez que cela fonctionne quasiment de la même manière avec 1&1 et MavenHosting.

Revenons à PlanetHoster. L'hébergeur propose plusieurs offres d'hébergement mutualisé :

PlanetHoster fait des **réductions** spéciales pour les visiteurs du Site du Zéro sur tous ces plans via un code promotionnel :

- 5% de remise pour le plan essentiel
- 15% de remise pour les plans performance et illimité

Ces remises sont valables si vous rentrez un code promotionnel (j'en reparle un peu plus bas) pour une commande annuelle de l'un de ces plans.

- **Plan essentiel** : 10 Go d'espace disque et 250 Go de trafic.
- **Plan performance** : 50 Go d'espace disque et trafic illimité.
- **Plan illimité** : espace disque et trafic illimités.

Ces offres sont en fait très similaires, elles diffèrent seulement au niveau de l'espace de stockage et du trafic.

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 créerez 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.

Si vous avez beaucoup de visiteurs (donc beaucoup de trafic), il faudra choisir un plan qui autorise plus de trafic.

Commander un hébergement pour votre site web

Après avoir cliqué sur n'importe quel bouton "Commander", nous arrivons sur la page suivante :

Nous avons ici trois informations :

- **Choisissez un produit** : indique quel plan vous avez choisi. Vous pouvez en changer directement via cet encart ;
- **Résumé** : ce cadre, comme son nom l'indique, résume votre commande avec le plan choisi ainsi que le prix à payer ;
- **Nom de domaine** : cette partie vous permet de choisir le nom de domaine de votre site web. Nous allons y venir.

Le champ de texte se trouvant dans le cadre **Résumé** vous permet de rentrer un code promotionnel :

- **SiteDuZero-Perso** : si vous commandez un plan essentiel (à 2,99€ / mois)
- **SiteDuZero** : si vous commandez n'importe quel autre plan

Par exemple pour le code SiteDuZero-Perso :

Validez, et vous aurez alors une réduction sur le montant total du pack d'hébergement !

Merci qui ?

Le champ de texte se trouvant sous **Nom de domaine** vous invite à saisir… votre nom de domaine. Le site de PlanetHoster va alors se charger de vérifier 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.

Ensuite, le site vous demande si vous désirez qu'il enregistre ce domaine ou si vous désirez l'enregistrer séparément.

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 Planethoster, le nom de domaine est offert (il est compris dans l'hébergement).

Il ne vous reste plus qu'à renseigner vos coordonnées et finaliser l'achat par carte bancaire ou Paypal.
Une fois les formalités et le paiement effectués, vous êtes redirigé vers PlanetHoster, 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**.

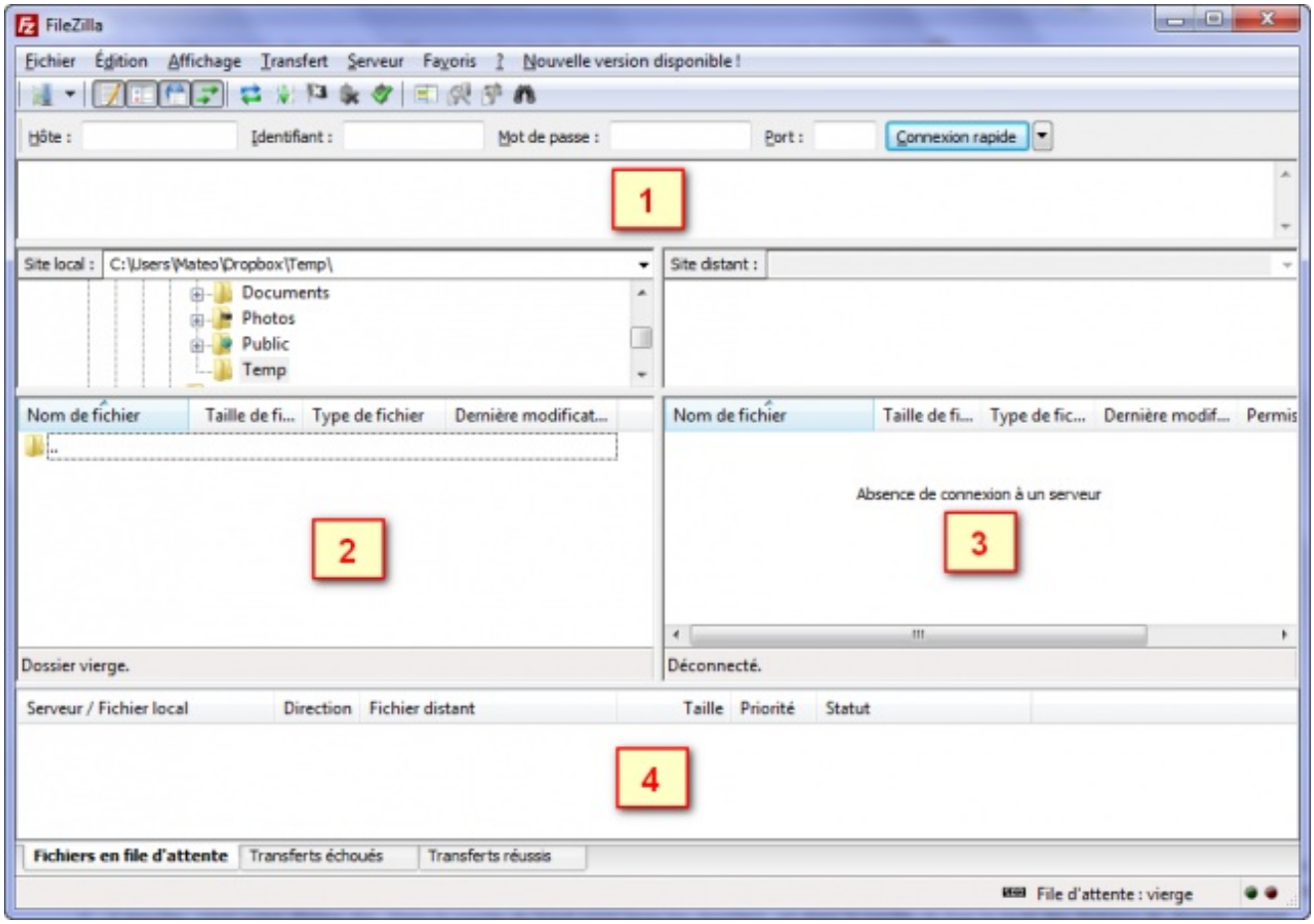
Ce logiciel n'a rien à avoir avec Mozilla, si ce n'est qu'il se termine lui aussi par "zilla". N'allez donc pas croire que je vous force à utiliser des logiciels d'un même éditeur, c'est tout à fait faux
D'ailleurs, vous pouvez utiliser n'importe quel autre logiciel FTP si ça vous chante, ça ne me dérange 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

