Janvier 2009
Styles Team

STYLES CSS

VOS PREMIERS PAS EN CSS

Ou Comment s'y Retrouver Dans Une Feuille de Style

Nul intention de vous faire un cours particulier sur les CSS (Feuilles de Style en Cascade), loin de moi cette possibilité, mais simplement d'essayer de vous en faire comprendre le fonctionnement et de vous permettre de vous y retrouver plus aisément dans une feuille de styles CSS !

Tutorial réalisé pour **NPDS**

Générateur de Portail Français en Open-Source http://www.npds.org

Réalisation : Fred – Alias Canasson http://www.champardennes.info / http://www.clicimpact.fr

Relecture : Developpeur – Jireck – Hotfirenet

Vos Premiers Pas en CSS (Feuilles de Style en Cascade)

1. Pourquoi Utiliser les Feuilles de Style

1.1. Idée Reçue !

Contrairement aux idées reçues, réaliser son thème en CSS, n'est pas uniquement le fruit de vouloir tenir compte de certaines règles imposées ou non concernant l'accessibilité !

Avec un peu de pratique, on se rend très vite compte qu'il est beaucoup plus simple d'effectuer des mises en pages complexes avec les CSS !

C'est le langage CSS qui est spécialement conçu pour appliquer des styles et non le (X)HTML ! Les éléments (X)HTML ont des styles prédéfinis qu'il est donc possible de modifier par le biais de styles CSS !

1.2. A quoi sert une Feuille de Style

Imaginez un site contenant une centaine de pages différentes mais dont la présentation graphique est exactement la même pour garantir l'uniformité de celui-ci ! Si vos pages sont construites en (X)HTML pur et que vous décidiez de changer un élément dans un fichier (*ne serait-ce qu'une taille de police*), vous devriez modifier un par un tous les fichiers.

Avec les CSS, chaque page du site va afficher son contenu en suivant les règles de présentation définies dans un seul fichier, *la feuille de style*. Outre le fait que cela soit beaucoup plus simple lors de mises à jour, recourir aux CSS permet que le code source ne soit écrit qu'une seule fois, d'où un temps de chargement des pages plus rapide et un gain de place sur votre serveur.

Les CSS servent à gérer le contenant - Le (X)HTML sert à gérer le contenu.

2. Le Principe des Feuilles de Style

2.1. Où placer vos règles de style

Il est possible d'inclure vos styles directement dans votre document (X)Html, dans ce cas ils sont définis entre une paire de balises <style type="text/css"> et </style> placée dans l'en-tête du document entre les balises <head> et </head>. Mais là, il faudra renouveler l'opération pour chaque page !

Une autre solution consiste à placer vos styles dans un fichier séparé, nommé en .css *(style.css* par ex.*)*. Il suffira simplement d'en indiquer le chemin dans l'entête de votre document (X)Html avec une balise de la forme <link rel="stylesheet" type="text/css" href="style.css">.

2.2. De quoi se compose une feuille de Style

Pour une mise en page souple et pratique, on associe certains paramètres à certaines balises (*les éléments*), ces paramètres sont en fait constituées de règles en texte simple qui vont gérer l'aspect de ces éléments.

D'une manière générale, la feuille de Style, qui est composée d'une multitudes de règles, est la partition qui va gérer l'apparence de vos documents !

C'est là, que vous retrouverez et ajouterez toutes les actions désirées, tant au point de vue de l'apparence *(textes, images, liens, etc…)* que sur le positionnement voulu ! Tous les attributs de taille, de position, de couleurs, d'alignement (*width, height, align, font, color,*...), figurent sur la feuille de style CSS et non plus sur la page html.

2.3. Comment créer et/ou modifier une feuille de Style

Une feuille de style n'est en fait qu'un simple fichier texte portant l'extension *.css* , un simple éditeur de texte suffit donc pour modifier ou réaliser une feuille de style.

