

# CSS : on reprend tout à zéro !

Par Joe Gillespie

Document original : <http://www.pompage.net/pompe/cssdezero-1/>

Pompage de © Copyright 1996-2007 WPDFD, LTD : [http://www.wpdfd.com/issues/70/css\\_from\\_the\\_ground\\_up/](http://www.wpdfd.com/issues/70/css_from_the_ground_up/)

Repompé et compilé par Stéphanie De Nadaï, [www.webdesigneuse.net](http://www.webdesigneuse.net)

## Introduction

Si l'idée d'utiliser des Feuilles de Styles en Cascade vous effraie, tranquillisez-vous. Le fait d'utiliser un ordinateur peut être terrifiant pour quelqu'un qui le découvre pour la première fois, mais après un certain temps, on n'y pense plus. Le tout, c'est d'y aller par petites étapes au départ, et c'est ce que je compte faire dans cette série de tutoriels. Une étape à la fois !

Que vous utilisiez un éditeur WYSIWYG sans jamais vous occuper du code source qu'il y a derrière, ou même si vous n'avez jamais créé la moindre page Web, ce tutoriel va vous mettre dans la bonne direction.

Je pars du principe que vous ne savez rien, ou pas grand-chose, de la façon dont on monte une page Web. Il s'agit vraiment de « tout reprendre à zéro ! »

## De quoi aurez-vous besoin ?

De rien de particulier. Si vous avez un éditeur de page web, très bien. Sinon, vous pouvez utiliser n'importe quel éditeur de texte : Bloc-Notes sur un PC, ou alors SimpleText ou TextEdit sur un Mac. Si vous utilisez déjà DreamWeaver ou GoLive, sachez que nous allons utiliser le mode « source. »

*Pas de panique !* Un éditeur graphique n'est pas essentiel, parce que je ne vais pas trop m'occuper de graphiques pour commencer.

Ah, vous aurez besoin d'un navigateur, ça va sans dire. En fait, je recommanderais de prendre plusieurs navigateurs différents. Si vous ne disposez que d'Explorer, vous devriez également prendre Mozilla (<http://www.mozilla-europe.org/fr/>) ou d'autres dans cette [liste des Nouveaux Navigateurs](http://www.wpdfd.com/editorial/wpd0204news.htm) (<http://www.wpdfd.com/editorial/wpd0204news.htm>) ( *établie en février 2004 et non remise à jour, NdT* ).

Si vous voulez télécharger vos pages vers un serveur Web, vous aurez besoin d'un programme FTP standard, mais ne vous en inquiétez pas pour l'instant, tout va être fait sur votre ordinateur.

## Première étape : une page de base

Avant toute chose, il nous faut une page Web de base. Voici la page Web la plus simple possible.

```
<html>
  <head>
</head>
  <body>
Voici ma page Web
  </body>
</html>
```

Nous avons ici trois ensembles de « balises »; les balises vont généralement par paires, mais pas toujours. Celle qui englobe

toute la page est la paire de balises `<html>...</html>`. La première `<html>` est la « balise ouvrante », et la « balise fermante » est identique sauf qu'elle a un caractère de barre oblique entre le `<` et le `html>`.

À l'intérieur de la balise `html` se trouvent deux autres paires de balises.

La partie `<head>...</head>` contient des informations qui ne vont pas se voir sur votre page Web, la plus importante étant le titre qui apparaît dans la barre de titre de votre navigateur. Il faut insérer le titre de votre page entre une paire de balises de titre comme ceci...

```
<html>
<head>
<title>Ma page</title>
</head>
<body>
Voici le contenu de ma page Web !
</body>
</html>
```

La paire de balises `<body>...</body>` contient l'intégralité de votre page Web, le texte, les images, les animations Flash, tout ce que vous voulez. Vous pouvez soit taper le code dans votre éditeur de texte ou dans la fenêtre « source » d'un éditeur WYSIWYG, soit le copier-coller à partir d'ici.



*Voici mon HTMLéphant. Bon d'accord, ce n'est pas très original, mais au moins vous vous en souviendrez !*

Donc, vous voilà avec une page Web qui fonctionnera dans n'importe quel navigateur (une fois que vous l'aurez enregistrée). Appelez-la « index.html », parce que c'est le nom de la première page sur tous les sites Web. Vous pouvez enregistrer le fichier n'importe où sur le disque dur de votre ordinateur, mais pour que les choses restent bien en ordre, créez un nouveau dossier pour y ranger vos pages Web.

La façon normale d'ouvrir votre page dans le navigateur est d'utiliser la commande « Ouvrir » dans le menu « Fichier » du navigateur, mais il y a un moyen plus rapide. Créez un raccourci pour votre fichier ou copiez-le, et placez-le sur le bureau. À partir de là, vous n'aurez qu'à faire glisser à la souris l'icône de votre fichier vers la fenêtre du navigateur, et votre page apparaîtra dans toute sa splendeur ! Elle devrait ressembler à ceci. ([http://www.pompage.net/IMG/html/cssdezero/page\\_modele01.html](http://www.pompage.net/IMG/html/cssdezero/page_modele01.html))

Bien qu'elle soit merveilleuse, il manque à votre page web un ingrédient essentiel : du contenu. Ça ne rime pas à grand-chose d'avoir une page Web si elle ne dit rien d'intéressant (en-dehors de « Voici le contenu de ma page Web ! »), mais c'est votre affaire.

Quel que soit le contenu, il devra être inclus entre la paire de balises `<body>...</body>` pour que le navigateur l'affiche.

Tant qu'on se contente de placer un texte dans une page Web nue comme ici, il n'a ni structure, ni style. Quand je parle de structure, je veux dire que les mots vont de gauche à droite et de haut en bas sans autre mise en valeur que l'ordre dans lequel ils apparaissent. Il est plus logique de regrouper ces mots et ces phrases en paragraphes, titres, sous-titres : bref, vous voyez, les trucs de base qu'on vous a appris à l'école.

Les navigateurs ignorent les retours à la ligne ou les paragraphes que vous avez dans votre texte brut. Ils ignorent aussi les tabulations, et si vous avez placé plus d'un espace entre deux mots, ils ignoreront également les espaces supplémentaires.

Il est utile, dans presque tous les textes, d'avoir un titre, un sous-titre, quelques paragraphes distincts et peut-être une signature à la fin.

Dans le HTML, le mécanisme permettant de faire ceci est fourni par encore d'autres balises de mise en forme.

On crée un paragraphe en incluant du texte au sein d'une paire de balises `<p>...</p>`. Un paragraphe en HTML, c'est tout simplement un bloc de texte d'une ou plusieurs phrases avec un petit bout d'espace pour le séparer du suivant : exactement comme ici.

Pour les titres, il y a six différents degrés de mise en valeur, qui vont du plus élevé `<h1>...</h1>` jusqu'au plus bas `<h6>...</h6>`. Ils ressemblent à ceci...

**Voici un titre h1**

**Voici un titre h2**

**Voici un titre h3**

**Voici un titre h4**

**Voici un titre h5**

**Voici un titre h6**

Comme vous le voyez, leur taille diminue quand la valeur de leur chiffre augmente, le `<h4>` faisant à peu près la même taille que la « petite » police du texte que vous lisez ici. `<h5>` et `<h6>` sont plus petits, mais en caractères gras.

Il existe une poignée d'autres petites astuces qui permettent de jouer avec le texte pour mettre en valeur certains mots ou expressions. Plutôt que de penser à leur aspect, réfléchissez à leur fonction réelle.

**Le caractère gras** est une version plus lourde de la police de texte, et se marque par `<b>...</b>`. Toutefois, le mot « gras » est un terme de style pour l'impression, et sur le Web il est préférable d'utiliser `<strong>...</strong>`. Même si cela revient au même sur votre écran, le HTML doit aussi fonctionner sur d'autres supports. Les logiciels qui lisent les pages Web aux personnes malvoyantes comprendront ce `strong` et le prononceront avec la force nécessaire.

*L'italique* s'applique, vous l'aurez deviné, grâce à `<i>...</i>`. Là aussi, avec HTML, mieux vaut ne pas utiliser ce style visuel mais plutôt `<em>...</em>`. On obtient ainsi un effet d'italique quel que soit l'outil qui reproduit le texte.

Le soulignement peut s'effectuer avec la paire de balises `<u>...</u>` mais un soulignement dans une page Web a un sens particulier. Il sert généralement à indiquer un lien. Mieux vaut ne pas utiliser le soulignement comme mode de mise en valeur, cela peut créer une confusion pour les lecteurs.

L'autre balise importante est le retour à la ligne `<br>`. Contrairement aux autres balises, celle-ci ne requiert pas de balise fermante, qui n'aurait pas beaucoup de sens d'ailleurs : où la mettrait-on ?

En utilisant ces balises de mise en forme de base, le texte commence à prendre forme. Il devient plus proche de ce que vous pourriez produire avec un traitement de texte.

Cette page de base aura sans doute un aspect différent dans d'autres navigateurs et sur d'autres ordinateurs. Chaque navigateur a un ensemble de styles par défaut, et sauf instruction contraire de votre part, il utilisera ces données par défaut. Pour ignorer ces styles par défaut, nous allons tout simplement produire nos propres styles qui seront regroupés dans une « feuille de styles » collective.

## **Deuxième étape : Une Feuille de Style.**

Une feuille de style est un concept assez simple, c'est une page de définitions ou de caractéristiques concernant le style qui

indiquent au navigateur comment afficher les divers éléments d'une page. Si vous vous demandez où le terme de « cascade » va arriver, ne vous inquiétez pas encore à son sujet. J'y reviendrai plus tard quand nous commencerons à appliquer les styles à notre [HTML](#).

Pour faire simple, nous allons écrire notre feuille de style dans la page Web. Plus tard, vous constaterez que vous pouvez avoir une « feuille de style externe » dans un fichier séparé, qui peut être appelé par plusieurs pages. Le grand avantage, c'est qu'une seule modification dans cette feuille de style principale affectera toutes les pages qui lui sont liées.

Les styles que nous allons employer sont définis dans la partie `<head>...</head>` de notre page comme ceci :

```
<style type="text/css" title="mes_styles" media="all">
<!--

-->
</style>
```

Ici, vous voyez une paire de balises `<style>...</style>` mais il y a en plus quelques attributs qui ont besoin d'être expliqués.

`type="text/css"` indique au navigateur que nous utilisons juste du texte pour décrire les styles, aucune fantaisie ici.

`title="mes_styles"` identifie simplement le style pour votre propre information.

`media="all"` C'est là où ça commence à devenir intéressant. Vous pouvez avoir une feuille de style pour décrire à quoi ressemble votre page sur un écran d'ordinateur (`media="screen"`) et une autre complètement différente pour définir l'impression (`media="print"`). Il y existe d'autres médias comme « projection », « tv », « braille » et « aural ». Maintenant vous pouvez comprendre la logique de ne pas employer « bold » et « italic ». Pour l'instant, nous emploierons seulement « all », qui est pour tous les usages.

Les caractères `<!--` et `-->` sont une manière de cacher le texte sur une page Web, on peut seulement le voir dans le code. Ceci s'appelle « commenter » mais comme les styles sont définis dans la section `<head>...</head>` de la page, ils ne devraient pas apparaître de toute façon.

La première chose que nous allons faire est de définir le style du corps de la page. Tout ce qui est contenu entre les balises `<body>...</body>` se verra appliquer ce ou ces styles.

La définition de base de la partie « body » ressemble à ceci : le mot **body** suivi d'une paire d'accolades.

```
body { }
```

Nous allons donner à la page une couleur de fond ...

Le style par défaut des navigateurs donne habituellement du texte noir sur une page blanche, mais nous allons changer cela en un gris pâle plus chaud. Voici comment nous définissons une valeur pour la couleur de fond du corps de la page. Notez que nous n'employons pas un signe égale « = » mais deux points « : ».

```
body { background-color: #e8eae8 }
```

Qu'est-ce que c'est, `#e8eae8` ?

