Des tutoriaux CSS gratuits

Le design du site, son apparence, constituent la partie la plus visible pour donner une première impression à un visiteur lorsqu'il ouvre un site. Mettre en gros, en gras les titres, changer la couleur des liens, afficher des données dans un tableau...
Toutes ces petites bricoles destinées à donner une belle allure à vos sites mais qui vous inquiètent parfois seront abordées dans ce tutoriel.

Pourquoi utiliser CSS ?

Pour mettre des couleurs et modifier la taille d'un texte par exemple, vous mettez peut-être enHTML quelque chose du genre :

```
<font size="2" color="#003300">
```

La forme et le contenu se trouvent donc mélangés, et il vous faudra répéter ces modifications si vous voulez faire pareil ailleurs.
Pour une page, ce n'est pas difficile à réaliser. **Cependant, imaginez si on doit faire cela pour toutes les pages d'un site web**. A chaque modification d'un style, il faudra parcourir les lignes de chaque page afin de mettre à jour les définitions de tous les styles. Il deviendra alors rapidement fastidieux de répéter la même chose pour chaque page d'un site web.
Le pire, c'est que si jamais le thème du site change, **il nous faudra refaire les mêmes tâches pour chaque page**. Non seulement cela consommera énormément de temps, mais cela augmentera d'autant les possibilités d'erreurs.
S'il vous faut faire ça pour le site en entier, n'y a-t-il pas une manière de le faire sans perdre du temps, sans avoir à parcourir les pages une à une ? Les CSS vous permettent de vous épargner cette galère, puisque c'est leur rôle.
Malins comme on est, on sait qu'il y a une certaine uniformité pour toutes les pages d'un site web. Une possibilité envisageable serait donc de grouper ces propriétés quelque part et de les appeler en cas de besoin. Comme ça, on peut très facilement faire des modifications sans perte de temps. D'où la notion de CSS ou Cascading Style Sheet traduite par feuille de style en cascade.

Cours pour apprendre le CSS

Avant d'apprendre le CSS, il est fortement conseillé d'avoir une notion de base en HTML/XHTML. Pour cela, vous pouvez consulter les **tutoriaux XHTML** sur ce même site. En ce qui concerne les outils, vous pouvez utiliser les mêmes outils que pour vos documents XHTML.
Vous aurez donc besoin au minimum d'un éditeur de texte comme le bloc-note par exemple. Il y a aussi d'autres logiciels plus spécialisés tel que Notepad ++ ou Dreamweaver qui vous permettront de créer vos styles plus facilement. Le choix vous appartient librement.
A part ça, il vous faut un peu de volonté et que vous vous donniez le temps d'apprendre. Dans ce tutoriel, nous allons aborder plusieurs notions en CSS. **Dans un premier temps, nous allons surtout parler des choses basiques telles que la syntaxe de base et la notion de classe**. Ensuite nous allons voir les différentes mises en forme possibles du formatage d'un texte jusqu'à la manipulation des positionnements. Tout ceci, sans oublier d'appliquer le style en mettant en évidence la relation étroite entre XHTML et CSS.

Inclure des styles CSS dans vos pages XHTML

Il existe trois façons de définir une feuille de style. Nous allons voir dans ce tutoriel quelles sont les possibilités pour utiliser CSS dans un document XHTML.

Définition CSS spécifique en ligne

On peut définir individuellement un (ou plusieurs) style(s) pour un mot ou paragraphe de page par exemple. Cela consiste donc à intégrer les styles dans le corps du document XHTML.
Code CSS :

```
<p style="font-size: 15pt ;" >
```

L'utilisation de cette méthode revient au même que celle de faire les fonts et présente les mêmes inconvénients. **Elle ne semble intéressante que lorsque le style que vous aller définir est vraiment spécifique à cet endroit**. Autrement dit, au cas où vous pensiez réutiliser le même style à un autre endroit, il vaut mieux utiliser les méthodes suivantes.