Pour une présentation plus claire et détaillée, il existe **TopStyle Lite**, un éditeur CSS gratuit et très efficace, à découvrir au bas de cette page : TopStyle Lite

3. Pour bien débuter en CSS

3.1. Comment définir les règles de mise en page

Schématiquement, on retrouve des éléments de type "**bloc**", ou <ul> *par ex.* et de type "**en-ligne**", <br> et <img> *par ex.* et contrairement à certaines idées reçues, une image, une liste, un paragraphe, etc…, peuvent être stylés sans avoir besoin d'être contenus dans un élément , il est donc inutile d'utiliser pour englober tous vos éléments !

Il est conseillé d'éviter de surcharger vos documents de inutiles.

Quelques Balises de Type Bloc	Quelques Balises de Type En-Ligne
l'élément div les titres h1, h2, h3, h4, h5, h6 le paragraphe p Les listes et éléments de liste ul, ol, li, dl, dd Le tableau table	l'élément span le lien a L'image img Les emphases em et strong La citation q

Les règles vous permettant de gérer l'aspect des éléments d'une page sont constituées de **trois parties**:

```
body { background-color: #438fc2; }
```

Le **sélecteur** (ici: "body") qui est l'élément affectée par cette règle
La **propriété** (ici: "background-color")
La **valeur** (ici: "#438fc2") qui indique la valeur à appliquer.

A Retenir :

Les **propriétés** sont suivies de deux points
Les **valeurs** sont suivie d'un point-virgule
Les **propriétés** et **valeurs** se placent entre accolades - entre **{** et **}** – elles forment un ensemble également appelé : la déclaration.

Il est également possible d'appliquer le même style à plusieurs *éléments* en séparant le nom de ceux-ci par une virgule (ici, body et h2).

```
body, h2 { background-color: #438fc2; }
```

Tout comme il est préférable de combiner les règles s'appliquant à un même *élément*, ce qui évite une duplication du code :

```
body { background-color: #438fc2; color: #000000; }
```

3.2. Le Sélecteur Class

Pour faire simple, une classe est un ensemble de règles auquel on a attribué un «*nom de classe*» qui permet de reproduire cet ensemble de règles sur n'importe quel élément d'une page. Un nom significatif vous permettra de vous y retrouver plus aisément.

Dans la feuille de style (Le nom de la classe doit obligatoirement être précédé d'un Point):

```
.nom_de_classe {
font-size: 10pt;
padding: 3px;
margin: 0px 3px 0 3px;
}
```

Appel et Affectation de cette classe dans votre fichier (X)Html (Cette fois le Point disparaît):

```
<p class="nom_de_classe"> Ici Nous appliquons notre classe à cet élément paragraphe. Mais nous pouvons de la même façon l'affecter à d'autres éléments de la même page, et cela autant de fois que vous le souhaitez</p>
```

```
<h1 class="nom_de_classe"> Ici Notre Titre h1 aura le même aspect que le paragraphe ci-contre</h1>
```

De la même manière, il est possible d'appliquer le même style à toutes les instances d'une balise (X)Html utilisées dans vos pages :

```
p {
font-size: 10pt;
padding: 3px;
margin: 0px 3px 0 3px;
}
```

```
<p>Tous les paragraphes ne comportant pas de classe spécifique (comme ci-dessus) auront l'aspect défini ci-contre</p>
```

```
h1 {
color: #0773A3;
font-weight: bold;
}
```

```
<h1>Tous les Titres h1 seront en bleu et en gras
Exceptés ceux comportant une classe spécifique
Comme dans l'exemple précédent.</h1>
```

Il ne vous aura pas échapper que dans cet exemple nous n'utilisons pas de point devant le sélecteur (**p**, **h1**) car nous utilisons directement le **Nom de la balise** (X)HTML et non l'attribut Class.