Les couleurs sur les pages Web sont définies en mélangeant 256 nuances de rouge, de vert et de bleu dans diverses proportions. Les humains ont 10 doigts sur leurs mains, donc ils comptent par dix. Les ordinateurs préfèrent compter par seize, bien qu'ils n'aient pas de doigts, mais une fois que vous allez au-dessus de neuf, il n'y a aucun numéro pour représenter 10, 11, 12, 13, 14, et 15. Aussi nous leurs substituons les lettres a, b, c, d, e, f. Ainsi, quand on compte en « hexadécimal », 10 est représenté par « a » et 15 par « f ». Quand vous allez au-dessus de 15, vous ajoutez un deuxième chiffre et « 10 »

représente 16. En utilisant cette méthode, tous les nombres de 0 à 255 peuvent être représentés par deux nombres ou lettres. 255 équivaut ainsi à FF, #ffffff sera du blanc et #000000 du noir.

Notre couleur de fond est donc rouge e8 (232), vert ea (234), bleu e8 (232). Le signe dièse devant précise qu'il s'agit de nombres hexadécimaux et non de décimaux ordinaires.

Parfois vous verrez seulement trois caractères, par exemple, #2a0. C'est une notation courte pour 22aa00. Quand il y a les deux mêmes caractères dans chacune des trois valeurs de couleur, vous pouvez vous passer du second caractère et votre navigateur comprendra ce que cela signifie.

#fff = #ffffff

En utilisant seulement trois chiffres, vous pouvez avoir 4096 couleurs différentes. Avec six chiffres, les possibilités sont de seize millions.

0 1 2 3 4 5 6 7 8 9 a b c d e f

0 1 2 3 4 5 6 7 8 9 a b c d e f

0 1 2 3 4 5 6 7 8 9 a b c d e f

Avec des couleurs à un seul chiffre, le rouge, le vert et le bleu ont chacun 16 étapes de luminosité, qui peuvent être combinées de nombreuses façons afin de produire toutes les autres.

#fff = #93f = #999 =

Si vous codez à la main, employer des valeurs de couleur à trois chiffres est plus simple et probablement suffisant.

Quoi qu'il en soit, ajoutons maintenant une couleur pour la police afin de remplacer le noir. Elle a été choisie avec l'outil « pipette » dans Photoshop, c'est ainsi un nombre hexadécimal de 6 chiffres ...

```
body { background-color: #e8eae8; color: #5d665b }
```

Notez comment la couleur de fond et la couleur de premier plan (pour le texte) sont séparées par un point-virgule. Faites attention à ne pas mélanger les deux points et les points-virgules ou les choses tourneront vraiment mal.

Et maintenant, de façon à ce que le texte ne s'étende pas jusqu'aux bords de notre page, nous pouvons entourer la page d'une marge. « margin : 50px » est ajouté à notre définition de style pour la partie « body », de nouveau séparée de la précédente par un point-virgule.

```
body {  
  background-color: #e8eae8;  
  color: #5d665b;  
  margin: 50px  
}
```

N'importe quel texte que nous ajouterons maintenant à la page sera par défaut d'un gris-vert foncé sur un fond gris pâle. Nous verrons plus tard comment ceci peut être redéfini pour des cas spéciaux.

[Jetons un oeil sur ce que nous avons fait jusqu'ici. \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele02.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele02.html)

## Troisième étape : appliquer du style au texte

Nous nous sommes déjà débarrassés du côté brut du noir et blanc pour le remplacer par quelque chose de plus doux, mais on peut faire beaucoup plus. On peut changer la police d'écriture, la taille des caractères, la quantité d'espace entre les lignes et ajouter des détails de mise en valeur comme des retraits de paragraphes.

## La police d'écriture

Nous allons commencer par le style de police, dans la mesure où c'est ce qui va apporter le plus grand changement. À la différence du papier, les pages Web ne peuvent utiliser que les polices qui sont installées sur l'ordinateur de l'utilisateur : on ne peut donc pas se contenter d'indiquer la police qui nous plaît le plus, il faut utiliser celles qui sont communes à tous les ordinateurs (celles qui sont préinstallées avec le système). Cela réduit considérablement le choix. Vous allez vous apercevoir que vous êtes limités à deux ou trois polices *serif*, à autant de *sans-serif* et à un style de type machine à écrire, à espacement constant : *monospace*. Au lieu de ne choisir qu'un seul style de police, on indique une famille, dans l'espoir qu'une d'entre elles sera installée sur l'ordinateur du visiteur.



*Ci-dessus : quelques polices courantes sur les ordinateurs Windows et Mac. Vous ne pouvez jamais être certain que telle ou telle police est présente, il faut donc fournir des alternatives.*

Voici une indication courante de famille de polices sans-serif :

```
font-family: Verdana, Geneva, Arial, sans-serif;
```

Voici une famille de polices serif :

```
font-family: Georgia, "Times New Roman", Times, serif;
```

Remarquez que lorsque le nom d'une police contient plus d'un mot, l'ensemble doit être placé entre guillemets à l'anglaise : "Times New Roman".

Ajoutons les polices sans-serif à nos styles et voyons ce que ça donne.

[Notre page en police sans-serif >> \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele03.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele03.html)

## La taille des caractères

La question de la taille qu'on assigne aux caractères est très délicate. Pour l'impression, on peut fixer une taille de police en points, et le caractère aura toujours cette même taille. Il occupera toujours le même espace sur la page, et les retours à la ligne seront toujours à l'endroit que vous aurez choisi.

Sur le Web, les tailles de caractères ne sont pas gravées dans le marbre, et elles peuvent changer d'un ordinateur à l'autre, parfois du tout au tout. En fonction de la taille de l'écran de l'internaute, de son système d'exploitation et de son navigateur, les caractères peuvent devenir tellement petits qu'ils en sont illisibles, ou aussi grands que dans des méthodes de lecture pour enfants. Les internautes ont heureusement la possibilité de choisir une taille « confortable » dans les préférences de leur navigateurs, et donc de réduire un peu ces variations de taille, mais le résultat peut rester très différent de ce que vous attendiez.

Vue la popularité grandissante du Wi-Fi et des systèmes de poche, les modes de navigation sur vos pages vont changer radicalement dans les prochaines années. Pour que vos pages restent lisibles à long terme, vous devez raisonner en termes de maquettes *liquides* qui s'adaptent à la taille de l'écran.

Répétez après moi : « Le Web, ce n'est pas la même chose que le papier ! »

## Tailles de caractères relatives

Si vous voulez que vos pages touchent le plus large public, mieux vaut assigner une taille de caractères relative à la taille que le visiteur aura choisie par défaut. Il y a plusieurs façons d'assigner des tailles de caractères relatives. Vous pouvez utiliser des pourcentages de la valeur par défaut (%), ou bien une unité nommée « em », qui correspond à 100%. 1.2em revient à 120%. La meilleure méthode est d'utiliser un ensemble de descriptions pré-définies qui font penser à des tailles de tee-shirts. « Medium » revient à 100%, ou 1em. « Smaller » est une taille en-dessous, et « Larger » une taille au dessus. Il y a aussi x-small, xx-small, x-large et xx-large. L'intérêt d'utiliser ce système par rapport aux pourcentages ou aux ems, c'est que les navigateurs ne laisseront pas la police d'écriture devenir si petite qu'on ne puisse plus la lire. Imaginez, par exemple, que vous indiquiez une taille de caractères de 0.7em ou de 70% qui rendrait très bien sur votre PC. Pour quelqu'un équipé d'un Mac avec une taille de caractères par défaut plus petite, ce 70% pourrait faire passer vos caractères sous le seuil de la lisibilité. xx-small est une bonne solution pour déterminer quelle est la taille de caractères minimum que l'ordinateur peut afficher.

**xx-large** x-large large medium small x-small xx-small

Caractères xx-small : C'est petit mais le texte reste lisible

Avec une taille fixe trop petite : C'est beaucoup trop petit

## Tailles de caractères absolues

Si votre site ne concerne qu'un nombre limité de visiteurs qui utilisent à peu près le même type d'ordinateur que vous, vous pouvez également indiquer des tailles de caractères en pixels. Les pixels ne varient pas tellement d'un ordinateur de bureau à l'autre, et la taille des caractères sera similaire, sinon identique. Dès que l'on passe à des ordinateurs portables ou bien de poche, ou encore à des moniteurs très grands et à haute résolution, les tailles de pixels s'éloignent de la norme, et la taille des caractères sera modifiée d'autant.

Pour que vous puissiez percevoir les différences, j'ai placé ci-dessous trois paragraphes de texte en utilisant trois méthodes : les pourcentages, les tailles de tee-shirts et les pixels. Sur mon écran sous Mozilla, leurs tailles sont quasi-identiques, mais ce n'est peut-être pas le cas sur le vôtre.

Cette ligne est définie à 85% de la taille du texte qui l'entoure.

Cette ligne est définie en x-small.

Cette ligne est définie à 11px absolument.

Notez qu'il ne faut *jamaïs* utiliser les « points » pour indiquer des tailles de caractères, c'est une unité d'impression.

## L'interlignage

La quantité d'espace entre les lignes est d'environ 120% de la taille de caractères par défaut. On peut en ajouter un peu plus, ce qui améliore généralement la lisibilité, surtout si les lignes sont longues. J'évoquerai la longueur des lignes plus tard. Dans

notre page d'exemple actuelle, la longueur des lignes est déterminée par la largeur de la fenêtre du navigateur, donc elle est peut-être plus grande qu'il ne faudrait.

Pour changer l'interlignage, nous avons à nouveau l'option relative (en % ou en ems) ou absolue (en px). Dans notre exemple, j'ai indiqué une hauteur de ligne (`line-height`), c'est-à-dire la hauteur du caractère et de l'espace supplémentaire au dessus, à 180% de la taille de caractères (`small`) pour toute la page. Comme la définition du style de la partie « body » commence à être un peu longue, je l'ai séparée en plusieurs lignes. Le navigateur s'en moque, mais c'est plus facile à lire par des humains. Tant que les accolades sont présentes et que chaque définition de style est séparée des autres par un point-virgule, tout va bien.

```
body {
background-color: #e8eae8;
color: #5d665b;
margin: 50px;
font-family: Verdana, Geneva, Arial, sans-serif;
font-size: small;
line-height: 180%
}
```

[Nous avons maintenant une page plus « aérée » >> \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele04.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele04.html)

## Les paragraphes

Tout ce que nous avons réalisé jusqu'à présent concernait l'ensemble de la page. Maintenant nous allons voir comment appliquer notre style à des zones plus précises.

Comme je l'ai évoqué précédemment, de gros morceaux de texte peuvent être découpés en paragraphes pour les rendre plus faciles à lire. Il existe plusieurs façons de séparer visuellement les paragraphes. On peut ajouter de l'espace supplémentaire, ou pourquoi pas faire un retrait de première ligne.

En typographie créative, les designers souhaitent parfois utiliser d'autres techniques moins courantes, comme un retrait d'à peu près la moitié de la largeur de la colonne, ou un retrait négatif. Les CSS sont capables de gérer tout cela, mais le séparateur de paragraphes par défaut est un « espacement de paragraphe » qui fait à peu près la moitié de la hauteur de ligne (`line-height`).

Pour qu'un bloc de texte devienne un paragraphe, contentez-vous de l'encadrer par une paire de balises `<p>...</p>`. Pour pouvoir manipuler le style des paragraphes avec CSS, on ajoute `p { }` aux définitions de style dans la partie « head » de la page, à la suite des styles pour « body ». Entre les accolades, ajoutez `text-indent: 3em;` (*indent* est le mot anglais pour « retrait », *NdT*)

```
<style type="text/css" title="mes_styles" media="all">
<!--
body { background-color: #e8eae8;
color: #5d665b;
margin: 50px;
font-family: Verdana, Geneva,
Arial, sans-serif;
font-size: small;
line-height: 180%
}

p { text-indent: 3em
} -->
```

</style>

[Cette page est divisée en paragraphes >> \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele05.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele05.html)

Remarquez que le premier paragraphe ne présente pas de retrait : il n'est pas encadré par `<p>...</p>`, donc ce n'est pas officiellement un paragraphe (en termes de HTML).