Méthode CSS interne

Cette deuxième méthode consiste à mettre le style dans l'entête de la page. La syntaxe est incorporée à l'entête du fichier XHTML entre les balises <style type="text/css"> .... </style> placées entre les tags <head> et </head>.
Le style ainsi défini peut être appliqué à tous les endroits de la page, sans avoir à redéfinir les propriétés. **Néanmoins, cela nécessite d'inclure le code du style à l'entête de chaque fichier, et de le répéter pour tous les fichiers**. Donc ce n'est pas encore optimal.

Méthode CSS externe

En linking, vous utilisez un fichier séparé, indépendant du code XHTML, et qui porte l'extension .css. (ou n'importe quelle autre extension, celle-ci présentant l'avantage de savoir qu'il s'agit d'un fichier de feuilles de style.).
Ce fichier contient tous les styles CSS utilisés dans toutes les pages. Vous placez une simple ligne (toujours la même) qui servira à appeler le fichier .css entre les tags <HEAD> et </HEAD>, et votre fichier est opérationnel.
Cette méthode présente divers avantages par rapport au deux précédentes :

- Les définitions de style ne sont faites qu'une seule fois, à un seul endroit
- Cela vous réduit considérablement la taille de votre page XHTML
- Les navigateurs peuvent mettre en cache le contenu de la feuille de styles qui s'appliquera sur toutes les pages du site. Une fois cette mise en cache effectuée, le navigateur n'aura plus qu'à charger le contenu de la page. D'où une forte réduction du temps de transfert.
- Cela facilite la maintenance. Vous n'avez qu'à manipuler un seul fichier lorsque vous voulez modifier un tel ou tel aspect de votre site.
- Cela donne une meilleure structuration de votre page puisque que le contenu lui-même est séparé de la forme.
- Il est possible de créer des règles d'appel à ce fichier : Si c'est un "ordinateur" (ou plus exactement un "écran") qui appelle la page alors on lui envoie le fichier CSS destiné à l'écran. Si c'est l'imprimante, alors on lui envoie le fichier CSS pour l'imprimante, etc..

Et il est donc possible, comme son nom l'indique (Cascade Stylesheet), de définir des styles "en cascade", en fonction des pages, et de l'environnement (écran, imprimante, flux RSS, etc...).

Comment attacher une feuille de style à un document XHTML :

Dans un fichier style.css, vous mettez par exemple le contenu de votre style :
Code CSS :

```
menu{
font-size :12px ;
font-weight :900 ;
}
```

Dans un premier temps il est important que vous compreniez le principe et l'organisation des fichiers. **Pour l'instant ne vous inquiétez pas si vous n'arrivez pas à traduire ce qui est écrit dans ce fichier** : Patience. Ensuite vient le moment de l'attacher au document XHTML.
Dans l'entête de chacun de vos fichiers, il y a juste à rajouter une ligne pour inclure le fichier style.css :
Code CSS :

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

Il faut donc spécifier dans href le fichier à inclure. L'attribut média permet de spécifier le type de média auquel le style sera appliqué. Le média peut aussi bien être l'ordinateur qu'un téléphone portable, une imprimante, etc...
Vous avez peut être vu quelque part une autre syntaxe comme ceci, toute aussi applicable :
Code CSS :

```
<style type="text/css">
@import url(style.css) ;
</style>
```

Il est possible, comme nous l'avons vu plus haut, d'incorporer plusieurs lignes de style, comme ceci (principe des 'cascades') :
Code CSS :

```
<head>
<link href="style-1.css" rel="stylesheet" type="text/css" media="all">
<link href="style-2.css" rel="stylesheet" type="text/css" media="print">
</head>
```

Cours pour apprendre la syntaxe CSS de base

Une feuille de style contient la définition de différentes règles CSS. **C'est en appliquant ces règles qu'on peut mettre en forme les éléments**, ou plus concrètement leur donner du style.
Une règleCSS est divisée en 2 parties :

- le sélecteur
- le bloc de déclaration

C'est le sélecteur qui détermine ou identifie les parties du document auxquelles sera appliqué le style. Il sert à sélectionner l'élément que l'on a envie de mettre en forme. Le bloc de déclaration, quant à lui, contient les déclarations des styles à appliquer. Il est toujours mis entre 2 accolades :
Code CSS :

```
body
{
color:red
}
```

Ici, body est le sélecteur alors que le bloc de déclaration est la zone délimitée par les accolades. Autrement dit, les styles définis entre accolades seront appliqués à la balise body.