Il également possible d'appliquer des styles différents à un même élément, dans ce cas le nom de la classe sera précédé du nom du sélecteur comme ceci par exemple :

```
b.noir {
color: #000000;
font-size: 8pt;
}
```

```
<b class="noir">Noir</b>
```

```
b.rouge {
color: #0773A3;
font-size: 9pt;
}
```

```
<b class="rouge">Rouge</b>
```

```
b.vert {
color: #0773A3;
font-size: 10pt;
}
```

```
<b class="vert">Vert</b>
```

Vous rencontrerez également des «*pseudo-classe*» qui sont en fait des classes prédéfinies, et qui permettent d'affecter un style sur un élément en fonction de son état, une réaction à un événement par exemple.

Les plus utilisées étant **les pseudo-classes de lien** :

```
a:active  a:visited  a:hover
```

Contrairement aux classes, il n'est pas possible de créer ses propres pseudo-classes puisqu'elles sont prédéfinies!

3.3. Le Sélecteur ID : #id

Le *sélecteur id* quand à lui, même s'il a pratiquement la même fonction, sera plutôt utilisé à la mise en page qu'à la mise en forme de caractères. En effet, lorsqu'on applique un *id* à un élément, on lui attribue non seulement des styles mais également une identité précise. **L'id ne doit donc désigner qu'un seul et unique élément d'un même document.**

Dans la feuille de style le nom de l'id (ici, *footer_info*) est précédé du signe dièse #

```
#footer_info {
        background-color: #9FB6D5;
        text-align: center;
        width: 100%;
}
```

Dans votre fichier Html on retrouvera notre id de cette façon (sans le #):

```
<div id="footer_info">
Le contenu de ce div se verra donc appliquer les styles définis ci-dessus.
Par contre, il ne pourra pas y avoir 2 fois cet id dans la même page !
</div>
```

3.4. Les Commentaires :

Pour vous y retrouver plus facilement dans votre feuille de style, il peut-être intéressant d'y ajouter des commentaires, ne serait-ce que pour retrouver le pourquoi du comment du choix de telle ou telle solution. Pour cela, il suffit simplement de délimiter votre texte par les caractères "**/***" et "***/**" :

```
/* Ici se Trouvent mes Titres */
h1 {
color: #0773A3;/*Là la Couleur*/
font-weight: bold;
}
```

4. Le Positionnement

4.1. Comprendre le Positionnement CSS

Sujet important et très étendu, que celui du positionnement !

Le gros avantage en CSS est qu'il n'est plus besoins d'une multitude de balises pour réaliser une mise en page au pixel près, le positionnement des éléments (*bloc de texte, paragraphe, images*...) vous permet de définir votre mise en page avec précision !

Le sujet est tellement vaste que je ne saurais être exhaustif, je voudrais juste vous en faire comprendre le fonctionnement et ne saurais que trop vous conseiller de profiter des nombreuses ressources existantes sur le web !

Pour commencer, il est important de connaître la notion de «*flux normal*», qui n'est pas un positionnement à proprement parler, mais la valeur par défaut lorsqu'aucune règle n'est spécifiée.

Par défaut, votre *navigateur* va afficher votre document en suivant l'ordre dans lequel apparaissent les balises dans votre code HTML, il procède verticalement, du haut de l'écran pour aller jusqu'en bas, et horizontalement de gauche à droite, sur la totalité de l'espace disponible et nécessaire en largeur comme en hauteur, chaque élément va donc réagir en fonction des éléments qui l'entourent.

Il est important de comprendre la différence entre deux grands groupes d'éléments (déjà cités). Par défaut les éléments de type **"blocs"**, se placeront l'un en dessous de l'autre et occuperont toute la largeur disponible alors que les éléments de type **"en-ligne"** eux se placeront l'un à côté de l'autre.

Pour simplifier, on peut dire que les éléments de type "*bloc*" servent à distinguer les parties entières de texte, comme des titres, des paragraphes, des listes, etc., alors que les éléments de type "*en-ligne*", eux sont prévus pour rester dans le linéaire du texte pour l'enrichir (*lien hypertexte, emphase, renforcement, etc.*).