## Les titres

J'ai déjà expliqué les six niveaux de titre que le HTML propose par défaut. Rien ne nous oblige à les conserver tels quels, nous pouvons les redéfinir selon nos besoins. Là encore, il ne s'agit que d'ajouter d'autres définitions aux styles.

Par défaut, les titres sont en grands caractères gras, et dotés d'espacements supplémentaires au-dessus et en-dessous. Souvenez-vous que les titres `h1`, `h2` et `h3` sont plus gros que le texte normal, tandis que `h5` et `h6` sont plus petits. Amusons-nous avec un titre `h3` pour changer sa couleur et sa police de caractères.

```
h3 {color: #966b72; font-family: Georgia, "Times New Roman", Times, serif }
```

[Nous avons maintenant des titres >> \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele06.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele06.html)

Vous remarquez peut-être que j'ai ajouté une ligne vide sous le deuxième titre. Si vous vous contentez de mettre un paragraphe vide (`<p></p>`) cela ne marchera pas, il doit y avoir quelque chose entre les balises. Un espace ne fonctionnera pas non plus (`<p> </p>`) parce que HTML ignore les espaces s'ils ne sont pas entre deux caractères. Ce dont nous avons besoin, c'est d'un caractère invisible que nous fournit HTML sous le nom d'*espace insécable* que l'on peut taper `&nbsp;`, comme ici :

```
<p>&nbsp;</p>
```

L'espace insécable est également très utile placé entre deux mots que vous ne souhaitez pas voir séparés lorsque la ligne se termine : des noms de personnes ou d'entreprises par exemple.

## D'autres petites manips

Avant d'aborder la partie suivante, essayons de modifier quelques petites choses dans notre page de texte.

Dans la définition de la partie « body », j'ai mis une marge de 50 pixels pour tous les côtés. Il est possible d'avoir différentes marges pour la gauche, la droite, le haut et le bas. Il n'y a qu'à les séparer comme ceci (respectivement marge du haut, de gauche, de droite et du bas, *NdT*) :

```
margin-top: 70px;  
margin-left: 120px;  
margin-right: 50px;  
margin-bottom: 70px;
```

Cela va donner quelque chose plus proche d'une page imprimée, avec une marge de gauche plus importante. J'ai aussi appliqué à deux ou trois mots du gras (`strong`) et des italiques (`em`). Attention, les italiques rendent mal sur certains ordinateurs, et deviennent particulièrement difficiles à lire quand la taille est réduite.

[Voyez notre page quand le texte a reçu ses styles >> \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele07.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele07.html)

Désormais, vous disposez d'une page qui commence à avoir du style, et sans avoir fourni trop d'efforts. Toutefois, elle

ressemble encore à un document tapé à la machine ou à l'aide d'un traitement de texte. Dans le prochain article, nous verrons d'autres possibilités de mise en page plus intéressantes.

En attendant, vous pouvez vous amuser avec les valeurs des styles de la page que vous venez de créer. Essayez différentes polices et tailles de caractères, différents interlignages. Mais surtout, visionnez votre page avec autant de navigateurs différents que possible, pour observer toutes les différences de rendu.

## Quatrième étape : plus de mise en forme pour le texte

Avant d'aller plus loin dans la mise en forme du texte, c'est une bonne idée de vous familiariser avec quelques termes basiques de typographie.

Quand nous parlons de la taille de la police, ou `font-size` en CSS, cela fait référence à la distance entre le sommet d'une lettre majuscule comme « A » jusqu'au bas d'une lettre à jambage comme « p » ou « y ». Il peut aussi y avoir un léger espace supplémentaire appelé « interlignage ». Ce mot vient du fait que les imprimeurs utilisaient des petites lattes de plomb entre les lignes pour leur donner plus d'espace. (NdT : En anglais, « interlignage » se dit « leading » et le plomb se traduit « lead ».)



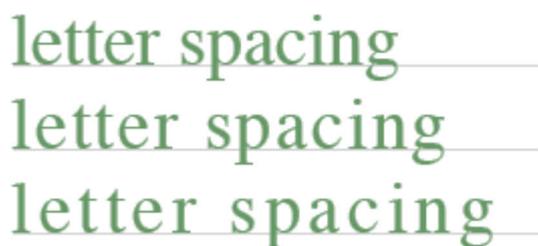
Aujourd'hui, on n'utilise plus de plomb. On dit juste qu'il y a un certain espace entre la base d'une ligne de caractères et la suivante. L'interlignage est donc la hauteur du caractère plus cet espace supplémentaire. Dans une définition CSS, nous pourrions dire `line-height: 180%`; avec la valeur spécifiée en pourcentage, em ou pixels (px). On reviendra sur l'interlignage un peu plus loin.



`Vertical-align` vous permet d'ajuster la ligne de base pour des lettres individuellement par rapport au reste du texte mais relève plutôt pour un usage spécialisé (comme les formules mathématiques), et on ne va pas s'étendre sur son cas ici. Dans l'exemple basique ci-dessous, j'aurais pu utiliser la balise `<sup>2</sup>` mais le recours à CSS permet plus de contrôle.

$$E=MC^2$$

Les CSS nous donnent également le contrôle sur l'interlettrage (ou crénage), c'est-à-dire l'espace entre chaque caractère ou mot.



C'est très pratique pour ajouter un style visuel à un gros titre.

UN TITRE ESPACÉ.

```
letter-spacing: 0.5em; word-spacing: 0.5em
```

## Penser à la lisibilité

Avant que les navigateurs ne supportent aussi bien les CSS qu'à l'heure actuelle, il y avait une caractéristique essentielle qui manquait au formatage du texte en HTML, et que maîtrisait même la plus modeste des machines à écrire, l'interlignage. Quand il y a trop peu ou pas du tout d'espace entre les lignes, elles deviennent déprimantes et difficiles à lire. Et plus les lignes s'allongent, plus le problème s'accroît.

Des lignes trop courtes interrompent trop le schéma de lecture de l'œil de la gauche vers la droite.

Des lignes plus longues que la normale sont aussi fatigantes et le début de la ligne suivante devient difficile à localiser.

Pour une lecture confortable, une ligne de texte devrait être de la même longueur que une fois et demie à deux fois la longueur d'un alphabet en minuscules. Ce qui signifie entre quarante et soixante caractères ou huit à dix mots.

Si les lignes sont plus longues, un interlettrage plus grand aide à la lisibilité. Au début, beaucoup de pages web avaient leurs textes qui s'étendaient d'un bord à l'autre de la fenêtre, sans se soucier de la taille d'écran. Les écrans sont devenus plus larges, mais la pratique continue chez certains.

Par défaut, il y a un petit interlettrage dans les paramètres de texte des navigateurs. C'est approximativement 20 % de la taille de la police (hauteur d'une majuscule plus le jambage inférieur). Avec CSS vous pouvez avoir autant d'interlignage que vous le souhaitez. Si vous utilisez des tailles de polices relatives, vous pouvez le spécifier comme un pourcentage de la taille de la police. Exemple : `font-size: 1em; line-height: 1.5em` (ce qui équivaut à 150%). Si vous utilisez des tailles de polices fixes, vous pouvez indiquer quelque chose comme : `font-size: 11px; line-height: 16px`.

La définition pour cette page (NdT: la [page \(http://www.wpdfd.com/editorial/basics/cssbasics4.html\)](http://www.wpdfd.com/editorial/basics/cssbasics4.html) originale de l'auteur) est `font-size: small; line-height: 180%` ce qui correspond à peu près à une fois et demie l'interlignage dans un traitement de texte ou une ancienne machine à écrire. Ainsi, dans cet [exemple \(http://www.pompage.net/IMG/html/cssdezero/page\\_modele07a.html\)](http://www.pompage.net/IMG/html/cssdezero/page_modele07a.html) le haut de la page n'a pas de marges et le texte n'a pas de paramètres d'interlignage. Les spécifications générales pour l'élément `body` de la page fournissent la famille de police, la taille de la police et les couleurs pour les caractères et le fond de la page.

```
body {
color: #5d665b;
font-size: small;
font-family: Verdana, Geneva, Arial, sans-serif;
background-color: #e8eae8
}
```

Le bloc de texte en bas de l'exemple est encapsulé dans une division de la page `<div>...</div>`. C'est comme une page dans une page. Elle possède une marge gauche de 250px, une marge droite de 200px, et l'interlignage est défini à 180%. Ces styles sont ajoutés aux spécifications générales de l'élément `body` par un procédé appelé « héritage ». Exactement comme un enfant qui peut hériter de certaines caractéristiques de ses parents, puis à son tour les transmettre à ses enfants, certains styles CSS « cascaded » quand ils sont imbriqués dans d'autres styles.

```
<div style="margin-left: 250px; margin-right: 200px;
line-height: 180%">...</div>
```

Les divisions de page (`<div>`) sont une part *très* importante de CSS et nous les étudierons plus en détail la prochaine fois.

## Le contraste

Contrairement à une page imprimée que vous lisez par la réflexion de la lumière, la page sur un écran d'ordinateur est lumineuse. La différence de luminosité entre le texte noir et la page blanche est d'une amplitude beaucoup plus grande sur un écran que sur une page imprimée. Quand on lit de grandes zones de texte noir sur fond blanc, c'est un peu comme quand on se promène sous un grand soleil sans lunettes noires. On finit peut-être par se protéger les yeux en mettant sa main en visière, mais par défaut on a plutôt tendance à plisser des yeux et à se sentir mal à l'aise sans vraiment savoir pourquoi.

Réduire le contraste du texte des pages web permet une lecture plus confortable et moins stressante. Cela dit, si vous diminuez trop le contraste, vous commencez à rendre les choses difficiles pour les gens ayant une faible vision.

Dans un article comme celui-ci (NdT: la [page \(http://www.wpdfd.com/editorial/basics/cssbasics4.html\)](http://www.wpdfd.com/editorial/basics/cssbasics4.html) originale de l'auteur) où il y a beaucoup de texte à lire et où le lecteur est amené à rester quelque temps devant son écran, il est important d'offrir un environnement aussi confortable que possible. J'ai donc utilisé une teinte de fond légère et des couleurs douces.

## Cinquième étape : des listes à gogo

Une autre technique très courante pour mettre en forme des documents consiste à utiliser des « listes ». Elles ressemblent à des paragraphes, avec de petites choses en plus.

HTML peut nous fournir des listes de base qui ont soit des puces (listes non numérotées) `<ul>...</ul>`, soit des chiffres (listes numérotées) `<ol>...</ol>` en tête de chaque élément de la liste `<li>...</li>`. Chaque catégorie présente quelques options que l'on peut indiquer dans le HTML : `li type="square"` donne des puces carrées, `li type="i"` donne des chiffres romains en bas de casse.

*Liste non numérotée par défaut :*

- Élément de liste non numérotée
- Élément de liste non numérotée
- Élément de liste non numérotée

*Liste non numérotée avec puces carrées :*

- Élément de liste non numérotée
- Élément de liste non numérotée
- Élément de liste non numérotée

*Liste numérotée par défaut :*

1. Élément de liste numérotée 1
2. Élément de liste numérotée 2
3. Élément de liste numérotée 3

*Liste ordonnée avec chiffres romains :*

- i. Élément de liste numérotée 1
- ii. Élément de liste numérotée 2
- iii. Élément de liste numérotée 3

Les CSS vous offrent davantage de choix et de contrôle : il suffit d'ajouter une définition pour `ol` ou `ul` à vos styles.

```
ol {list-style-type: lower-roman; margin: 1em 0 1em 40px }
```

Cela revient au même que si vous l'ajoutiez à chaque élément de liste en HTML, et vous avez tout contrôle sur les marges qui entourent l'ensemble de la liste. Les quatre valeurs `1em 0 1em 40px` correspondent à haut, droite, bas et gauche, et peuvent être exprimées en ems, en pourcentages ou en pixels.

Si vous souhaitez mieux contrôler individuellement les éléments de liste, vous pouvez également leur attribuer des marges, pour avoir davantage d'espace entre les lignes : par exemple...

```
ol li { margin: .5em 0 .5em 0 }
```

...ajoute un demi-em d'espace au dessus et en-dessous de chaque élément de liste pour une liste numérotée. Pour une liste non numérotée, il faudrait le changer en `ul li`.