Le bloc de déclaration CSS

Dans un bloc de déclaration, on liste tous les styles que l'on veut utiliser pour l'élément sélectionné. Ce bloc est composé de déclarations qui elles-mêmes contiennent une propriété et une valeur. **La propriété correspond à l'attribut qu'on souhaite changer**. A chaque propriété peut être affectée une valeur. Une propriété et une valeur sont toujours séparées par le symbole "deux points" (:).
Reprenons l'exemple de tout à l'heure, tout en regardant bien le bloc :
Code CSS :

```
body
{
color:red
}
```

color est une propriété qui spécifie la couleur, red est une valeur de la propriété qui signifie rouge. Le style défini sous entend qu'il faut donc mettre la couleur des textes de la balise body en rouge.
Si la valeur est composée de plusieurs mots, il serait mieux de le mettre entre guillemets comme dans cet exemple :
Code CSS :

```
body
{
font-family: "Times new roman"
}
```

Un bloc peut contenir plusieurs déclarations. Dans ce cas, les déclarations sont obligatoirement séparées par des points virgules (;).
Code CSS :

```
body
color:red ;
font-family: "Times new roman";
}
```

Il est d'usage d'écrire les déclarations ligne par ligne. **Sachant que c'est facultatif, il est tout à fait possible de les écrire sur la même ligne**. C'est juste une question de lisibilité. On peut omettre le point virgule sur la dernière déclaration. Il vaut mieux quand même le mettre partout afin d'éviter les oublis.
Le CSS n'est pas sensible à la casse. C'est-à-dire que cela ne change rien si vous écrivez 'MENU', ou 'MeNu' au lieu de 'menu'. Cependant, il est plus convenable de tout écrire en minuscules.

Les commentaires en CSS

Il est très conseillé de mettre des commentaires dans votre code CSS. L'avantage de les mettre est qu'ils permettent à d'autres utilisateurs de comprendre votre code facilement en cas de travail en groupe par exemple. Sinon, c'est aussi dans votre intérêt personnel. **Si vous décidez de reprendre le code après un long moment sans l'ouvrir, les commentaires vous aideront beaucoup à vous rafraîchir la mémoire**.
De plus, les commentaires permettent de structurer le code et le rend plus lisible. **Délimités par slash étoile et étoile slash, les commentaires ne sont pas interprétés par les navigateurs**. Vous pouvez mettre tout ce que vous voulez dedans, ils n'apparaîtront pas sur l'écran. N'ayez donc aucune crainte, ils n'ont aucune influence sur l'affichage des contenus.
Code CSS :

```
/*commentaire */
```

Par contre, il ne faut jamais les mettre en imbriqué comme suit :
Code CSS :

```
/* commentaire
/* encore un commentaire */
```

A retenir dans ce tutoriel

Afin que nous puissions partir du bon pied, voici les essentiels à retenir :

- Il faut toujours bien fermer { .... } les "accolades" quand on définit des blocs
- N'hésitez pas à mettre une déclaration par ligne, pour plus de clarté
- Ne jamais oublier d'utiliser les " : " deux-points pour séparer la propriété de sa valeur
- Mettre un " ; " point-virgule en fin des blocs en cas de plusieurs déclarations
- Placer les commentaires entre /*et */ , ne jamais les emboîter

Les sélecteurs de type en CSS

Un sélecteur de type est un des moyens les plus simples pour appliquer un style. Il nous permet d'affecter un style à un élément XHTML. En spécifiant son nom, vous pouvez appliquer directement le style à une balise. Pour appliquer un style à la balise p par exemple on peut faire comme ceci :

```
p{
color : green
}
```

Les classes CSS

Dans le sélecteur de type, on n'a utilisé qu'un seul bloc de déclaration pour les paragraphes. Mais supposons que vous voulez définir 2 types de paragraphe : des paragraphes en vert et des paragraphes en rouge. **Vous ne pouvez pas définir ces deux propriétés dans un même style**. Cela est réalisable avec les classes. La première étape sera de définir 2 classes. Les noms des classes sont donc : .vert et .rouge et sont toujours suivis d'un point.
Code CSS :

```
p.vert{
color:green
}

P.rouge {
color : red
}
```