	Eléments de type bloc en flux normal	**Elément de type en-ligne (span) en flux normal**

CSS	.rouge {
	background-color: #ff0000;
	}
	.orange {
	background-color: #ff9900;
	}
		.rouge {
		background-color: #ff0000;
		}
		.orange {
		background-color: #ff9900;
		}

Code Html	 Premier Bloc Second bloc	Le Premier Elément le Second Elément
Résultat	Premier Bloc Second bloc	Le Premier Elément le Second Elément

Mais puisque rien ou presque n'est figé en CSS, il est bon de savoir qu'avec la propriété «**display**» il vous sera possible de modifier les propriétés d'affichage de ces éléments pour faire en sorte que des boîtes de type bloc s'affichent en-ligne et vice-versa.

Maintenant que vous avez compris ce qu'était le «*flux normal*» nous pouvons aborder le positionnement en lui-même. En fait, il existe 3 propriétés de «*position*» en CSS : *relative*, *absolute* et *fixed*, chaque type ayant ses propres règles de positionnement.

4.2. La Position Relative : {position: relative}

Ce positionnement est le plus proche du «*flux normal*», il vous permet de déplacer un élément horizontalement et/ou verticalement par rapport à ça position de référence dans ce «*flux normal*» sans que l'élément précédent ou suivant n'en soit affecté. Cette propriété qui est valable pour tous les types de balises sera principalement employée pour appliquer un décalage à un élément par rapport à sa position "*normale*" tout en gardant à l'esprit que son emplacement initial dans le «*flux normal*» ne sera pas laissé libre, en effet cette place lui reste acquise puisque cet élément n'est pas sorti du «*flux normal*».

	Elément en position relative

CSS	.orange {
	position: relative;
	bottom: 5px; /* positionné à 5 px du bas*/
	left: 5px; /* et à 5px de la gauche*/
	background-color: *#ff9900*;
	}

Code HTML	Iciun bloc en position relativen'affecte pas le reste
Résultat	Ici Une position relative n'affecte pas le reste

Comprenez que l'élément suivant se place donc par rapport à l'emplacement initialement prévu dans le «flux normal», et non par rapport à l'élément en position relative.

4.3. La Position absolue : {position: absolute}

La position absolue se détermine par rapport au coin supérieur gauche de votre page, les coordonnées étant top= 0 et left= 0. Il est à noter que dans ce cas, votre élément sera placé au même endroit de votre page quelque soit sa position dans le code Html. En effet, un élément en position absolue est dit "*positionné*", il est retiré du flux normal et sa position est déterminée par référence à un autre élément lui-même positionné, à défaut il sera positionné par rapport au coin supérieur gauche de votre page, et non pas en décalage de la position théorique qu'il occuperait dans le flux normal. Ce mode de positionnement est à considérer fortement!

Contrairement au positionnement relatif, un élément positionné en absolu ne gardera pas sa place initialement prévue dans le «*flux normal*», cette place devient donc libre.

Comprenez que l'élément suivant se placera donc dans le «flux normal» de votre page sans tenir compte de cet élément placé en absolu !

4.4. La Position fixed : {position: fixed}

Ce mode de positionnement est plutôt considéré comme une variante du positionnement absolu, de la même manière, l'élément en position fixed est dit "*positionné*", c'est-à-dire qu'il est retiré du flux normal. Mais il est cette fois positionné uniquement par rapport aux limites de la zone de visualisation, autrement dit la fenêtre du *navigateur*. Le défilement de la page n'a aucun effet sur un contenu en position fixed.

4.5. La Propriété float : {float: left} ou {float: right}

Un élément en position *float* est lui aussi retiré du flux normal du document mais il va prendre place le plus à gauche ou le plus à droite (*en fonction de la valeur left ou right que vous lui attribuez*) du bloc qui le contient.

Mais ce positionnement aura des influences différentes sur les éléments suivants, selon qu'ils fassent parti du même conteneur ou non, c'est pour cela qu'il est souvent préférable d'utiliser ce positionnement sur des éléments faisant parti du même conteneur. En fait, un élément positionné en float, même s'il est considéré comme retiré du flux aura des interactions sur le flux courant! Très utile par exemple pour positionner du texte autour d'une image.