Mieux encore, vous pouvez utiliser vos propres graphiques comme puces.

```
ul { list-style-image: url(images/smiley.gif) }  
ul li { margin: 1em 0 1em 0 }
```

- 😊 Ça, c'est génial
- 😊 Ça aussi, vous aimerez
- 😊 Et ça aussi bien sûr!

Et le top, une image de puce différente pour chaque élément de liste...

```
<li style="list-style-image:url(images/icone1.gif);  
margin: 1em 0 1em 0">Des hommes</li>  
<li style="list-style-image:url(images/icone2.gif);  
margin: 1em 0 1em 0">Des images</li>  
<li style="list-style-image:url(images/icone3.gif);  
margin: 1em 0 1em 0">Des idées</li>  
<li style="list-style-image:url(images/icone4.gif);  
margin: 1em 0 1em 0">Des écrits</li>
```

-  Des hommes
-  Des images
-  Des idées
-  Des écrits

Les styles définis dans la partie « head » concernent l'ensemble de la page, ce que nous ne souhaitons pas dans le cas présent. Pour appliquer des styles séparément à des éléments de liste, il faut le faire « sur place », ou bien encore « inline ». Les définitions sont insérées entre les balises HTML `<li>` et `</li>`, dans la partie « body » de la page. Notez cependant qu'[IE6](#) (Windows) place les icônes très près du texte. J'ai ajouté un peu d'espace supplémentaire au début de chaque ligne pour compenser, comme ceci :

&nbsp; Des hommes.

## Sixième étape : établir des liens

L'important, dans le mot « HTML », c'est la partie « HT », c'est-à-dire Hyper Texte. Sur une page, des liens qu'on ajoute à des mots, des phrases ou des images, peuvent être cliqués pour vous transporter à un autre endroit. Ces autres endroits s'appellent des « ancrs ». Imaginez des bateaux immobiles sur la grande mer du Web, avec une chaîne constituée de liens qui descend jusqu'à une ancre posée au fond de la mer. Les bateaux sont ancrés à des emplacements bien précis, ce qui les rend plus faciles à repérer que s'ils dérivait sur la mer.



Chaque page contient au moins une ancre. Celle par défaut est tout en haut de la page, mais vous pouvez ajouter d'autres ancrs à n'importe quel point d'une page auquel vous souhaitez pouvoir accéder d'un simple clic.

```
<a href="http://www.wpdfd.com/index.htm">  
Ceci est un lien vers le haut de ma page d'accueil  
</a>
```

Vous voyez là une paire de balises `<a>...</a>`, avec en plus une partie `href="..."` dans la première balise. Il s'agit de l'adresse hypertexte de référence où vous souhaitez vous rendre.

Si vous placez une ancre à un emplacement bien précis dans une page (en fait, il faudrait la placer juste au-dessus de l'endroit où on veut atterrir), cela donne :

```
<a name="mon_ancre"></a>
```

Il s'agit donc d'une paire de balises `<a></a>` avec, dans notre exemple, `name="mon_ancre"` placé dans la balise ouvrante. Il n'y a rien entre les deux balises.

Pour accéder à cet emplacement, il faut ajouter un petit bout de code à l'extrémité du lien :

```
<a href="http://www.wpdfd.com/index.htm#mon_ancre">  
Ceci est un lien vers une ancre de ma page d'accueil</a>
```

Comme vous le savez, les liens sur une page sont indiqués en les rendant visuellement différents du texte environnant. Par défaut, on souligne le texte du lien, et on le colorie en bleu. Quand vous cliquez sur un lien, il réagit visuellement en passant à la couleur rouge. Quand vous revenez ensuite à ce lien, il est passé à une autre couleur encore, le violet, pour montrer qu'il a déjà été visité.

Quiconque a déjà utilisé une page Web comprend le principe de ces conventions très rapidement.

La présentation par défaut des liens hypertextes en HTML se compose de trois états distincts.



Remarquez que le curseur change, lui aussi. Le pointeur qui était par défaut une flèche se transforme en main qui montre de l'index lorsque le curseur survole un lien.

Bien sûr, en tant que « designers », l'aspect par défaut des liens ne nous satisfait pas. Le texte souligné, cela peut être pratique mais aussi parfois très laid. Ce qu'il faut en priorité, c'est qu'un lien soit d'un aspect suffisamment différent de son

environnement pour que l'on voie qu'il y a là quelque chose de particulier. Le contexte est également un facteur important. Tel ou tel texte constitue de toute évidence un « menu » de choix, qu'il soit souligné ou non. Sa mise en valeur, sa position dans la page, le fait que les mots vous invitent à vous rendre à d'autres endroits sont autant d'indices quant à sa fonction.

De la même façon, quand un mot ou un groupe de mots au sein d'un bloc de texte a un aspect différent, c'est sans doute pour une bonne raison. Cette raison pourrait être que l'on veut insister sur l'importance de cette partie, mais là encore, c'est le contexte qui donne l'indice, et aussi le fait que d'autres mots ou groupes de mots similaires présentent partout le même aspect. Tout cela ne fonctionnerait pas si chaque lien était d'une couleur différente.

Les CSS nous permettent de jouer un peu avec l'aspect des liens. Non, les liens ne sont pas **obligatoirement** soulignés et bleus. Tout ce qu'il y a à faire, c'est d'indiquer une définition pour « a » (les ancres).

```
a:link { color: #696; text-decoration: none }
```

Comme cette page (*la page originale (<http://www.wpdtd.com/editorial/basics/cssbasics6.html>) de J. Gillespie, NdT*) présente un style de couleurs qui sort un peu du blanc et noir, j'ai changé les couleurs des liens en autre chose que le bleu/violet par défaut. De plus, `text-decoration:none` nous débarrasse du soulignage.

Pour qu'un lien déjà visité soit d'une couleur différente, tapez ceci :

```
a:visited { color: #699; text-decoration: none }
```

En CSS, il existe aussi un état supplémentaire appelé « hover ». Il permet de changer la couleur du texte au moment où le pointeur de la souris le survole, ce qui donne très utilement un indice supplémentaire qu'il s'agit d'un lien hypertexte.

```
a:hover { color: #c93; text-decoration: underline }
```

Vous voyez ainsi apparaître le soulignement traditionnel si vous le souhaitez, mais seulement de manière temporaire, lorsque le pointeur survole le texte.

L'état « actif » d'un lien, c'est ce que l'on voit à l'instant où on clique sur un lien. Il change généralement de couleur, et certains navigateurs font aussi apparaître une boîte autour du texte.



lien non visité   état survolé   état actif   lien déjà visité

Pour définir des liens qui s'appliquent à toute la page, il faut placer quelque chose comme ceci dans les définitions de style dans la partie « head » de la page :

```
a:link
{color: #696;
text-decoration: none;
background-color: transparent }
a:visited
{ color: #699;
text-decoration: none;
background-color: transparent }
a:hover
{ color: #c93;
text-decoration: underline;
background-color: transparent }
a:active
{ color: #900;
```

```
text-decoration: underline;
background-color: transparent }
```

L'ordre des définitions de style est important ici. D'habitude, l'ordre n'a aucune importance en définitions CSS, mais ici c'est **très** important de placer les définitions `a:hover` et `a:active` en dernier, sinon elles risquent de ne pas fonctionner.

Voici maintenant un autre état, pas officialisé en CSS.

The image shows the text "état mort" in a light gray font. A mouse cursor is positioned over the text, indicating it is being interacted with.

C'est un lien mort. Il ne fait rien de particulier. Si vous avez un lien sur une page qui ramène à cette même page, ma foi, ça ne rime pas à grand-chose de pouvoir y cliquer pour revenir au point de départ. Ce serait comme un panneau indicateur retourné sur lui-même et indiquant sa propre direction.

Il y a deux façons de gérer cela. Vous pouvez transformer le lien en titre, en le mettant davantage en valeur que le reste, ou bien vous pouvez le rendre pratiquement invisible, montrant ainsi que ce n'est pas un choix possible. On appelle cela le « grisage », et vous en reconnaîtrez le principe pour l'avoir vu en usage sur d'autres programmes de votre ordinateur.

Mais, direz-vous, pourquoi le placer là si on ne souhaite pas que les gens le choisissent ?

Eh bien, l'un des principes les plus importants dans le design d'une interface est que l'on essaie de garder pour tous les éléments la plus grande cohérence possible. Si on retire un élément d'un menu, par exemple, certains autres vont changer de place. Ce sera moins gênant pour l'utilisateur si l'interface reste clouée en place.

Enfin, l'autre élément dont nous disposons ici est la couleur de fond (`background-color`) derrière le texte. Les designers changent parfois la couleur de fond pour donner un effet proche du surlignage pour les états `hover` ou `active`.

The image shows the text "état survolé" in a yellow font. A hand cursor is positioned over the text, indicating it is being interacted with.

Il ya encore beaucoup d'autres choses à faire avec les liens CSS, nous n'avons fait qu'effleurer la question, mais cela devrait suffire à vous permettre de démarrer. Cette [page d'exemple \(http://www.pompage.net/IMG/html/page\\_modele08.html\)](http://www.pompage.net/IMG/html/page_modele08.html) met en application les principes que vous venez d'apprendre.

## Septième étape : les boîtes CSS

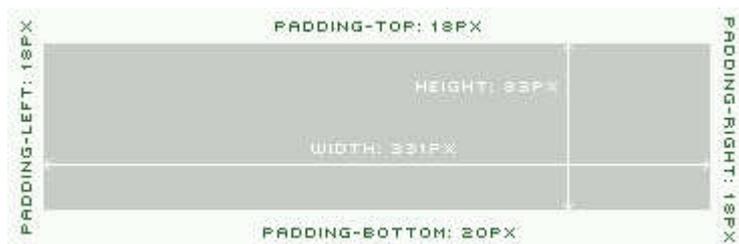
Nous avons déjà vu que le texte dans une page Web peut être découpé en titres ou en paragraphes, et que le langage `HTML` prend en charge ces opérations. Nous avons également vu comment les styles par défaut peuvent être modifiés à votre convenance, en leur attribuant de nouvelles définitions de style qui remplacent celles d'origine. Maintenant, je vais me pencher sur la façon de créer vos propres définitions de style pour étendre les possibilités du `HTML` et faire des mises en page plus intéressantes.

La partie « `body` » d'une page contient tout ce qui est visible ; cependant, comme je l'ai montré à [l'étape 3](#), il est possible de découper cette partie en divisions plus petites et qui ont leurs propres styles. Ces divisions peuvent ne contenir qu'un simple caractère, ou au contraire inclure de grands pans rectangulaires de la page. Tout ce que nous avons à faire, c'est d'entourer la zone que nous voulons styler d'une paire de balises `<div>...</div>`. Chacune de ces divisions est comme une mini-page, et on les appelle souvent « boîtes CSS ». Les boîtes CSS sont les matériaux de base que l'on utilise pour construire des mises en page pour le Web, et je vais m'attarder sur la façon dont elles fonctionnent parce que c'est très important.

Les boîtes CSS peuvent avoir une hauteur et une largeur, une couleur de fond ou même une image de fond que l'on peut faire répéter pour créer un effet graphique. Par défaut, une boîte CSS s'étend de la marge gauche à la marge droite du corps de la page. Si vous n'avez pas indiqué de marge pour la partie « `body` », elle prendra donc toute la largeur de la fenêtre du navigateur. Si vous n'indiquez pas de hauteur pour une boîte CSS, elle n'en aura pas. Si vous y placez du texte, elle s'étendra

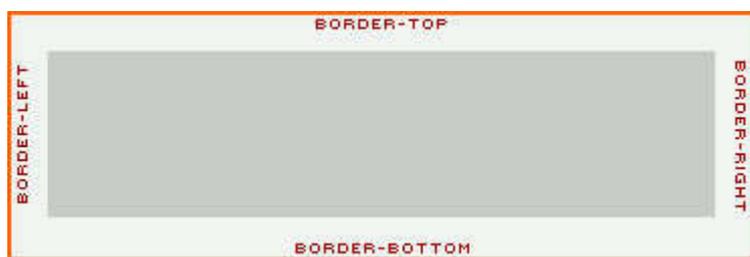
verticalement pour contenir ce texte (ou cette image).