Pour appliquer ces styles aux documents XHTML, il suffit de mettre le nom de la classe comme valeur de l'attribut class.
Code XHTML :

```
<p class="vert">
Ce paragraphe est de couleur verte.
</p>
<p class="rouge">
Ce paragraphe est de couleur rouge.
</p>
```

Nous pouvons aussi omettre le nom de la balise ('p') dans le sélecteur pour définir par exemple un style qui sera utilisé par d'autres balises XHTML. Dans l'exemple suivant, tout élément ayant comme classe .vert sera coloré en vert.
Code CSS :

```
.vert{
color:green
}
```

Code XHTML :

```
<h1 class="vert">Ce titre est coloré en vert</h1>
<p class="vert">
Tout comme le titre ci-dessus, ce paragraphe est aussi de couleur verte.
</p>
```

L'intérêt d'utiliser les classes c'est de pouvoir regrouper des propriétés de style dans une classe et d'appliquer ces styles à différents éléments, et en différents endroits. Définissons par exemple trois groupes de propriété :

- Premier groupe de propriété ou style : centré en bleu, avec des textes plus petits
- Deuxième groupe de propriété ou style : aligné à droite en rouge, avec des textes plus grands
- Troisième groupe de propriété ou style : normal en noir, avec des textes moyens.

Et mieux encore nous avons la possibilité de donner n'importe quel nom à notre classe à condition de bien respecter les règles de syntaxe (il ne doit pas contenir d'espace). Profitons alors de cette possibilité et donnons à nos classes des noms significatifs que l'on pourra retenir facilement.
Nous voulons par exemple assigner : aux remarques, le premier groupe de style, aux notes, le deuxième groupe de style et aux paragraphes normaux, le troisième groupe de style. On va créer trois classes :

- remarque pour dire qu'il s'agit d'une remarque
- note pour dire que c'est 'à noter'
- normal pour dire qu'il s'agit d'un paragraphe normal

Code CSS :

```
.remarque {
font-size:small;
text-align:center;
color:blue;
}
.note {
font-size: large;
text-align:right;
color:red;
}

.normal {
font-size: medium;
}
```

Code XHTML :

```
<p class="remarque">
Ce paragraphe est une remarque.
</p>
<p class="note">
Ce passage est à noter. C'est quelque chose de très important donc à retenir.
</p>
<p class="normal">
C'est un paragraphe normal.
</p>
```

Les 3 éléments ont bien la même balise HTML ('p'), mais des styles différents, qui les présentent de manière différente.

Les id en CSS

Contrairement aux classes, un id s'applique à un objet unique. **Il n'est pas possible d'avoir deux éléments de même id dans une même page**. Autrement dit, si un élément XHTML possède déjà un id, cet id ne peut plus être utilisé pour un autre élément XHTML. Alors qu'il est tout à fait possible que deux éléments utilisent la même classe, bien au contraire. La définition d'un ID en CSS est identique à celle d'une classe, excepté le dièse qui précède le sélecteur :

```
test-id {
color :green ;
}
```

Comme pour la classe, pour appliquer le style à une balise div, on spécifie le nom de l'id comme valeur de l'attribut id :

```
<div id="test-id" >un exemple d'id</div>
```

Rappel : Il ne peut y avoir qu'un seul élément dont l'id est test-id dans la même page. **On utilise généralement l'id pour placer des blocs dans une page** (menu droit, menu gauche, contenu principal, etc...) . L'id peut aussi servir d'ancre nommée. Cela veut dire que si on est en bas de

page, et qu'on veut y retourner rapidement en haut, il suffit de faire un lien sur la même page. L'id spécifie l'endroit où l'on doit se rendre.

Rappel pour renvoi vers une ancre :

```
<a href="#test-id">retour à test-id</a>
```

Pour plus d'explications, voir le **chapitre sur les liens** dans le tuto XHTML.

Comment réaliser un groupement de sélecteurs ?