	Elément (image) flottant à droite

CSS	.image { /* notre maison */
	float: right;
	}
	.orange {
	width: 90%;
	background-color: #ff9900;
	}

Code HTML	 Ici un Texte beaucoup plus long ! Pourquoi pas même encore plus long ?
Résultat	Ici un Texte beaucoup plus long !  Pourquoi pas même encore plus long ?

Dans certains cas, dans un éléments (qui peut être lui-même flottant) placé directement à la suite d'élément flottant il vous sera utile d'utiliser la propriété clear avec la valeur both (clear : both). Cette déclaration à pour effet de mettre fin au flottement et donc de vous assurer que votre élément sera bien placé en dessous des autres !

5. Imbrication : Notion de Parents & Enfants

Vous l'aurez compris, chaque propriété a ses propres règles de positionnement mais, et oui il y a un mais, il faut également tenir compte de l'imbrication des éléments les uns dans les autres ! Bah Oui, vous étiez prévenu, le *Positionnement* est un vaste sujet.

Même si c'est un peu poussé, les styles CSS pourraient être comparé au code génétique d'une famille, il y a donc lieux de tenir compte d'un principe de parenté et d'héritage.

L'idée est qu'un élément X contenu dans un élément Y hérite des propriétés de ce même élément Y, on considère qu'un élément *Enfant* reçoit en héritage tous les styles de son élément *Parent*.

Pratique me direz-vous, cela nous évite de répéter inutilement nos règles de styles, en plus ça ne s'arrête pas là, puisque même les *Enfants* de cet *Enfant* hériteront à leur tour des même règles et ainsi de suite !

Le seul *Hic*, et oui il en fallait un, c'est tout simplement que toutes les propriétés ne peuvent pas être héritées, par exemple un élément de type «*en-ligne*» n'ayant pas de marges, n'héritera pas des marges de son conteneur de type «*bloc*», tout comme deux éléments contenus dans un élément positionné en float n'hériterons pas de cette propriété.

> Concrètement, si vous attribuez une propriété «color : black» à votre body, tous vos textes seront en noir exceptés ceux pour lesquels vous aurez redéfini une nouvelle propriété color !

De la même manière, les *pseudo-classes* hériterons des propriétés de leur parent, ainsi les pseudo-classes, *link* ou *hover* hériterons des propriétés appliquées à la balise parent <a> si elles ne sont pas re-définies.