Pour éloigner du texte du bord d'une boîte, on peut ajouter du remplissage (« padding ») comme ceci :

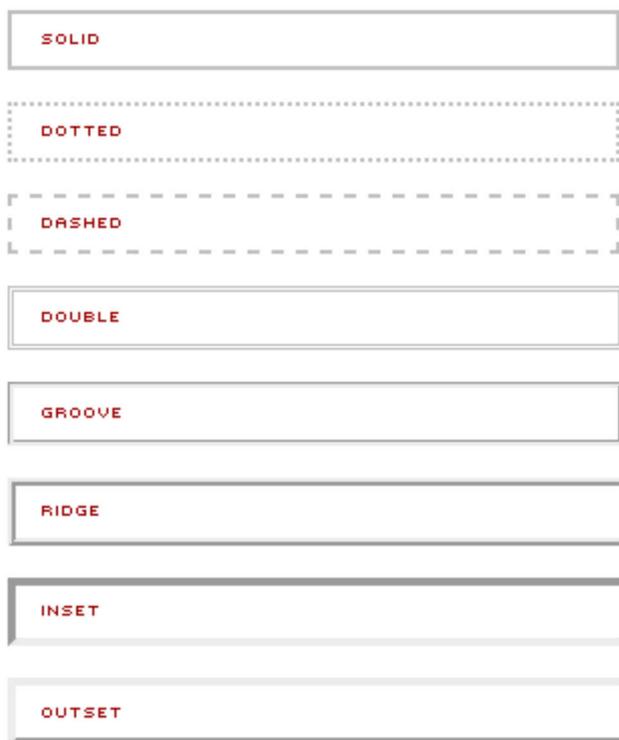


Mieux vaut faire très attention avec le remplissage, parce que son interprétation dans Internet Explorer est très différente de celle de tous les autres navigateurs. Contrairement à notre illustration, IE place le remplissage à l'intérieur de la boîte, de telle sorte que sa hauteur et sa largeur ne changent pas. Tous les autres navigateurs placent le remplissage en-dehors de la boîte, ajoutant ainsi à sa largeur et à sa hauteur. Donc la taille de votre boîte peut varier selon le navigateur avec lequel vous voyez la page, ce qui peut avoir des conséquences désastreuses si vous êtes tenus à des mesures précises au pixel près.

Après le remplissage, vous pouvez placer une bordure (« border ») :

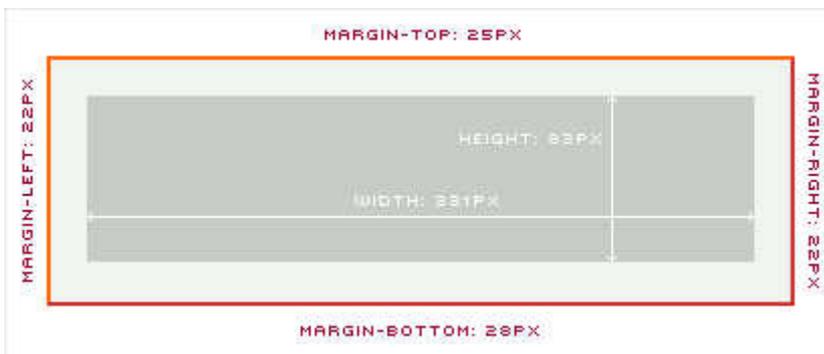


Les bordures existent dans plusieurs styles différents, et vous pouvez traiter chacun des quatre côtés séparément pour ce qui concerne l'épaisseur, le style et la couleur. Leur aspect varie un peu selon les navigateurs, mais voici à quoi elles ressemblent dans Mozilla :



L'épaisseur de la bordure augmente d'autant la taille d'une boîte.

Et pour séparer les boîtes les unes des autres, vous pouvez leur donner des marges (« margins »).



Il est fréquent de confondre les marges et le remplissage, surtout que ce qu'on appelle couramment la « marge » d'une page dans un livre ou une page Web est en fait du « remplissage » dans la mesure où c'est un espace qui ne peut sortir de la page, il est forcément à l'intérieur. Mais souvenez-vous que c'est un cas particulier. Pour le « body » d'une page Web, la marge va des côtés vers l'intérieur, tandis que toutes les autres marges CSS vont vers l'extérieur.

## Le positionnement absolu

En plus d'avoir une hauteur et une largeur, les boîtes CSS peuvent être placées à n'importe quel emplacement sur la page en indiquant `position: absolute;`

Le principe du positionnement absolu est facile à comprendre, il correspond à la façon de voir les choses dans la vie de tous les jours. Par exemple, votre maison peut être située à 50 mètres de la Rue du Nord et à 100 mètres de la Rue de l'Est.

La façon la plus courante de donner une position à une boîte est tout simplement d'indiquer quelque chose comme `top:50px;` `left:100px;`, mais on n'est pas obligé d'utiliser `top` et `left`, cela peut aussi bien être `top` et `right`, `bottom` et `left` ou `bottom` et `right`.

Voici quelques boîtes CSS ([http://www.pompage.net/IMG/html/page\\_modele09.html](http://www.pompage.net/IMG/html/page_modele09.html)) avec différents styles, chacune affichant sa définition de style. Les possibilités de combinaisons sont infinies.

Quand les boîtes sont placées à des coordonnées absolues, avec des largeurs et hauteurs bien précisées, cela ressemble beaucoup à un travail de collage de coupures de presse ou de photos dans un cahier de souvenirs. Le problème, c'est que les cahiers conservent toujours la même taille, et pas les pages Web.

## Le positionnement relatif

Si vous cherchez à décrire la position de la maison du voisin d'à-côté, vous pouvez dire qu'elle est à 70m de la rue du Nord et à 100m de la rue de l'Est, mais vous pouvez aussi dire qu'elle est à 5m, juste après la vôtre. Le positionnement relatif part du principe que les boîtes CSS sont placées *l'une après l'autre*. La première boîte est celle du dessus, la suivante est en-dessous, et la suivante encore en-dessous. De fait, elles *flottent* de haut en bas, à partir du haut de la page. Ce n'est peut-être pas très réaliste, mais imaginez une piscine couverte, avec à l'intérieur toute une pile de matelas gonflables. Le premier flotterait jusqu'à la surface, le suivant flotterait vers le haut et se retrouverait coincé sous celui du dessus, et ainsi de suite.

Pour obtenir que les boîtes CSS soient côte-à-côte au lieu de l'une au-dessous de l'autre, il faut leur dire de flotter à gauche ou à droite : `float:left;` ou `float:right;`. Tout comme nos matelas gonflables si la piscine est suffisamment large, elles se tiendront côte-à-côte, tant qu'il y aura de l'espace disponible. Puis elles glisseront à l'étage inférieur. Mieux vaut éviter cette situation, en vous assurant que les largeurs cumulées de toute une rangée de boîtes n'excèdent pas la largeur de la page. On peut le faire en précisant la largeur en pixels, mais en maintenant le total en-dessous de la largeur minimum d'un navigateur. On peut aussi utiliser les pourcentages, à condition de vérifier que la somme des pourcentages arrive bien à 100, ou mieux, un peu moins. Dans un tel contexte de positionnement relatif, évitez de mélanger pixels et pourcentages, sans quoi les résultats sont imprévisibles.

Ces exemples de boîtes flottantes ([http://www.pompage.net/IMG/html/page\\_modele10.html](http://www.pompage.net/IMG/html/page_modele10.html)) montrent le comportement de

boîtes positionnées de manière relative, en colonnes simples ou multiples.

## Huitième étape : des *div* sur mesure

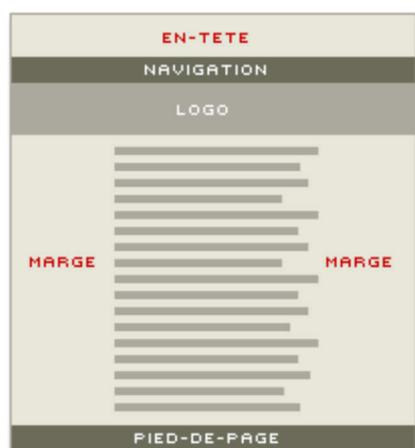
### Qu'est-ce qu'une *id* ?

Lorsque vous créez des boîtes CSS avec des définitions personnalisées, il n'existe pas d'éléments HTML pouvant y faire appel. Il vous faut les inventer.

Comme il y aura sans doute plusieurs, voire beaucoup de boîtes CSS (des « *div* ») dans une page donnée, il faut leur trouver des noms. Pour ce faire, on leur attribue des « identités », qu'on appelle *id* en abrégé. Une *id* fournit à une boîte CSS une identité qui lui est propre, qui vous permet à vous de bien la repérer et au navigateur de l'afficher correctement.

Prenons un exemple simple : vous voulez diviser une page en trois parties horizontales. Un « en-tête » en haut présente un logo et les liens de navigation. La partie du milieu contient le texte et les images. Tout en bas, vous pouvez aussi avoir un « pied de page » donnant des informations de copyright, et peut-être une répétition de la navigation du haut de page.

Dans [cet exemple \(http://www.pompage.net/IMG/html/page\\_modele11.html\)](http://www.pompage.net/IMG/html/page_modele11.html), plus précisément, vous trouverez cinq boîtes horizontales, les deux supplémentaires servant pour le logo et la barre de navigation, entre l'en-tête et la zone de texte.



Remarquez que la colonne de droite est plus étroite que celle de gauche, ce qui permet de compenser visuellement l'espace supplémentaire dû à l'aspect crénelé de la partie droite du texte, sans quoi celui-ci paraîtrait décentré.

Dans les définitions de style, nous collons un signe dièse (#) devant le nom que nous voulons donner à la boîte, ce qui en fait une « identité ».

```
#en-tete {...}
#navigation {...}
#logo {...}
#milieu {...}
#pied-de-page {...}
```

Puis nous devons lier les définitions aux boîtes CSS elles-mêmes, tout simplement en renvoyant à leurs identités (*id*) comme ceci :

```
<div id="en-tete">...</div>
<div id="navigation">...</div>
<div id="logo">...</div>
<div id="milieu">...</div>
<div id="pied-de-page">...</div>
```

Si l'on a déjà indiqué, par exemple, une police d'écriture dans la déclaration de styles pour la partie *body*, alors nos boîtes vont « hériter » de ces styles. Mais toute définition de style ajoutée aux définitions des boîtes prendra le pas sur la définition du *body*, qui est plus large. C'est le principe même de la « cascade ».

## Qu'est-ce qu'une *class* ?

Vous aurez parfois besoin d'utiliser la même boîte plusieurs fois sur une même page. Imaginons que vous voulez séparer la partie du milieu en deux boîtes distinctes, ou plus. Dans ce cas, il vous faut une boîte « réutilisable » et non pas unique. Une boîte réutilisable s'appelle une *class*. Tout comme on a des « classes » de fleurs ou d'insectes, une *class* désigne une même boîte mais avec un contenu différent. Pour montrer que nous voulons une *class* réutilisable au lieu d'une *id* unique, nous utilisons le point (.) au lieu du signe dièse.

`.milieu {}` peut désormais être utilisé aussi souvent qu'on le veut, et sur la *div* elle-même, on précise :

```
<div class="milieu">...</div>
<div class="milieu">...</div>
<div class="milieu">...</div>
```

## Qu'est-ce qu'un *span* ?

Les *class*, contrairement aux *id*, peuvent s'appliquer au niveau du texte. Si on veut simplement mettre quelques mots en rouge, on peut créer une *class* spéciale, appelons-la `.texterouge` :

```
.texterouge { #c00 }
```

```
<span class="texterouge">Ce texte est rouge!</span>
```

Au lieu d'entourer le texte que l'on veut en rouge d'un `<div>...</div>`, on utilise `<span>...</span>`. On dit que ce style est *inline*, c'est-à-dire *en ligne*, parce qu'il s'applique à un morceau de texte lui-même déjà contenu dans un `<div>...</div>`.

Utilisez `<span>...</span>` dès que vous voulez marquer sur du texte une variation par rapport aux indications générales. Il peut s'agir de couleur, de police d'écriture, de taille, de poids, etc.