On peut grouper les sélecteurs. Pour cela, il faut juste les lister et séparer les éléments de la liste par une virgule. Dans l'exemple ci-dessous on a groupé tous les titres. Cela permet d'avoir des titres qui ont tous la couleur verte

```
h1,h2,h3,h4,h5,h6{
color: green
}
<h1>Un titre h1 en vert</h1>
<h2>Un titre h2 en vert aussi</h2>
...
<h6>Un titre h6 en vert aussi</h6>
```

On peut même grouper des sélecteurs de types différents. C'est-à-dire qu'il est possible de définir un style à plusieurs éléments, classes, identifiants. Comme tout à l'heure, il suffit de lister les éléments en les séparant par une virgule.

Code CSS :

```
h1, .uneclasse, #idspecial {
color : red;
font-weight : bold;
}
```

Ainsi, Ies éléments h1, les classes uneclasse et l'identifiant idspecial auront les mêmes styles : colorés en rouge et mis en gras.

Code XHTML :

```
<div id="idspecial">Un id avec le même style</div>
<h1>Un titre avec le même style</h1>
<p class= "uneclasse">Un paragraphe avec le même style</p>
```

Et l'inverse alors ?

Un élément XHTML peut avoir plusieurs classes. On peut ainsi combiner les styles comme on veut.**Pour appliquer plusieurs classes à un élément, la syntaxe est presque la même que celle définie ci-dessus**. Il suffit de mettre comme valeur de l'attribut class la liste des noms des classes séparés par un espace.

Code CSS :

```
.vert {
color:green;
}

.centre{
text-align : center;
}
.gras {
font-weight:bold;
}

.fondjaune {
background-color:yellow;
}
```

Code XHTML :

```
<h1 class="vert centre fondjaune">Application de 3 classes pour le titre </h1>
<p class="vert gras">On n'applique que deux styles au paragraphe : .vert et .gras.</p>
<p class="vert">Et là, c'est un paragraphe avec un seul style. Donc une seule classe : .vert.</p>
```

Comment fonctionne l'héritage en CSS ?

On a pu voir que chaque élément XHTML peut avoir chacun soit sa propre classe, soit son propre style. Mais les balises XHTML peuvent aussi s'emboîter les unes dans les autres.

Dans ce cas, comment le navigateur effectue t-il la mise en forme ? En d'autres termes, dans le cas d'emboîtement des balises comme celui-ci, comment cela se passe t-il :

Code XHTML :

```
<body class="stylebody">
<p class= "stylep"> </p>
< /body>
```

On veut savoir quel style sera appliqué au contenu de la balise <p> qui fait aussi partie du contenu de la balise body. C'est là qu'intervient la notion d'héritage. **On appellera donc "parent", le style du niveau supérieur (le stylebody dans notre cas) et enfant les styles de la balise en dessous (stylep dans notre cas)**. La balise "body", avec ses différents styles, se voit hériter des styles de la classe transmise.
En l'occurrence "stylebody". Les éléments de "body", comme le paragraphe "p" par exemple, seront donc conformes au style de "body", auquel on aura appliqué le style de la classe "stylebody". En tant qu'enfant de body, de stylebody, et de p, , stylep hérite des propriétés de ses parents. **S'il y a deux valeurs différentes pour la même propriété, l'enfant garde la sienne**, sinon on remonte l'arborescence pour connaître le style associé. Voyons ça plus explicitement à l'aide d'un exemple.
Code CSS :

```
.stylebody {
font-size: 10pt;
font-family: arial, verdana, sans-serif;
color: green;
text-align: left;
background-color: #FFFFFF ;
}

.stylep {
background-color:yellow;
font-style:italic;
text-align:center;
}
```

Code XHTML :

```
<body class="stylebody">

<p>Voici un paragraphe sans style, il hérite donc entièrement de body, puis de stylebody .</p>

<p class= "stylep">Un paragraphe avec style, qui hérite encore de body et de stylebody, mais qui garde ses propres styles, définis par la classe 'stylep'..</p>
```

Explication de l'exemple

Dans le paragraphe avec style les 3 propriétés de la balise body sont gardées:

- font-size: 10pt ;
- font-family: arial, verdana, sans-serif
- color: green ;

Les 2 autres propriétés : **text-align: left** et **background-color: #FFFFFF** ont été remplacées respectivement par **text-align:center** et **background-color:yellow** qui sont des propriétés propres à lui-même. Et il a une autre propriété qu'il n'a pas hérité mais qui lui est aussi propre font-style:italic;
Le tableau suivant nous résume la fusion des styles par héritage dans cet exemple.