6. Notion de Bloc : Marges et Remplissage

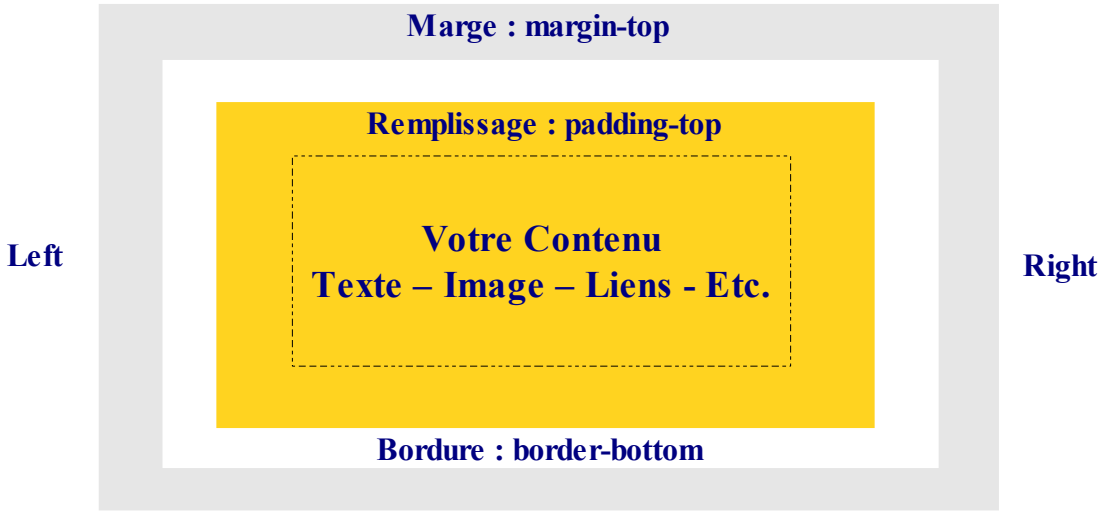
6.1. Marges, Ajustement et Bordures

Nous avons déjà évoqué à plusieurs reprises qu'ils existaient des différences entre un élément de type «*bloc*» et un élément de type «*en-ligne*», ce dernier étant plutôt utilisé pour enrichir vos textes ne possède pas de dimensions propres, au contraire de l'élément de type «*bloc*» qui lui possède ses propres dimensions, ce qui lui confère la possibilité d'être facilement positionnable (*cf. le Positionnement*).

Ce sont ces éléments de type «*bloc*», très facilement comparables à de multiples divisions de votre page, qui permettent de structurer aisément un document et ainsi de réaliser les mises en page voulues.

Un bloc (*qui pourrait être un simple Paragraphe *) est caractérisé par trois zones essentielles; une zone qui concerne l'ajustement du contenu *(qui entoure le contenu)*, appelée le remplissage *(padding)*, une bordure *(border)* et une marge *(margin)*, chacun de ces *éléments* possédant ses propres propriétés. Il est bon de retenir que la largeur et/ou épaisseur de chaque bordure ou marge peut être définie de façon individuelle ou de façon globale en CSS, en effet chaque côté de ces *éléments* possède un mot-clé approprié :

> *top* pour le haut, *bottom* pour le bas, *right* pour le côté droit et *left* pour le côté gauche



De cette façon, pour obtenir un bloc avec une marge supérieure de 10px, une marge de droite de 5px, une marge en dessous de 15px et une marge à gauche de 3px, nous aurions comme règles :

```
margin-top: 10px;
margin-right: 5px;
margin-bottom: 15px;
margin-left: 3px;
```

Cette façon d'écrire est généralement utilisée pour donner une valeur spécifique à un seul côté. Pour éviter une duplication du code, il existe une manière plus rapide d'écrire ce code :

```
margin: 10px 5px 15px 3px;
```

Les quatre valeurs sont définies en une seule et même ligne, et interprétées dans un sens bien défini. Les valeurs sont lues dans le sens horaire en partant du haut (*top*) pour finir par le côté gauche (*left*). Nous avons donc : *top* 10px, *right* 5px, *bottom* 15px, *left* 3px.

Il faut savoir qu'il vous est possible d'indiquer de 1 à 4 valeurs qui seront donc interprétées dans un sens également bien défini :

`margin: 10px 5px 15px ;`	`margin: 10px 5px ;`	`margin: 10px ;`
Le Côté qui n'est pas défini, Ici le côté gauche prendra par Défaut la valeur du côté droit. Soit : 5px	Ici, le bas(bottom) aura la même valeur que le haut donc 10px. Le côté Gauche la même valeur que le Côté droit, soit 5px.	Ici, tous les côtés auront la Même valeur. Soit : 10px.

Dans cette partie nous avons vu les marges, (*margin*) mais les mêmes règles s'appliquent pour définir une bordure (*border*) ou un espacement (*padding*).

Il vous est également possible de préciser deux autres propriétés pour ajuster un bloc, sa largeur (width) et sa hauteur (height). Gardez à l'esprit que là encore, il existe quelques différences d'interprétations en fonction du navigateur utilisé, un même bloc ne s'affichera pas forcément à l'identique dans tous les navigateurs !

6.2. Reset CSS : Késako

Comme vous n'êtes pas sans le savoir, il est assez fréquent d'avoir des différences d'affichage d'un *Navigateur* à l'autre ! En effet, espaces indésirables, marges fantaisistes, tailles de police différentes sont encore monnaie courante de nos jours, pour résumer, cela est dû aux styles par défaut de ces différents *Navigateurs*.