Donc, en CSS, il y a des éléments dits **block** qui sont constitués de boîtes rectangulaires, dotées soit d'*id* uniques, soit de *class* réutilisables. Et nous avons aussi des éléments dits **inline**, qui traitent le texte au caractère près.

On commence à aborder les choses sérieuses !

## Neuvième étape : on met tout ensemble

Dès qu'on aborde des pages Web un peu élaborées, on utilise un mélange d'*id* et de *class*, voire de positionnement absolu et relatif. Retournons à notre page toute simple de la huitième étape, [celle en cinq parties \(http://www.pompage.net/IMG/html/page\\_modele11.html\)](http://www.pompage.net/IMG/html/page_modele11.html) : peut-être pourrait-on avoir plusieurs colonnes dans la partie du milieu. Cela revient à utiliser la propriété de flottement (`float`), et je vais choisir `float:left` pour m'en tenir au plus simple.

En général, sur beaucoup de pages Web, la colonne de gauche contient les menus, ce qui veut dire qu'elle peut être assez étroite.

```
.col-gauche {}
```

La colonne du milieu contiendra le texte principal.

```
.col-milieu {}
```

Et tout à droite, nous allons placer une autre colonne étroite, pour des liens de référence ou de la publicité.

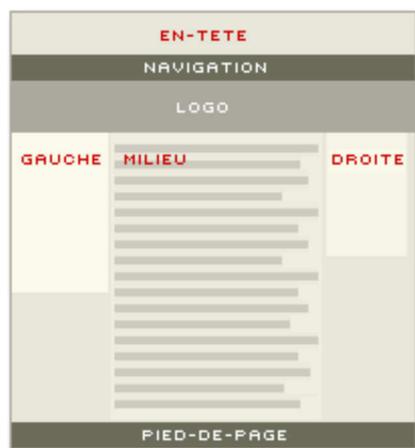
```
.col-droite {}
```

La taille de chaque colonne est indiquée en pourcentage, pour qu'elle soit fluide et s'adapte à la largeur de la page.

Vous noterez que j'utilise des classes (`class`) pour ces colonnes, parce que je pourrais souhaiter les réutiliser plus bas sur ma page.

Dans [cet exemple \(http://www.pompage.net/IMG/html/page\\_modele12.html\)](http://www.pompage.net/IMG/html/page_modele12.html), on remplace la simple partie `.milieu` par trois colonnes flottant à gauche.

```
<div class="col-gauche">...</div>
<div class="col-milieu">...</div>
<div class="col-droite">...</div>
```



Voilà qui ouvre des perspectives de mise en page beaucoup plus intéressantes.

Ne laissez pas de `div` complètement vides, même si elles ne doivent contenir que du « blanc ». Placez-y un espace insécable `&nbsp;`.

## Dixième étape : Doctypes et validation

Nous voilà arrivés à la dixième leçon de « On reprend tout à zéro », et il faut que je vous avoue quelque chose. Tout ce que je vous ai montré jusqu'ici, le balisage, les pages d'exemples, eh bien tout ceci est faux ! Bon, en fait, ça va sans doute fonctionner correctement (sauf si vous utilisez un navigateur trop ancien), mais si vous testez vos pages avec un vérificateur de syntaxe, ou bien si vous les soumettez à un quelconque programme de validation, elles ne passeront pas l'épreuve.

Vous vous demandez peut-être ce qu'est un programme de validation.

Vous connaissez sûrement les correcteurs d'orthographe. Votre programme de traitement de texte en contient vraisemblablement un. Ils servent à comparer les mots que vous avez tapés avec ceux de leur dictionnaire intégré, et vous avertissent s'ils ne trouvent pas de concordance. Peut-être le mot n'est-il pas dans le dictionnaire, mais peut-être aussi ne l'avez-vous pas tapé correctement. Dans les traitements de texte les plus élaborés, votre texte peut également subir une

vérification grammaticale, et si vous tapez une phrase qui ne contient pas de verbe ou qui contient trop d'occurrences du mot « et », ils vous harcèleront sans arrêt.

Pour qu'un correcteur d'orthographe ou de grammaire puisse faire son travail, il doit savoir quelle langue vous utilisez. Un correcteur d'anglais britannique est déjà différent d'un correcteur d'anglais américain, alors que dire de ceux qui utilisent une toute autre langue que l'anglais ?

De même qu'on corrige son anglais, son français, ou toute autre langue, on peut (et on doit !) corriger également son HTML et son CSS. Si votre anglais contient une erreur, les gens penseront qu'il s'agit d'une coquille, mais comprendront ce que vous voulez dire, il n'y aura pas grand-mal. En revanche, une petite erreur en HTML ou en CSS, c'est une toute autre histoire. Quelque chose d'aussi insignifiant en apparence qu'une virgule ou un guillemet mal placés peut faire toute la différence entre une page qui fonctionne ou non : tout dépend du type d'erreur et du navigateur qui essaie de comprendre ce qui se passe. Il est très recommandé de « vérifier l'orthographe » de votre HTML comme de votre CSS avec un « correcteur de syntaxe » ou un « validateur ».

L'un des grands avantages dont vous bénéficiez si vous utilisez un éditeur HTML de bonne qualité, c'est qu'il contient un correcteur de syntaxe intégré. Vous ne trouverez pas cela avec de simples éditeurs de texte polyvalents. Si vous avez un éditeur HTML comme HomeSite ou BBEdit, ou bien un éditeur WYSIWYG tel que Dreamweaver ou GoLive, vous pourrez vérifier votre balisage et si possible trouver des suggestions pour en corriger les erreurs. Si vous ne possédez pas de tels programmes, vous pouvez utiliser les validateurs en ligne offerts par le W3C : le validateur HTML (<http://validator.w3.org/>) et le validateur CSS (<http://jigsaw.w3.org/css-validator/>) du W3C, où vous pouvez tout simplement envoyer vos fichiers et recevoir un rapport immédiat indiquant la liste exhaustive de vos erreurs, sauf si vous n'en avez commis aucune !

Bien entendu, le programme de correction doit d'abord savoir quel « langue » vous utilisez pour votre balisage : il y a différentes « gammes » de HTML. Jusqu'ici, tous les exemples que j'ai montrés sont écrits en HTML 4.01 : il s'agit de la version actuelle, et c'est la plus répandue. Mais il y a différentes versions de HTML 4.01 : « Strict », « Transitional » et « FrameSet ».

Le **Strict** est constitué d'un ensemble de règles qui définissent très strictement ce que l'on peut faire ou ne pas faire en HTML.

Le **Transitional** est un peu plus compréhensif, et vous permet d'utiliser un HTML un peu plus ancien ainsi que quelques particularités propres à certains navigateurs.

Le **Frame Set** n'est plus beaucoup utilisé de nos jours. Nous n'avons d'ailleurs pas évoqué les « frames » (cadres) dans cette série de cours, dans la mesure où ils sont largement passé de mode depuis l'arrivée des CSS.

La plupart des gens se contentent très bien du « Transitional » parce qu'il leur laisse un peu de souplesse, mais il y en a d'autres qui aiment que tout soit bien net et règlementaire et qui préfèrent utiliser le « Strict ».

Pour que le navigateur sache quel langage vous utilisez pour votre balisage, et dans quelle version, placez une déclaration de Doctype tout en haut de la page, au-dessus de la première balise <html>. En voici un exemple :

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

Je ne vais pas analyser tout de suite tout ce que cela signifie ; contentez-vous de savoir qu'avec ce charabia au sommet de votre page, le vérificateur de syntaxe ou le validateur sera capable de faire correctement son travail. Il sait désormais quels sont les standards que vous voulez appliquer à votre page, et va pouvoir la juger en fonction de ceux-ci.

Dès lors que j'ai placé un Doctype, je peux passer mes pages au vérificateur de syntaxe de mon éditeur HTML et m'assurer qu'elles sont impeccables. Maintenant qu'on a réglé ce petit détail annexe, revenons au style de notre page !

## Onzième étape : déclarer les styles

Jusqu'ici, tous les styles que j'ai utilisés ont été intégrés à la section <head> des pages. De cette façon, il est facile de

regarder le code-source de la page et de voir ce qui s'y passe. Tout est bien mis ensemble au même endroit.

Toutefois, il est très rare que des pages individuelles soient fabriquées à l'unité : il est beaucoup plus probable qu'on trouvera, dans un site Web donné, tout un ensemble de pages dont l'aspect est similaire.

Au lieu de reproduire une déclaration de styles sur chaque page, il vaut bien mieux n'avoir qu'une seule feuille de style et faire en sorte que chaque page y renvoie individuellement. Du coup, si vous voulez changer la couleur de fond de toutes les pages de votre site, ou bien modifier la couleur ou la taille des caractères, il n'y aura qu'à changer la feuille de style principale. En fait, on peut changer totalement l'aspect de tout un site en appliquant juste quelques modifications à un seul et unique fichier.

Allez donc jeter un oeil sur <http://www.csszengarden.com/> (<http://www.csszengarden.com/>) pour voir ce principe en action. Différents designers ont fourni leurs propres feuilles de style pour habiller le contenu d'une même page, démontrant ainsi de manière admirable comment un seul et même squelette peut recevoir différentes « peaux » qui changent sa personnalité du tout au tout. Pour utiliser une feuille de style externe, tout ce que nous avons à faire consiste à intégrer ce type de lien dans notre page :

```
<link rel="stylesheet"
href="styles_de_base.css"
type="text/css"
media="screen">
```

Cela indique au navigateur qu'il doit chercher les indications de style dans un fichier nommé « `styles_de_base.css` ». On précise aussi qu'il s'agit d'un simple fichier texte ( d'où `type="text/css"` ) et que les styles sont prévus pour un affichage sur écran ( `media="screen"` )

Les feuilles de style ne sont pas limitées à l'affichage sur des écrans d'ordinateur : vous pouvez produire des feuilles de style pour d'autres supports comme l'impression, la projection ou même le braille. Quand on y réfléchit bien, une feuille de papier d'imprimante n'a pas du tout la même taille ni la même forme qu'un écran d'ordinateur, et personne n'aura envie de gâcher sa très chère encre à imprimer de vastes zones colorées alors que tout ce qu'on veut, c'est conserver comme référence un texte noir sur blanc. Une feuille de style pour l'impression doit donc réduire la largeur de la page pour qu'elle corresponde à une feuille de papier, et contrairement à la feuille de style « écran », les tailles de caractères doivent être indiquées en points ( pt ). Les couleurs doivent être simplifiées, voire complètement abandonnées. Enfin, tout ce qui restera à faire sera d'ajouter un second lien dans la partie `<head>` juste après celui pour l'écran ( `screen` ).

```
<link rel="stylesheet"
href="styles_pour_impression.css"
type="text/css"
media="print">
```

Si vous vous contentez amplement de la même feuille de style pour l'écran comme pour l'impression, alors vous pouvez utiliser ce lien :

```
<link rel="stylesheet"
href="styles.css"
type="text/css"
media="all">
```

Bien que la plupart de mes feuilles de style finissent toujours par se retrouver enregistrées à part, je démarre généralement avec une feuille de style interne pour les premières étapes du design. Une fois satisfait de l'aspect général, je coupe et je colle les styles dans un nouveau fichier, et je les remplace par un lien.

## Douzième étape : les tableaux

Je sais bien qu'on utilise des tableaux pour réaliser des mises en page depuis la préhistoire du Web, mais je pense avoir démontré dans ce feuilleton que les mises en page par [CSS](#) offrent davantage de possibilités et de souplesse. Les tableaux sont faits pour présenter des données tabulaires, comme on le ferait avec les feuilles de calcul qu'on utilise pour organiser des chiffres ou des mots en rangs et en colonnes.

Autrefois, avec les anciens navigateurs, il était possible de changer l'aspect des tableaux et de leurs cellules en utilisant les différentes propriétés de tableaux : couleurs de fond, images de fond, etc. Beaucoup de ces caractéristiques sont désormais dépassées et ne sont plus valides ; même s'il est encore possible qu'elles fonctionnent, on ne peut plus compter sur elles. D'ailleurs, on n'a jamais vraiment pu compter dessus. Quand on cherchait à placer une image de fond pour un tableau sous Netscape 4 ou une version antérieure, celle-ci apparaissait individuellement dans chacune des cellules, un peu comme quand on regarde à travers un kaléidoscope, ce qui rendait le tout aussi pénible à faire qu'inutile.