Fusion des styles par héritage

Classes	Styles
Styles dans .stylebody	font-size: 10pt ; font-family: arial, verdana, sans-serif ; color: green ; text-align: left ; background-color: #FFFFFF ;
Styles dans le .stylep	font-style:italic; text-align:center; background-color:yellow;

Styles appliqués au paragraphe de classe stylep par héritagefont-style:italic;	text-align:center; background-color:yellow; font-size: 10pt ; font-family: arial, verdana, sans-serif ; color: green ;

Attention, le principe de l'héritage n'est pas toujours valable. C'est applicable pour certains styles comme la couleur par exemple. **Cependant, il y a des exceptions pour certains propriétés des éléments de type bloc tels que padding, marging...**
Ces propriétés ne se transmettent pas aux balises emboîtées. Autrement dit, un élément de type en-ligne emboités dans un type bloc et n'ayant pas de marge n'héritera jamais des marges de son conteneur. Néanmoins, la notion d'héritage est très pratique aux niveaux des CSS donc ça valait la peine de l'avoir étudié.

Comment est géré l'ordre de priorité des styles ?

Priorité entre les classes et les sélecteurs de type
Dans l'exemple suivant on applique la couleur verte et la mise en gras à toutes les balises p.
Code CSS> :

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

.pspecial {
color:green;
}
```

Au niveau du code XHTML, voyons ce qui se passe si on utilise la classe .pspecial dans une balise p particulier.
Code XHTML> :

```
<p> Ce paragraphe s'affiche en rouge, et en gras.</p> </p> <p class=".pspecial">test<p> Ce paragraphe s'affiche en vert parce que le style de classe est prioritaire par rapport au style standard de la balise p. <p> Ce paragraphe s'affiche aussi en gras rouge. </p>
```

Remarquons que tous les paragraphes sont colorés en rouge sauf un. Il s'agit du paragraphe où l'on a appliqué le style .pspecial. Par contre, le paragraphe en question a gardé la mise en gras. Les propriétés des deux sélecteurs (p et .pspecial) se combinent donc entre elles.
S'il y a des propriétés communes (color par exemple), c'est la valeur définie dans la classe (color:green) qui sera considérée. En d'autres termes, les classes (.pspecial) sont prioritaires par rapport aux styles standards définis à l'aide des sélecteurs de type (p)

Priorité entre deux classes de même nom

Si deux classes de même nom sont définies avec des propriétés différentes, c'est celle de la dernière classe définie qui sera prise en compte.
Code CSS :

```
.pspecial {
color:red;
}

.pspecial {
color:yellow;
}
```

Lorsqu'on utilise la classe .pspecial dans une balise p par exemple, le paragraphe sera coloré en jaune mais pas en rouge puisque c'est la dernière définition qui est prioritaire.
Code XHTML :

```
<p class="pspecial"> C'est la dernière définition qui sera prise en compte. </p>
```

Priorité entre les classes et les id

Définissons maintenant deux types de styles, un à l'aide d'un id et un autre en utilisant une classe.
Code CSS :

```
#unid {
color:blue; }

.uneclasse
{
color:red;
font-weight:bold;
}
```

Code XHTML :

```
<p id="unid" class="une-classe">
Eh voilà, c'est l'id qui emporte.
</p>
```

Le paragraphe s'affiche en bleu puisque c'est le style de l'id qui est prioritaire sur la classe. Priorité entre les styles internes et la feuille externe css
Par exemple, vous réunissez tout votre style dans une feuille de style CSS. Dans cette feuille de style, vous spécifiez que tous les paragraphes des pages XHTML auxquelles sera attachée cette feuille de style auront une taille de police de 20px.
Pour une raison ou pour une autre, vous auriez besoin de définir un autre style qui est local à votre page tout en rattachant la feuille de style externe. Il se peut que dans cette page, il y ait un paragraphe spécial qui doit être spécifié en taille plus grande. Dans ce cas c'est le style défini en interne qui est prioritaire par rapport au style externe.
Voici par exemple le code dans la feuille de style externe qu'on va nommer feuille.css :
Code CSS :