Prenez la version correspondant à votre système d'exploitation (Windows, Mac OS X ou Linux).

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 en rouge...
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

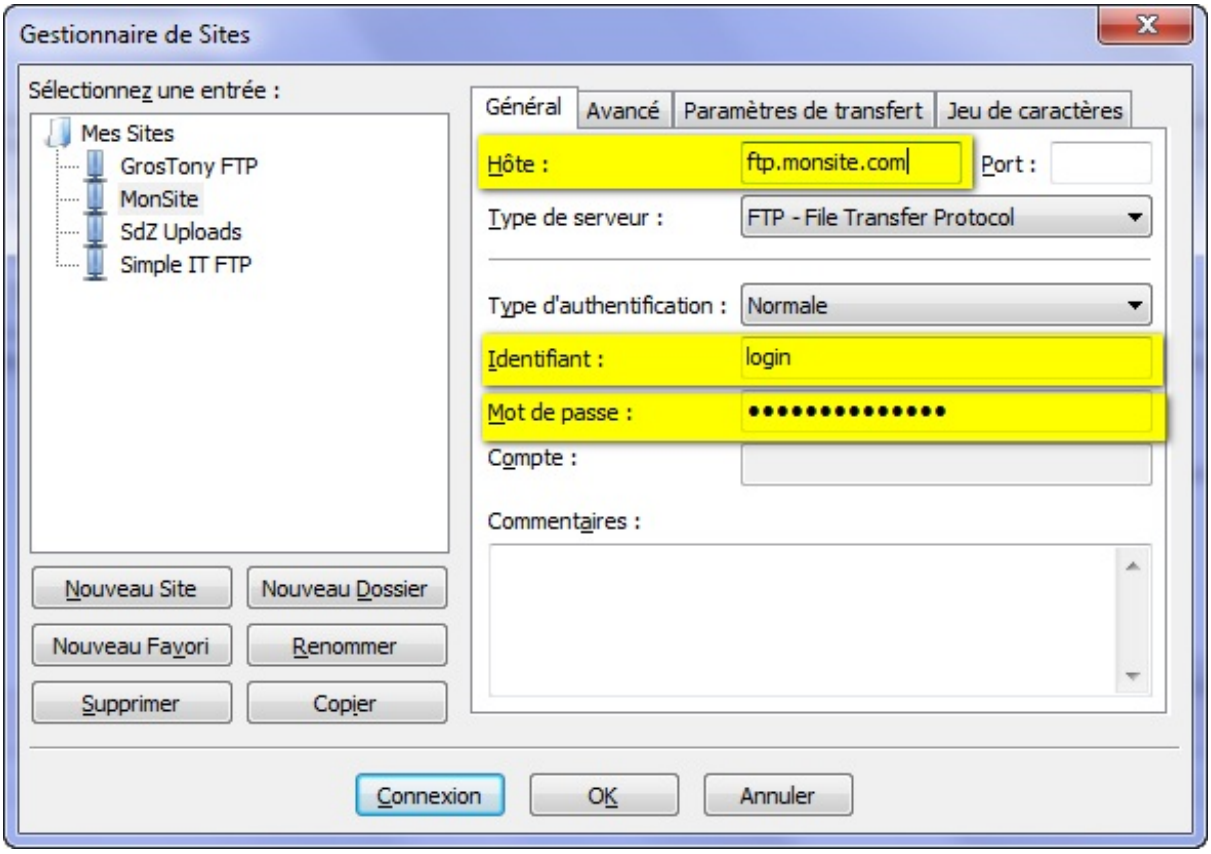
truc imprononçable du genre *crf45u7h*)