Avant, on pouvait sans problème indiquer une hauteur de tableau en pixels, ou encore en pourcentage de la hauteur de la fenêtre, mais ce n'est plus possible. Les tableaux fixent leur propre hauteur en fonction de leur contenu.

La gestion du style des tableaux avec les [CSS](#) peut se révéler un peu délicate, et il est difficile de tout prévoir. On peut construire sa propre structure de tableau en CSS à l'aide de `display:table;` et de `display:table-cell;`, en considérant que chaque cellule constitue une boîte (`div`) séparée, mais dans la plupart des cas, il est plus simple de se contenter de fournir les définitions de style pour les éléments [HTML](#) du tableau : `table`, `tr`, `td`, `th`, etc.

`tr` représente une rangée de tableau, c'est-à-dire une rangée horizontale de cellules. `td` est une division de rangée de tableau. On peut le voir comme une colonne, mais il ne se comporte pas comme une colonne, et on ne peut pas indiquer une couleur de fond ni un style d'écriture propre à une colonne.

Les tableaux héritent de certains styles des `divs` auxquelles ils appartiennent, mais pas de tous. Si la partie `body` de votre page a pour caractéristiques `font-family: verdana, arial, sans-serif style` et `color:#009`, le tableau héritera de celles-ci, mais pas de la taille du texte ni de son alignement. Il y a également une hiérarchie. Toute règle s'appliquant à un `td` prendra le pas sur celle du `tr`, qui elle-même passera avant toute règle générale de style imposée pour l'élément `table`, c'est-à-dire le tableau. Vous pouvez tourner ceci à votre avantage. Si vous choisissez le noir comme couleur de fond de tableau, et le blanc comme couleur de fond du `td`, et si vous donnez au tableau un écartement des cellules de 2 pixels (`cellspacing="2"`), vous obtiendrez un tableau avec des cellules blanches séparées par une bordure noire de 2 pixels.

1	2	3	4
1	2	3	4
1	2	3	4

Le tableau 1 ci-dessus, plus subtil, présente une couleur de fond de tableau transparente, et une légère teinte de couleur de fond est appliquée aux `td`.

Bien entendu, vous avez également accès à tous les styles CSS pour les bordures, ce qui signifie que vous n'êtes pas prisonnier des horribles bordures de tableau par défaut en HTML. Vous pouvez également jouer avec les couleurs ou les aspects des bordures pour n'afficher que des lignes verticales ou des horizontales, qui seront pleines, pointillées, hachurées ou autres.

1	2	3	4
1	2	3	4
1	2	3	4

Tableau 2. Selon les données, on veut parfois insister sur les rangées horizontales, ou bien sur les colonnes verticales comme ci-dessus.

Titre 1	Titre 2	Titre 3	Titre 4
1	2	3	4
1	2	3	4

Tableau 3. Ici un effet de relief en 3D est créé en modifiant les couleurs de bordures. L'en-tête de tableau `th` reçoit une couleur de fond plus sombre.

Une autre idée intéressante consiste à attribuer aux `tr` des images de fond en dégradé de couleur.

Si vous créez une petite image suffisamment haute pour remplir la cellule tout en restant très étroite, disons 8 pixels de large, il sera possible de la répéter horizontalement avec `background-repeat: repeat-x`, ce qui est plus fin et élégant que des bordures.

 Cette petite image (227 octets) peut être répétée horizontalement...

<b>Rangée 1</b>	2	3	4
<b>Rangée 1</b>	2	3	4
<b>Rangée 1</b>	2	3	4
<b>Rangée 1</b>	2	3	4

Tableau 4. J'ai utilisé un dégradé légèrement plus sombre pour les en-têtes de rangées, et j'ai ajouté une bordure gauche gris pâle pour séparer les colonnes.

Si vous voulez appliquer aux cellules individuelles un style différent de l'ensemble du tableau, vous pouvez le faire en utilisant des définitions de style « en ligne » adjointes aux balises `<td>` appropriées.

```
<td style="background-color: #f00">cellule 1</td>
```

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

Tableau 5. Bon d'accord, je suis un peu trop fan de Mondrian, mais l'important c'est de voir qu'on peut appliquer à volonté des styles sur des cellules individuelles !

Cette approche, en revanche, a le désavantage de ne pas permettre l'utilisation d'une feuille de style externe. Tout est gravé dans le code même de la page ; mais on peut se l'accorder, de manière exceptionnelle. Une façon plus adroite de styler rangées et colonnes consiste à les placer au sein de leurs propres `div` et à les nommer ainsi :

```
#colonne1 td { ... }  
#colonne2 td { ... }
```

Avec cette méthode, vous bénéficiez d'un certain contrôle sans pour autant devoir recourir à une construction de tableau par `div` individuelles (ce qui reste une excellente solution, si vous en avez le temps et la patience).

Je n'ai pas placé toutes les déclarations de style dans la page, il y en a trop, mais un coup d'oeil sur le code-source vous permettra de comprendre ce qui se passe.

De nombreuses possibilités vous sauteront à l'esprit quand vous commencerez à styler vos tableaux. Mais non, ce n'est pas obligatoirement barbant de faire des tableaux !

Le mois prochain, nous verrons comment styler des formulaires, ainsi que quelques autres petits détails.

## Treizième étape : les formulaires

Pour de nombreux designers, le style des formulaires dans les pages Web ne constitue pas une grande priorité. Les formulaires sont généralement laissés « tels quels », et pour de bonnes raisons d'ailleurs.

- I. Les conventions qui sont utilisées par défaut pour les formulaires fonctionnent assez bien en l'état.
- II. Si un formulaire n'a pas l'aspect attendu, les lecteurs pourraient ne pas s'y retrouver.
- III. Le style des formulaires peut devenir très différent d'un navigateur à l'autre, selon l'interprétation qu'ils en font.

Voyons tout cela de plus près.

## « Les conventions qui sont utilisées par défaut pour les formulaires fonctionnent assez bien en l'état. »

Bon, peut-être. Pourtant on trouve encore un peu partout beaucoup de tableaux mal conçus, que ce soit sur papier ou sur le Web.

Moi en tous cas, je déteste remplir des formulaires, et je sais que je ne suis pas seul à réagir ainsi.

Mais les formulaires sont bien pratiques pour récolter de l'information, d'autant qu'ils la récoltent d'une manière qui en rend l'exploitation assez facile. Alors, quelle est l'erreur commise par les « designers » de formulaires ?

Le premier gros écueil apparaît lorsque le formulaire **semble** compliqué. L'utilisateur est intimidé et se place en position défensive.

Et puis, on nous pose des questions qui peuvent paraître soit inutiles, soit indiscrètes. Pourquoi voulez-vous absolument connaître ma date de naissance ?

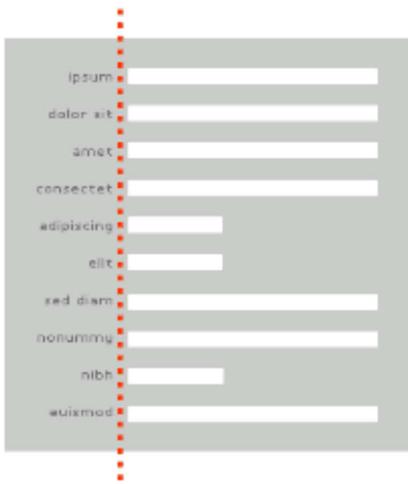
L'aspect compliqué est une question de mise en forme. On peut donner à un formulaire une apparence simple en le présentant de manière propre et ordonnée. La première chose dont on doit s'assurer, c'est de bien contrôler les tailles de police pour ne pas les laisser démolir la mise en page.

La taille et la position des légendes ou des textes explicatifs doivent aussi être réfléchies. Le mieux est de les placer soit au-dessus ou en dessous des champs de saisie, soit sur la droite. Si vous tenez vraiment à mettre des textes sur la gauche, utilisez `align: right;` pour éviter de vous retrouver avec d'énormes trous qui déconnectent visuellement le texte de son champ, et augmentent la complexité visuelle de la mise en page. Essayez également de ne pas avoir trop de tailles différentes pour vos champs de saisie, et ne remplissez pas les espaces vides à tout prix. Si le champ du code postal est plus court que les autres champs d'adresse, ce n'est pas un problème : ne placez pas pour autant différentes tailles de champs sur une même ligne horizontale juste pour gagner de l'espace.

Sur une page Web, on n'a pas besoin de gagner de l'espace. Choisissez intelligemment les tailles de vos champs de saisie. Alignez-les verticalement. Gardez-les propres et dégagés.

Le diagramme illustre un formulaire mal conçu. Il est composé de dix lignes de champs de saisie. Les champs sont représentés par des rectangles blancs. Les légendes (textes explicatifs) sont placés à gauche de chaque champ. Les légendes sont : 'ipsum', 'dolor sit', 'amet', 'consectet', 'adipiscing', 'elit', 'sed diam', 'nonummy', 'nibh', et 'euismod'. Les champs de saisie sont de différentes tailles et sont mal alignés. Les légendes sont alignées à gauche, mais les champs de saisie ne sont pas alignés à droite. Il y a des espaces vides entre les champs et les légendes. Des lignes verticales pointillées (une bleue à gauche, une orange à droite) indiquent des axes de référence. Des lignes horizontales pointillées (une violette) indiquent des alignements incorrects. Les légendes 'adipiscing' et 'nonummy' ont des mots ('autem' et 'iriure') placés à droite de leurs champs respectifs.

Ci-dessus, un formulaire qui semble compliqué. Il y a deux axes verticaux, et le déplacement du regard est perturbé par la nécessité de sauter horizontalement et verticalement pour trouver le champ suivant.



Voici un design plus simple, avec un seul axe vertical en plein milieu, ce qui n'interrompt pas le déplacement du regard.

L'avantage par rapport à un formulaire papier, c'est qu'un formulaire Web masque tout ce qui n'est pas immédiatement visible dans la fenêtre du navigateur, évitant ainsi de noyer le visiteur sous la masse. Néanmoins, un formulaire très long sur une seule page exigera de nombreux défilements verticaux, ce qui peut être encore pire. Mieux vaut diviser les formulaires longs sur toute une série de pages, à condition toutefois de penser à permettre aux utilisateurs de revenir en arrière s'ils veulent modifier quelque chose.

### « Si un formulaire n'a pas l'aspect attendu, les lecteurs pourraient ne pas s'y retrouver. »

Il est clair qu'on ne peut pas « s'éclater » dans le design de formulaires. Il faut bien que le lecteur comprenne la fonction d'un champ de saisie de texte, comme pour une fenêtre pop-up ou un bouton d'envoi. Ça ne signifie pas pour autant que nous sommes condamnés au texte noir sur fond blanc (encore que certains navigateurs n'acceptent rien d'autre !)

Ce serait une erreur d'assimiler un champ de saisie à du texte ordinaire, la fenêtre doit avoir son apparence propre, tout en indiquant sa fonction.

Les éléments que l'on peut styler sont `input`, `textarea` et `select`, mais attention : tout style attribué à l'élément `input` affectera non seulement le champ de saisie de texte, mais également les autres outils de saisie que sont les boutons radio, les cases à cocher, ainsi que les boutons « Envoyer » et « Effacer ».

On peut sans risque jouer sur les styles `font-family` et `font-size` des polices. La couleur du texte (`text-color`) peut varier si on le souhaite. Pour la couleur de fond (`background-color`), c'est plus discutable. Si elle est d'une teinte plus pâle que celle de la page, c'est acceptable, et cela peut rendre l'aspect du formulaire moins agressif.