Vous découvrirez sur le web qu'il existe de nombreuses possibilités qui vous sont proposées pour résoudre tel ou tel problème ! Nécessaires, vraiment adaptées, conseillées ou non ? Ce n'est pas à moi d'en juger, mais comme nous venons de détailler les marges, je voulais à titre d'exemple vous citer un *sélecteur universel* que vous risquez de rencontrer sur certaines feuilles de styles :

```
* {margin: 0; padding: 0;}
```

De cette manière en fait, toutes les marges (*margin*) et espacements (*padding*) sont remis à zéro dès le début de la *feuille de style* pour tous les *Navigateurs*. Mais là, vous serez obligé de préciser des marges et padding pour tous les éléments de votre page, sous peine d'obtenir un résultat encore pire !

6.3. Le Remplissage d'un Bloc

Une autre propriété forte intéressante concerne la possibilité d'appliquer un fond à vos blocs.

En effet, il est possible d'appliquer une couleur unie en utilisant la propriété *background-color* ou plus subtilement une image en utilisant cette fois : *background-image*. Il vous sera même possible de faire se répéter cette image pour remplir un bloc par exemple.

A retenir :
Si vous avez appliqué des marges (margin) à votre bloc, ces dernières restent transparentes même si vous lui appliqué un fond. Par contre si vous appliquez une couleur à votre contenu, elle s'appliquera automatiquement à votre padding.

Pour spécifier une image d'arrière-plan à votre bloc il suffit simplement d'utiliser la valeurs d'<uri>, pointant vers votre image :

```
{ background-image: url("mon_image.gif");}
```

Il vous est également possible de faire se répéter votre image par exemple pour remplir votre bloc à partir d'une image plus petite :

```
{ background-image: url("mon_image.gif");
  background-repeat: repeat-y;}
```

Repeat : L'image se répète horizontalement et verticalement.
Repeat-x : L'image se répète horizontalement.
Repeat-y : L'image se répète verticalement.
No-repeat : L'image ne se répète pas.

Par défaut, une image se placera dans le coin supérieur gauche de son conteneur, mais il est possible d'en décider autrement !

Grâce à la propriété *background-position* vous pouvez décider de la centrer verticalement et/ou horizontalement, de la placer en bas ou en haut, ou encore de la décaler de quelques pixels…

Tout simplement avec : En Haut (*top*), à Droite (*right*), en Bas (*bottom*), à Gauche (*left*) ou au Centre (*center*) :

```
{ background-image: url("mon_image.gif");
  background-repeat: repeat-y;
  background-position: left;}
```

Ou plus proprement :

```
{ background: url("mon_image.gif")repeat-y left;}
```

L'image va donc s'afficher sur la gauche du bloc et se répéter suivant l'axe vertical (y).

Ou alors plus précisément en indiquant deux nombres comme valeur, représentant la position X (horizontale) puis Y (verticale) de l'image :

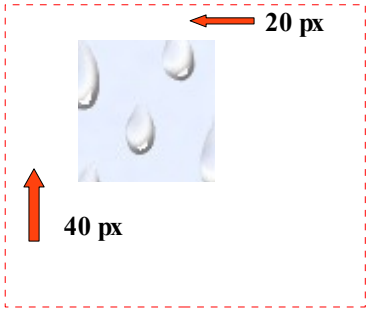
```
{ background-image: url("mon_image.gif") no-repeat;
  background-position: 40px 20px;}
```

L'image s'affichera à 40px du bord gauche et à 20px du bord supérieur.

Background repeat-x

Background repeat-y

Background 40px 20px



Vous l'aurez compris, ces possibilités de remplissage fortes intéressantes vous permettront de réaliser des mises ne page n'ayant comme limites que votre créativité !

D'ailleurs, sans entrer dans les détails, votre curiosité naturelle vous permettra de découvrir qu'il est également possible d'être encore plus précis en ce qui concerne la position exacte de vos images en utilisant par exemple les *pourcentages* à la place des *pixels* !