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 par exemple *ftp.monsite.com*).
Pour pouvoir entrer le login / mot de passe, cochez **Type d'authentification : Normal**

Cliquez sur "Connexion" et le tour est (presque) joué.

Transférer les fichiers

A ce stade, 2 possibilités :

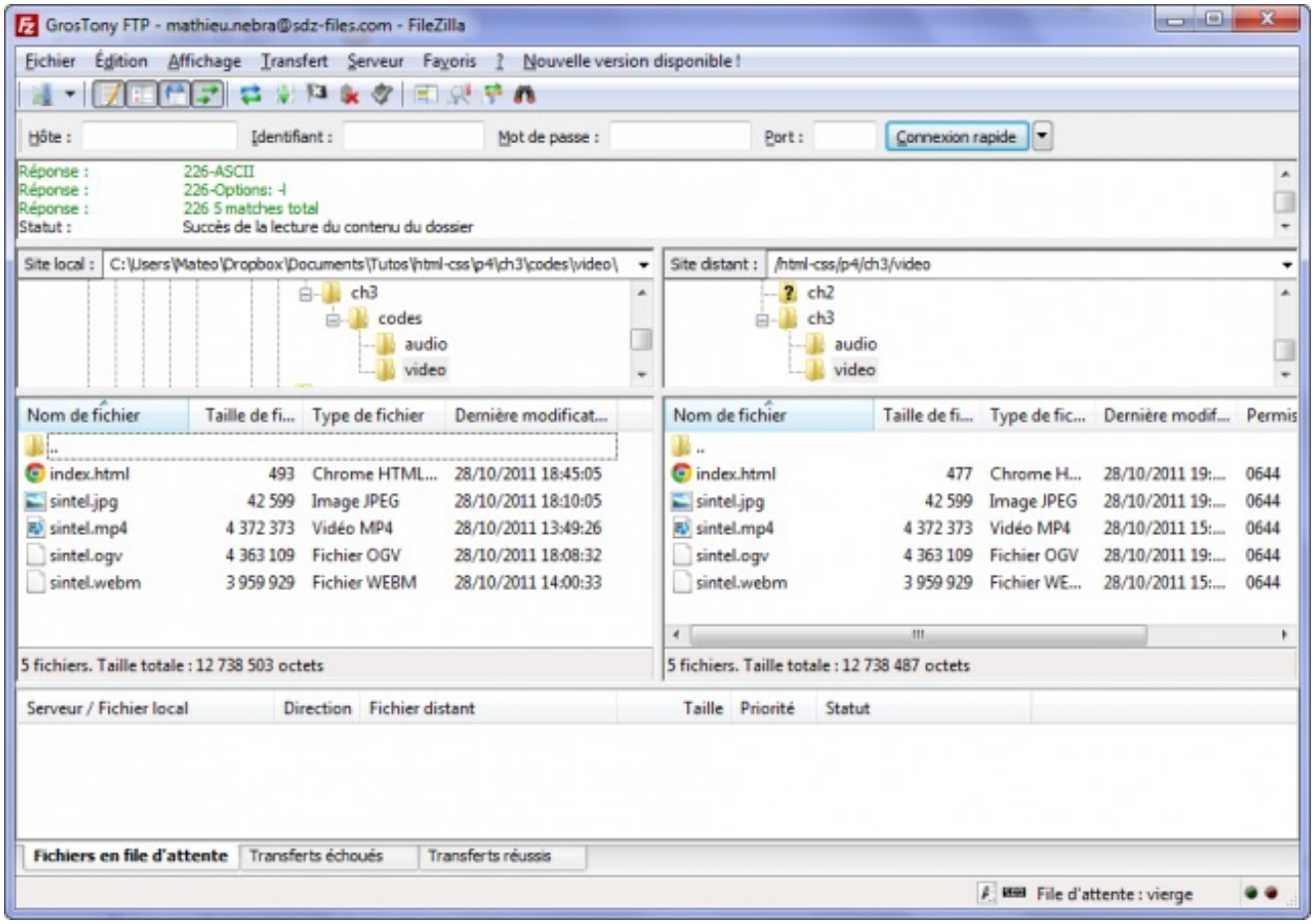
- Soit la connexion a réussi, et vous voyez apparaître en haut des messages en vert comme "Connecté". Dans ce cas, la zone de droite de la fenêtre de FileZilla devrait s'activer et vous verrez les fichiers qui se trouvent déjà sur le serveur (il se peut qu'il y en ait quelques-uns déjà présents).
- Soit 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, veillez 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 .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 après avoir transféré un fichier `index.html` et quelques autres fichiers :



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.