Si on change le contour d'un champ de saisie de données (« `input` »), cela modifie également le contour du bouton d'envoi, qui du coup peut se mettre à ressembler à n'importe quelle zone de texte si on n'y prend garde. Pour éviter cela, il faut donc « envelopper » le champ de saisie dans une `div` qui lui sera propre, et se contenter d'appliquer des styles sur `#saisie input` comme ci-dessous :

```
#saisie { }
```

```
#saisie input
{
  color: #633;
  font-size: 10px;
  background-color: #ebbd9;
  padding: 3px;
  border: double 3px orange
}
```

NdT : les éléments `input` et `textarea` ne peuvent être traduits, car il s'agit de code HTML. Rappelons que `input` fait référence à de la saisie de données de formulaire (un nom, une date, un chiffre...), tandis que `textarea` permet de saisir des commentaires, des remarques ou autres textes longs.

Si maintenant vous voulez qu'un champ `textarea` ressemble à un champ `input`, vous pouvez faire ceci :

```
#saisie input, #saisie textarea
{
  color: #633;
  font-size: 10px;
  background-color: #ebeb9;
  padding: 3px;
  border: double 3px orange
}
```

Quant à l'élément `select` (il s'agit des listes déroulantes), son rendu varie d'un navigateur et d'une plateforme à l'autre. Tout ce que nous pouvons faire, c'est fixer une taille et une famille de police. De toute manière, vous ne pouvez généralement avoir aucune idée de l'aspect final, une fois qu'ils auront été cliqués.

## « Le style des formulaires peut devenir très différent d'un navigateur à l'autre. »

La meilleure des preuves réside dans ces quelques copies d'écran ([http://www.pompage.net/IMG/html/tests\\_formulaires.html](http://www.pompage.net/IMG/html/tests_formulaires.html))

Les mêmes styles interprétés sous Explorer (Windows), Mozilla, Safari (Mac) et Opera montrent quelle variété on peut trouver. Je reviendrai dans le prochain épisode sur les différences entre les interprétations des navigateurs.

L'important ici est que nous ne cherchons pas à enjoliver les formulaires. Nous essayons de les rendre plus faciles à utiliser et moins impressionnants grâce à une présentation visuelle réfléchie.

On peut encore faciliter davantage la vie de l'utilisateur avec des scripts de validation de formulaires qui sont bien utiles, mais c'est une tout autre histoire, dans laquelle je ne compte pas m'engager tout de suite.

## Quatorzième étape : le navigateur

Ah, les navigateurs. Que ferions-nous sans eux ? Et combien de fois ai-je entendu les gens regretter qu'il n'y en ait pas un et un seul ? Oui, mais lequel ?

Chaque navigateur possède sa propre feuille de style par défaut. C'est celle que vous utiliserez, à moins de fournir la vôtre. Et bien entendu, les feuilles de style intégrées sont toutes différentes. Elles sont issues de différentes sociétés, et fonctionnent différemment sur les diverses plate-formes informatiques.

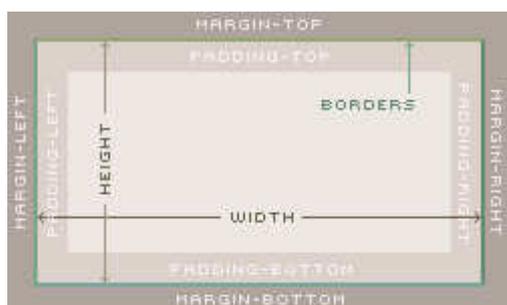
Leurs ressemblances sont nombreuses, bien sûr, mais les différences ne sont pas négligeables pour autant. Si vous partez du principe que tous les autres navigateurs vont offrir un rendu « similaire » de votre page Web si amoureusement conçue, vous allez vite vous apercevoir que « similaire » ne signifie pas « identique », et que cela peut même signifier « franchement différent » dans certains cas - même si votre feuille de style CSS est valide.

Je ne peux pas me lancer ici dans une comparaison exhaustive de tous les navigateurs, d'autant qu'il en existe énormément qui sont déjà publiés ailleurs (<http://www.google.com/search?hl=en&lr=&ie=ISO-8859-1&oe=ISO-8859-1&q=diff%E9rences+navigateurs+CSS&btnG=Search>). Je soulignerai simplement les quelques points qui me posent le plus de problèmes. Généralement, cela revient à une opposition du style Internet Explorer - Reste Du Monde, mais pas toujours. Une liste de liens vers des navigateurs récents est toujours disponible sur la page d'infos de mon site [WPDFD](http://www.wpdfd.com/index.htm) (<http://www.wpdfd.com/index.htm>). Ils sont gratuits pour la plupart, ou peuvent être essayés gratuitement.

Internet Explorer de Microsoft est le navigateur le plus répandu de tous. Sous ses différentes versions, il représente quelque chose comme 85% de tous les navigateurs utilisés. Malheureusement, il n'obéit qu'à ses propres règles et ne se conforme pas aux mêmes normes que d'autres navigateurs. Il en respecte la plupart, mais dans le cas contraire, les problèmes sont inévitables.

## Le modèle de boîtes CSS

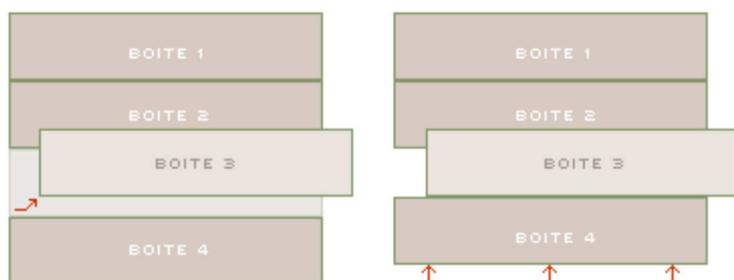
Dans toutes les versions d'IE (NdT : à l'exception du mode strict d'IE6, dont le fonctionnement dépasse largement le périmètre de cet article), l'interprétation du remplissage ou « `padding` » est différente de celle des autres navigateurs. En conséquence, si vous indiquez une largeur de boîte CSS en pixels, et que celle-ci est dotée d'un padding droit ou gauche, la boîte aura une largeur différente entre IE et les autres navigateurs, ce qui peut détruire une mise en page quand elle est très précise.



IE part du principe que le padding est situé à l'intérieur de la boîte, réduisant d'autant la largeur du texte (ce qui d'ailleurs me semble tout à fait logique). Les standards du W3C disent, eux, que le padding doit être ajouté en dehors de la boîte, pour maintenir identique la largeur du texte. On peut débattre *ad nauseam* sur ce qui est le plus logique des deux, mais quoi qu'il en soit, vous allez être confrontés à cette différence fondamentale qui risque de vous jouer des tours si vous ne vérifiez pas vos pages.

## Tout est relatif

Le positionnement « relatif » des boîtes CSS semble être un concept assez simple. Chaque boîte occupe une place que l'on définit relativement à celle qui la précède immédiatement. Mais attention ! Si vous sortez une boîte de son « flux naturel » en lui donnant une valeur par rapport au haut ou au bas de page (valeur `top` ou `bottom`), cette boîte est supposée se déplacer tout en laissant libre sa place initiale, comme le ferait une voiture quittant sa place de parking. Mais IE (version Mac) referme cet espace dès qu'il est libéré, remontant donc d'autant tout ce qui est situé au-dessous.



On voit à gauche ce qui doit se produire quand on déplace une boîte CSS relative : l'espace libéré devient de fait une boîte invisible. À droite, on voit ce qu'IE (Mac) en fait : cela peut mettre à mal une présentation si on n'y prend garde.

## Attention au format du balisage

Ça, c'est quelque chose qui m'agace vraiment, parce qu'il s'agit bel et bien d'un bogue, et non d'une mauvaise interprétation des standards. En principe, les navigateurs doivent ignorer complètement le format du balisage `HTML`. Peu importe si vous tapez tout votre code sur une seule longue ligne horizontale, ou bien si vous préférez avoir un interligne supplémentaire entre chaque ligne. Le bogue dans IE (Mac) est ainsi fait que si vous placez des boîtes en `float` sur une seule ligne, flottant toutes

vers la gauche, mais qu'en même temps vous tapez le balisage correspondant sur plusieurs lignes de code distinctes, votre navigateur perd complètement les pédales. Il faut *obligatoirement* placer tout le balisage sur une seule ligne pour obtenir le résultat escompté. Franchement, ce ne serait pas un travail de Titan pour réparer ce bogue, mais le développement d'IE (Mac) est mort et enterré, et il existe aujourd'hui un tel choix de meilleurs navigateurs... mais les vieux Macs, c'est comme les vieilles habitudes, on ne s'en débarrasse pas comme ça !

## L'avenir

Les spécifications et les possibilités des Feuilles de Style en Cascade (*Cascading Style Sheets*) sont sans cesse actualisées, et les navigateurs également s'adaptent continuellement pour pouvoir faire écho à ce nouveau potentiel. C'est un processus en déroulement, toujours en évolution, que pilotent les membres du W3C (le *World Wide Web Consortium*).

On peut espérer que les nombreuses différences d'interprétation que nous subissons actuellement finiront par être réduites, sinon éliminées, mais évidemment il y a d'autres facteurs en jeu.

Le concept même de la lecture de pages Web sur un écran d'ordinateur est appelé à changer. On peut déjà voir des pages Web sur des machines qui ne sont pas des ordinateurs, et du contenu transmis par le Web mais qui n'est pas visible par les navigateurs.

Et ce n'est pas fini !

Les entreprises s'acharnent à tenter d'identifier des moyens pour intégrer des fonctions d'interactivité à leurs produits ; les téléphones, les télévisions, les voitures, les réfrigérateurs, les fours à micro-ondes sont tous des cibles privilégiées. Au bout du compte, l'implant cérébral connecté au Web nous reliera tous ensemble, permettant ainsi à la race humaine de parvenir, pour la communication, au même niveau de perfection que les fourmis !

Avec les CSS, il importera peu de savoir sur quel support sera affiché le contenu. Elles s'adaptent déjà à une grande variété de types de médias, même si nous en utilisons essentiellement deux : « `screen` » et « `print` ».

Mais ce qui est à retenir de tout cela, c'est qu'il n'est pas question de se reposer sur ses lauriers, il faut soit avancer, soit renoncer.

J'arrive au terme de cette courte série « CSS : on reprend tout à zéro ! », et j'espère qu'elle vous aura donné l'envie d'en savoir davantage - et croyez-moi, il y en a encore beaucoup à savoir.

Voici une liste de quelques articles en anglais sur les CSS à lire sur [WPDFFD \(http://www.wpdffd.com/index.htm\)](http://www.wpdffd.com/index.htm).

### [Turning the Tables](http://www.wpdffd.com/editorial/wpd0403b.htm)

<http://www.wpdffd.com/editorial/wpd0403b.htm> Convertir en CSS des mises en page en tableaux

### [Style Sheets Without Tears](http://www.wpdffd.com/editorial/wpd0402.htm)

<http://www.wpdffd.com/editorial/wpd0402.htm> Série en 3 parties sur des feuilles de style basiques

### [CSS - Browser Support](http://www.wpdffd.com/editorial/wpd0902.htm)

<http://www.wpdffd.com/editorial/wpd0902.htm> Comment les navigateurs gèrent les CSS

### [Thinking Inside the Box \(http://www.wpdffd.com/editorial/wpd1002.htm\)](http://www.wpdffd.com/editorial/wpd1002.htm)

Les éléments de construction de base pour les CSS

### [Box of Tricks](http://www.wpdffd.com/editorial/wpd1102.htm)

<http://www.wpdffd.com/editorial/wpd1102.htm> Des effets de « rollover » sans JavaScript

### [Scripting the Box \(http://www.wpdffd.com/editorial/wpd0103.htm\)](http://www.wpdffd.com/editorial/wpd0103.htm)

La puissance des CSS associées au Javascript

### [Drawing with CSS \(http://www.wpdffd.com/editorial/wpd0104review.htm#footnote\)](http://www.wpdffd.com/editorial/wpd0104review.htm#footnote)

Comment dessiner des graphiques simples grâce aux boîtes CSS

[Film-strip Rollovers](#)

<http://www.wpdfd.com/editorial/wpd0404footnote.htm> Une méthode simplifiée pour faire des « rollovers »

[Dynamic CSS Animation](#)

<http://www.wpdfd.com/editorial/wpd0504news.htm#feature2> Comment faire bouger vos divs