7. Pour Conclure

7.1. Conclusion

Vous voilà arrivé au terme de ce petit tutoriel et je l'espère aux premiers balbutiements de vos nouvelles créations !

Vous l'aurez compris, les feuilles de styles CSS associées au (X)HTML permettent de séparer totalement le design du contenu d'un site, elles offrent des possibilités immenses, un positionnement plus rigoureux des éléments, une plus grande lisibilité du (X)HTML, et facilite la portabilité du contenu . Les pages étant maintenant indépendantes du design, elles peuvent aisément s'adapter à d'autres médias, et être rendu accessibles à tous (aux handicapés notamment).

Sans avoir chercher à réinventer la poudre, les ressources à ce sujet ne manquant pas sur la toile, j'espère au moins vous avoir permit de vous y retrouver plus simplement à la lecture d'une feuille de styles et pourquoi pas, vous avoir donner envie d'aller encore plus loin concernant ce sujet.

7.2. Quelques Références

D'ailleurs, pour aller plus loin je ne saurais que trop vous conseiller de profiter des nombreuses ressources existantes sur le web.

A titre d'exemples, et sans que cette liste ne soit exhaustive je vous conseillerais:

- Plusieurs articles et tutoriels à découvrir sur le site de référence, **Alsacréations** : http://css.alsacreations.com

- Découvrir leur tutoriel pratique et complet pour la **réalisation d'un design complet en 5 étapes** : http://css.alsacreations.com/Faire-une-mise-en-page-sans-tableaux/design-css

- Un Très bon tutoriel à découvrir également, réalisé sous forme d'une série de 15 Articles : http://www.pompage.net/pompe/cssdezero-1/

- Cours et Tutoriels sur le site des développeur francophones : http://css.developpez.com/cours/

- Une expérience ultime pour ceux qui douteraient encore du potentiel énorme des CSS en matière de design, l'incontournable **CSS Zen Garden** : http://www.csszengarden.com/tr/francais/

- Le très bon site **Mammouthland** : http://css.mammouthland.net/index.php

- **Générateur de code CSS** en ligne : http://css.mammouthland.net/genecss.php

- La liste des propriétés CSS **:** http://wiki.mediabox.fr/documentation/css

- Une introduction aux différents langages servant à écrire une page Web, XHTML et bien sûr CSS : http://www.tuteurs.ens.fr/internet/web/html/

- Mais également toutes les spécifications et normes Web visibles sur le site **OpenWeb** *(une référence dans le domaine des Standards)* : http://openweb.eu.org/ressources/specifications/

Le **World Wide Web** Consortium (W3C) : http://www.w3.org/

ANNEXE

Quelques balises (x)html essentielles qui vous permettront de vous y retrouver dans une feuille de style :

Document

<html></html> : document html
<body></body> : corps du document

Titres

<h1></h1> : titre 1er niveau
<h2></h2> : titre 2ème niveau
<h3></h3> : titre 3ème niveau (... ainsi de suite jusque h6)

Éléments de texte

 : paragraphe
 : liste à puce
 : liste numérotée
: élément d'une liste
 : lien hypertexte

Tableau

<table></table>: tableau
<th></th> : légende de ligne ou de colonne
<tr></tr> : ligne du tableau
<td></td> : colonne du tableau

Positionnement

 : délimitation d'un bloc

Quelques éléments créant des éléments de rendu En-ligne :

<**span**> : élément span<a> : lien
<img> : Insertion d'une image
, : emphases
<q> : citation
<code> : élément code
**
 :** passage à la ligne
<input> : champs de saisie dans un formulaire – etc.

Quelques éléments créant des éléments de rendu Bloc :

** :** bloc container
<h1>, <h2>…<h6> : titres hiérarchisés
** :** paragraphe
<ul>, <ol>, <li>, <dl>, <dd> : éléments de liste
<blockquote> : bloc de citation
<pre> : texte pré-formaté
<form> : formulaire de saisie
<hr> : filet horizontal – etc.