Une fois configuré, vous pouvez voir que l'envoi de fichiers est très simple.
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

Mémento des balises HTML

Cette page est une liste *non exhaustive* des balises HTML qui existent.

Vous trouverez ici un grand nombre de balises HTML. Nous en avons déjà vu certaines dans le cours, mais il y en a d'autres qu'on n'a pas eu l'occasion d'étudier. Généralement, les balises qu'on n'a pas étudiées sont des balises un peu plus rarement utilisées. Peut-être trouverez-vous votre bonheur dans ce lot de nouvelles balises.

Vous pouvez vous servir de cette page comme d'un aide-mémoire 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 HTML. Elles sont indispensables pour réaliser le "code minimal" d'une page web.

Balises	Description
<html>	Balise principale de toute page web. Doit englober tout le code de votre page web.
<head>	En-tête de la page
<body>	Corps de la page

Le code minimal d'une page HTML

Vous trouverez ci-dessous le code minimal d'une page HTML.

Code : HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </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
	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 : **Code : HTML** `<link rel="stylesheet" href="style.css" />` On peut aussi s'en servir pour 2-3 autres choses :

<link />

Code : HTML

```
<!-- Page d'accueil du site -->
<link rel="start" title="Accueil" href="index.html" />
<!-- Page d'aide du site -->
<link rel="help" title="Politique d'accessibilité"
href="accessibilite.html" />
<!-- Fil RSS du site -->
<link rel="alternate" type="application/rss+xml" title="News
de mon site" href="news.xml" />
<!-- Icône du site (favicon) -->
<link rel="shortcut icon" type="image/x-icon"
href="favicon.ico" />
```

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 HTML / CSS ça ne vous intéresse donc pas pour le moment).

<meta />

Cette balise permet de définir les propriétés de la page web.
On s'en sert pour une foule de choses. Voici quelques exemples pratiques :

Code : HTML

```
<!-- Table de caractères -->
<meta charset="utf-8" />
<!-- Auteur de la page -->
<meta name="author" content="Jean Dupont" />
<!-- Description de la page -->
<meta name="description" content="La page personnelle de
Jean Dupont" />
<!-- Mots-clés de la page -->
<meta name="keywords" content="expériences, recherche,
laboratoire, chimie" />
<!-- Adresse de contact -->
<meta name="reply-to" content="monadresse@email.com" />
<!-- Empêcher la mise en cache de la page par le navigateur
-->
<meta http-equiv="pragma" content="no-cache" />
<!-- Rafraîchissement automatique au bout de 10 secondes -->
<meta http-equiv="refresh" content="10;
URL=http://www.monsite.com" />
```

En général, on utilise surtout le meta pour la table de caractères.

<script>

Permet de placer un script.
On l'utilise souvent pour mettre du code Javascript :

Code : HTML

```
<script>
/* Votre script ici */
</script>
```

Permet de définir du code CSS pour la page.

<style>	Exemple :
	Code : HTML
	`<style>`
	`/* Votre code CSS ici */`
	`</style>`

<title>	Titre de la page web. 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). **Code : HTML** `<title>Les petites expériences chimiques de M. Dupont</title>`

Balises de structuration du texte

Balise	Type	Description
<abbr>	*Inline*	Abréviation. Utiliser l'attribut `title` pour indiquer sa signification.

<blockquote>	*Block*	Citation (longue)
		Vous devez obligatoirement mettre une balise de paragraphe à l'intérieur du blockquote. Par exemple :
		
		Code : HTML
		`<blockquote>`
		`  <p>`
		`  Texte de la citation`
		`  </p>`
		`</blockquote>`

<cite>	*Inline*	Citation du titre d'une oeuvre ou d'un évènement.
<q>	*Inline*	Citation (courte)
<sup>	*Inline*	Mise en exposant
<sub>	*Inline*	Mise en indice
	Inline	Mise en valeur (forte) Le texte est généralement mis en gras.
<mark>	*Inline*	Mise en valeur visuelle. Le texte est généralement surligné.
	Inline	Mise en valeur (faible) Le texte est généralement mis en italique.
<h6>	*Block*	Titre de niveau 6
<h5>	*Block*	Titre de niveau 5
<h4>	*Block*	Titre de niveau 4
<h3>	*Block*	Titre de niveau 3
<h2>	*Block*	Titre de niveau 2
<h1>	*Block*	Titre de niveau 1

<img />	*Inline*	Insère une image. Utilisez les attributs *src* (pour indiquer l'adresse de l'image) et *alt* (pour indiquer un texte de remplacement). Ces 2 attributs sont obligatoires. Exemple : **Code : HTML** `<img src="Templates/images/smiley.png" alt=":)" />`
<figure>	*Block*	Indique la présence d'une figure (image, code...) illustrant le texte.
<figcaption>	*Inline*	Description de la figure.
<audio>	*Inline*	Insère un son.
<video>	*Inline*	Insère une vidéo.
<source>	-	Indique un format possible pour les balises <audio> et <video>.
<a>	*Inline*	Lien hypertexte. Indiquez l'url de destination grâce à l'attribut *href* : **Code : HTML** `<a href="autrepage.html">Rendez-vous sur l'autre page</a>`
 	Inline	Retour à la ligne
<p>	*Block*	Paragraphe
<hr />	*Block*	Crée une ligne de séparation horizontale
<address>	*Block*	Permet d'indiquer une adresse, ou éventuellement l'auteur d'un document. Le texte est généralement mis en italique.
<del>	*Inline*	Permet d'indiquer un texte qui a été supprimé. Le texte est généralement barré.
<ins>	*Inline*	Permet d'indiquer un texte qui a été inséré. Le texte est généralement souligné.
<dfn>	*Inline*	Permet d'indiquer une définition.
<kbd>	*Inline*	Permet d'indiquer un code que doit taper le visiteur.
<pre>	*Block*	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.
<progress>	*Inline*	Affiche une barre de progression. A utiliser conjointement avec les attributs `value` et `max`.
<time>	*Inline*	Pour indiquer une date ou une heure.

Balises de liste

Cette partie énumère toutes les balises HTML permettant de créer des listes (listes à puces, listes numérotées, listes de définitions...)

Balise	Type	Description

	Block	Liste à puces non numérotée. Vous devez mettre un par élément de la liste. Exemple :
		Code : HTML
		`<ul>`
		`<li>Un élément</li>`

```
<li>Un autre élément</li>
</ul>
```

	Block	Liste à puces numérotée. Vous devez mettre un par élément de la liste. Exemple : **Code : HTML** `<ol><li>Elément n°1</li><li>Elément n°2</li></ol>`
	list-item	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 *list-item*.
<dl>	*Block*	Liste de définitions. Vous devez alterner chaque terme <dt> par sa définition <dd>. Exemple : **Code : HTML** `<dl><dt>Porte</dt><dd>Ouverture dans un mur permettant d'entrer et de sortir</dd><dt>Théâtre</dt><dd>Lieu où l'on représente des ouvrages dramatiques</dd></dl>`
<dt>	*Block*	Terme à définir
<dd>	*Block*	Définition du terme

Balises de tableau

Balise	Type	Description
<table>	*Block*	Délimite un tableau. Voici un exemple de tableau simple : **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>
   <tr>
       <td>François</td>
```

```
            <td>43 ans</td>
            <td>France</td>
        </tr>
    </table>
```

<caption>	-	Permet de donner un titre au tableau
<tr>	-	Ligne de tableau
<th>	-	Cellule d'en-tête du tableau (généralement mise en gras)
<td>	-	Cellule du tableau
<thead>	-	Balise non obligatoire permettant d'insérer l'en-tête du tableau. Si vous choisissez d'utiliser <thead>, <tfoot> et <tbody>, vous devez les mettre dans l'ordre suivant dans votre code source : 1. <thead> 2. <tfoot> 3. <tbody>
<tbody>	-	Balise non obligatoire permettant d'insérer le corps du tableau
<tfoot>	-	Balise non obligatoire permettant d'insérer le pied du tableau

Balises de formulaire

Balise	Type	Description
<form>	*Block*	Délimite un formulaire. Vous devrez généralement donner 2 attributs à la balise <form> • **method** : indique la méthode d'envoi du formulaire (get ou post). Si vous ne savez pas quoi utiliser, mettez post. • **action** : la page vers laquelle le visiteur doit être redirigé lorsqu'il a validé votre formulaire.
<fieldset>	*Block*	Permet de regrouper plusieurs éléments d'un formulaire. On l'utilise généralement dans de grands formulaires. Pour donner un titre à votre groupe, utilisez la balise <legend>
<legend>	*Inline*	Titre d'un groupe dans un formulaire. A utiliser à l'intérieur d'un <fieldset>
<label>	*Inline*	Titre d'un élément de formulaire. Généralement, vous devrez mettre l'attribut *for* sur cette balise pour indiquer l'ID de l'élément auquel correspond le label.

<input />	*Inline*	Champ de formulaire.
		Il existe de nombreux types de champs différents. Vous choisissez le type de champ que vous désirez grâce à l'attribut *type* :
		
		Code : HTML
		
		`<!-- Zone de texte d'une ligne -->`
		`<input type="text" />`
		`<!-- Mot de passe (le texte est caché) -->`
		`<input type="password" />`
		`<!-- Envoi de fichier -->`
		`<input type="file" />`
		`<!-- Case à cocher -->`
		`<input type="checkbox" />`
		`<!-- Bouton d'option -->`
		`<input type="radio" />`
		`<!-- Bouton -->`
		`<input type="button" />`

		<!-- Bouton d'envoi --> <input type="submit" /> <!-- Bouton de remise à zéro --> <input type="reset" /> <!-- Champ caché --> <input type="hidden" /> Pensez à donner un nom à vos champs grâce à l'attribut *name*
<textarea>	*Inline*	Zone de saisie multiligne. Vous pouvez définir sa taille grâce aux attributs *rows* et *cols* (nombre de lignes et colonnes) ou bien le faire en CSS grâce aux propriétés *width* et *height*.
<select>	*Inline*	Liste déroulante. Utilisez la balise <option> pour créer chaque élément de la liste : **Code : HTML** <select name="pays"> <option value="france">France</option> <option value="espagne">Espagne</option> <option value="italie">Italie</option> </select>
<option>	*Block*	Element d'une liste déroulante
<optgroup>	*Block*	Groupe d'éléments d'une liste déroulante. A utiliser dans le cas d'une grande liste déroulante. Vous devez utiliser l'attribut *label* pour donner un nom au groupe.

Balises sectionnantes

Ces balises permettent de construire le squelette de notre site web :

Balise	Type	Description
<header>	*Block*	En-tête
<nav>	*Block*	Liens principaux de navigation
<footer>	*Block*	Pied de page
<section>	*Block*	Section de page
<article>	*Block*	Contenu ayant un sens propre (billet de blog, actualité...) pouvant être repris sur un autre site.
<aside>	*Block*	Informations complémentaires

Balises génériques

Les balises génériques sont des balises qui n'ont pas de sens sémantique.

En effet, toutes les autres balises HTML 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
	Inline	Balise générique de type inline
	Block	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 HTML 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 et (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 notamment plus de chances d'être mieux indexée dans Google qu'une page utilisant des balises inadaptées.

Souvenez-vous en !

Mémento des propriétés CSS

Cette page est une liste non exhaustive des propriétés CSS qui existent en CSS3.

Pour la plupart, ce sont des propriétés que nous avons vues dans le cours, mais vous trouverez aussi quelques nouvelles propriétés que nous n'avons pas abordées.

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 vraiment trop (plus de 200 !) et certaines sont très rarement utilisées.

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

<table>
<tr><th>Type</th><th>Nom</th><th>Valeurs possibles</th></tr>
<tr><td>Nom de police</td><td>font-family</td><td>Indiquer les noms de polices possibles par ordre de préférence :
Code : CSS
<pre>font-family: police1, police2, police3;</pre>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".
Code : CSS
<pre>font-family: "Arial Black", Arial, Verdana,
serif;</pre>Il est possible d'utiliser des polices personnalisées en combinaison avec @font-face.</td></tr>
<tr><td>Police personnalisée</td><td>@font-face</td><td>Permet de déclarer une nouvelle police, qui sera téléchargée sur l'ordinateur de vos visiteurs.
Code : CSS
<pre>@font-face {
    font-family: 'MaSuperPolice';
    src: url('MaSuperPolice.eot') format('eot'),
         url('MaSuperPolice.woff')
format('woff'),
         url('MaSuperPolice.ttf')
format('truetype'),
         url('MaSuperPolice.svg') format('svg');
}</pre></td></tr>
<tr><td></td><td></td><td>Indiquez la taille du texte.
Plusieurs unités sont possibles :
px (pixels)</td></tr>
</table>

Taille du texte	font-size	• % (pourcentage, 100% = normal)
		• em (taille relative, 1.0 = normal)
		• ex (taille relative à la hauteur de la lettre "x". 1.0 = normal)
		• nom de taille :
		  ◦ xx-small : très très petit
		  ◦ x-small : très petit
		  ◦ small : petit
		  ◦ medium : moyen
		  ◦ large : grand
		  ◦ x-large : très grand
		  ◦ xx-large : très très grand

Gras	font-weight	**bold** : gras **bolder** : plus gras **lighter** : plus fin **normal** : pas gras (par défaut)
Italique	font-style	**italic** : italique **oblique** : autre façon de mettre en italique **normal** : normal (par défaut)
Décoration	text-decoration	**underline** : souligné **overline** : ligne au-dessus **line-through** : barré **blink** : clignotant **none** : normal (par défaut)
Petites capitales	font-variant	**small-caps** : petites capitales **normal** : normal (par défaut)
Capitales	text-transform	**uppercase** : tout mettre en majuscules **lowercase** : tout mettre en minuscules **capitalize** : début des mots en majuscules **none** : normal (par défaut)
Super-propriété de police	font	Indiquez dans n'importe quel ordre des valeurs possibles pour *font-weight, font-style, font-size, font-variant, font-family*. **Attention exception** : le nom de la police (*font-family*) doit être placé en dernier dans la liste dans tous les cas. Vous n'êtes pas obligés de mettre une valeur de chacune de ces propriétés. Exemple : **Code : CSS** `font: bold 16px Arial;` Cela mettra votre texte en gras, 16 pixels, Arial.

Alignement

Type	Propriété	Valeurs possibles
Alignement horizontal	text-align	**left** : à gauche (par défaut) **center** : centré **right** : à droite **justify** : texte justifié (prend toute la largeur de la page)
Alignement	vertical	A utiliser dans des cellules de tableau, ou dans des éléments inline-block.

Alignement vertical	vertical-align	**top** : en haut **middle** : au milieu **bottom** : 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	**normal** : le passage à la ligne est automatique (par défaut) **nowrap** : pas de passage à la ligne automatique, à moins qu'une balise HTML comme ne soit présente. **pre** : le passage à la ligne se fait tel que le texte a été saisi dans le code source (comme la balise <pre>)
Césure forcée	word-wrap	Avec la valeur `break-word`, le texte sera coupé s'il dépasse du cadre.
Ombre de texte	text-shadow	Décalage horizontal, décalage vertical, adoucissement, couleur **Code : CSS** `text-shadow: 2px 2px 4px black;`

Propriétés de couleur et de fond

Couleur

Type	Propriété	Valeurs possibles
Couleur de texte	color	Indiquer une couleur avec l'une des méthodes suivantes : • En tapant le nom de la couleur en anglais (black, blue, green, white, red...). • En indiquant la couleur en hexadécimal (#CC48A1) • En indiquant la couleur en RGB : rgb (128, 255, 0)
Couleur de fond	background-color	Même fonctionnement que *color*. 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)
		Code : CSS
		`background-image:url("images/fond.png"); /* Notation relative */`
		`background-image:url("http://www.monsite.com/images/fond.png"); /* Notation absolue */`
		Il est possible de combiner plusieurs images de fond, en séparant les déclarations par des virgules.

Fond fixé	background-attachment	**fixed** : le fond reste fixe quand on descend plus bas sur la page **scroll** : le fond défile avec le texte (par défaut)
Répétition du	background-	**repeat** : le fond se répète (par défaut) **repeat-x** : le fond ne se répète que sur une ligne, horizontalement

fond	repeat	**repeat-y** : le fond ne se répète que sur une colonne, verticalement **no-repeat** : le fond ne se répète pas, il n'est affiché qu'une fois

Position du fond	background-position	2 façons de faire :
		• En notant une distance en px ou %, par rapport au coin en haut à gauche.
		Code : CSS
		`background-position:50px 200px; /* 50 px à droite, 200px en bas */`
		• En utilisant des valeurs prédéfinies, une pour la verticale et une pour l'horizontale :
		top : en haut, verticalement
		center : au milieu, verticalement
		bottom : en bas, verticalement
		
		left : à gauche, horizontalement
		center : au centre, horizontalement
		right : à droite, horizontalement
		Code : CSS
		`background-position : bottom right; /* en bas à droite */`

Super-propriété de fond	background	Indiquer une ou plusieurs valeurs issues des propriétés *background-image, background-repeat, background-attachment, background-position*.
		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)
		Code : CSS
		`/* 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;`

Transparence	opacity	Valeur entre 0 (transparence totale) et 1 (opacité totale). Exemple pour une transparence de 50% : **Code : CSS** `opacity:0.5;`

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
Super-propriété de marge	margin	Indiquez de 1 à 4 valeurs à la suite. Selon le nombre de valeurs que vous mettez, la signification change : • 1 valeur : ce sera la marge pour le haut, le bas, la gauche et la droite • 2 valeurs : la première correspond à la marge pour le haut et le bas, la seconde pour la gauche et la droite • 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 • 4 valeurs : respectivement la marge du haut, de la droite, du bas, de la gauche. Par exemple, si je mets 2 valeurs : **Code : CSS** `margin:20px 5px; /* 20px de marge en haut et en bas, 5px à gauche et à droite */`

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
Super-propriété de	padding	Indiquez de 1 à 4 valeurs à la suite. Selon le nombre de valeurs que vous mettez, la signification change : • 1 valeur : ce sera la marge pour le haut, le bas, la gauche et la droite • 2 valeurs : la première correspond à la marge pour le haut et le bas, la seconde

marge intérieure	padding	pour la gauche et la droite • 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 • 4 valeurs : respectivement la marge du haut, de la droite, du bas, de la gauche.

Bordures

Type	Propriété	Valeurs possibles
Epaisseur 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	**none** : pas de bordure (par défaut) **hidden** : bordure cachée **solid** : ligne pleine **double** : ligne double (nécessite une taille de bordure de 3px minimum) **dashed** : en tirets **dotted** : en pointillés **inset** : effet 3D "enfoncé" **outset** : effet 3D "surélevé" **ridge** : autre effet 3D
Bordure à gauche	border-left	Indiquer la couleur, l'épaisseur et le type de bordure pour la bordure gauche. L'ordre n'a pas d'importance. Exemple : **Code : CSS** `border-left: 2px inset blue; /* Bordure bleue de 2px avec effet 3D "enfoncé" */`
Bordure en haut	border-top	Idem
Bordure à droite	border-right	Idem
Bordure en bas	border-bottom	Idem
Super-propriété de bordure	border	Indiquera l'apparence des bordures en haut, à droite, en bas et à gauche.
Bordure arrondie	border-radius	Indiquer une valeur en px, ou 4 valeurs pour chacun des coins, en partant de celui en haut à gauche.
Ombre	box-shadow	Décalage horizontal, décalage vertical, adoucissement, couleur **Code : CSS** `box-shadow: 2px 2px 4px black;`

Propriétés de positionnement et d'affichage

Affichage

Type	Propriété	Valeurs possibles
Type d'élément	display	**none** : l'élément ne sera pas affiché **block** : l'élément devient de type "block" (bloc, comme <p>) **inline** : l'élément devient de type "inline" (en ligne, comme) **inline-block** : l'élément est affiché comme un inline mais peut être redimensionné comme un block. **list-item** : l'élément devient de type "élément de liste à puce" (comme)
Affichage	visibility	**hidden** : masqué **visible** : visible (par défaut)  display:none; fait complètement disparaître l'élément, tandis que visibility:hidden; 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 : **Code : CSS** `clip: rect(valeur1, valeur2, valeur3, valeur4);` 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	**visible** : tout l'élément sera affiché (par défaut). **hidden** : l'élément sera coupé s'il dépasse les limites définies par height et width. On ne pourra pas voir la partie du texte coupée. **scroll** : 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. **auto** : 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	**left** : flottant à gauche **right** : flottant à droite **none** : pas de flottant (par défaut)
Stopper un flottant	clear	**left** : supprime l'effet d'un flottant à gauche précédent **right** : supprime l'effet d'un flottant à droite précédent **both** : supprime l'effet d'un flottant précédent, qu'il soit à gauche ou à droite **none** : pas de suppression de l'effet du flottant (par défaut)
Type de positionnement	position	**absolute** : position absolue par rapport au coin en haut à gauche **fixed** : position fixe (fonctionne comme la position absolue). L'élément reste à sa position même quand on descend plus bas dans la page. **relative** : position relative, par rapport à la position "normale" de l'élément **static** : 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		

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	• Pour les listes non ordonnées ( ) : ◦ disc : un disque noir. ◦ circle : un cercle (par défaut). ◦ square : un carré. ◦ none : aucune puce ne sera utilisée. • Pour les listes ordonnées ( ) : ◦ 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.
Position en retrait	list-style-position	**inside** : sans retrait **outside** : avec retrait (par défaut)
Puce personnalisée	list-style-image	Indiquer l'url de l'image qui servira de puce. Exemple : **Code : CSS** `list-style-image: url("images/puce.png");`
Super-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 : **Code : CSS** `list-style: inside square;`

Propriétés des tableaux

Type	Propriété	Valeurs possibles
Type de bordure	border-collapse	**collapse** : les bordures du tableau et des cellules sont mélangées. **separate** : les bordures du tableau et des cellules sont séparées (par défaut).
Cellules vides	empty-cells	**show** : les bordures des cellules vides sont affichées.

Cellules vides	empty-cells	**collapse** : les cellules vides sont masquées (par défaut).
Position du titre	caption-side	Indique la position du titre du tableau, défini via la balise <caption> **top** : en haut du tableau **bottom** : en bas du tableau **left** : à gauche du tableau **right** : à droite du tableau

Autres propriétés

Type	Propriété	Valeurs possibles
Curseur de souris	cursor	**auto** : curseur automatique (par défaut) **default** : curseur standard **pointer** : curseur en forme de main, comme quand on pointe sur un lien **text** : curseur utilisé quand on pointe sur du texte **wait** : curseur utilisé pour indiquer une attente (sablier) **progress** : curseur utilisé pour indiquer une tâche de fond (curseur avec sablier) **help** : curseur en forme de point d'interrogation, indiquant une aide **move** : curseur en forme de croix, indiquant un déplacement possible **n-resize** : flèche vers le nord **ne-resize** : flèche vers le nord-est **e-resize** : flèche vers l'est **se-resize** : flèche vers le sud-est **s-resize** : flèche vers le sud **sw-resize** : flèche vers le sud-ouest **w-resize** : flèche vers l'ouest **nw-resize** : flèche vers le nord-ouest **url** : curseur personnalisé, de type .cur ou .ani. Exemple : **Code : CSS** `cursor: url("images/curseur.cur"), auto;` Vous devez utiliser un logiciel dédié à la création de curseurs pour créer des .cur et des .ani. Notez aussi la présence du mot **auto** 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 **auto** sera utilisé.

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.

Le cours s'arrête ici ! Pensez à consulter les annexes si vous ne l'avez pas fait, vous y découvrirez de nouvelles pistes pour poursuivre votre apprentissage de HTML et CSS !