```
p {
font-size: 20px;
}
```

Voici la page XHTML à laquelle on rattache la feuille de style feuille.css :
Code XHTML :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Priorité des balises</title>

<style type="text/css">
p.italic {
font-style:italic;
font-size: 40px;
}

</style>
</head>

<body>
<p>
Ce paragraphe suit le style externe.
</p>
<p class="italic">
Ce paragraphe suit le style interne.
```

```
</p>
</body>
</html>
```

Le style en interne p.italic est appliqué parce qu'il est prioritaire par rapport au style externe.

Les unités de mesure

La spécification de certaines valeurs en CSS nécessite l'utilisation d'un certain nombre d'unités de mesure. Cette notion est importante pour que l'on ne s'y perde pas nous-mêmes dans nos spécifications.
Il y a plusieurs façons d'indiquer des valeurs en CSS et en fonction de celles-ci, on utilise différentes unités de mesure. Pour spécifier une valeur en CSS on l'exprime parfois juste avec des termes en anglais.
Parfois il est jugé utile d'employer des valeurs numériques pour être plus précis. En général, les propriétés en CSS prennent le plus souvent un des cinq types de valeurs suivantes :

- **Mots-clés** : Par exemple : bold
- **Valeurs en pourcentage** : % (par exemple 50%)
- **Couleurs** : #RRVVBB, #RVB, rgb, mot clé (par exemple #D9BB7A, #C9B, red)
- **URL** : Par exemple : http://www.google.com
- **Valeurs** de longueur : Par exemple : 1cm

Dans ce chapitre, nous allons nous intéresser surtout aux valeurs de longueur. Quant aux autres valeurs, on va en parler très prochainement dans les tutoriels suivants. Il faut tout simplement les noter quelque part pour l'instant.

Les valeurs de longueur

Les valeurs de longueur sont données en valeur numérique. Pour cela, il y a deux sortes d'unités :

- unités absolues : in, mm, cm, pt, pc
- unités relatives : em, ex, px

Les unités de longueur absolues

Les unités absolues ont un certain rapport entre elles comme l'indiquent les valeurs suivantes :

- cm (centimeter = 1cm = 10 mm)
- in (inche = 1in = 2.54 cm)
- mm (millimeter)
- pc (pica = 1pc = 12pt)
- pt (point = 1pt = 1/72 in)

Par contre, elles n'ont pas de rapport avec le contexte, avec le reste du document. Nous allons voir maintenant les unités de longueur relatives

L'unité em

L'unité em se rapporte à la taille de la police. Avec elle on peut affecter une mesure relative à la taille de police de l'élément parent. **Elle permet d'avoir des feuilles de style plus facilement adaptables d'un média à un autre**. Les nombres décimaux sont autorisés, mais il faut tout simplement remplacer la virgule par un point. Cette valeur em est utilisable pour d'autres propriétés acceptant la mention de longueur.

L'unité ex

L'unité ex est utilisée souvent pour exprimer la hauteur des caractères. Le point de référence est la hauteur des minuscules souvent appelée hauteur "x" lowercase. En d'autres termes, dans le cas où la taille est donnée avec une unité "ex", celle-ci est alors mesurée par rapport à la hauteur de la minuscule x de l'élément parent.
Tout comme l'unité em, cette unité est utilisée pour toutes les propriétés acceptant la mention de longueur. Toutefois, la plupart des navigateurs ont encore une certaine hésitation à traduire correctement cette valeur.

L'unité px

L'écran d'un ordinateur ou moniteur est formé par plusieurs petits "carrés". Ces carrés définissent la résolution ou densité de l'écran en pixels d'affichage selon l'unité de sortie, c'est-à-dire l'écran de l'ordinateur. L'unité px qui veut dire pixel correspond à un de ces petits carrés.
C'est en fait le plus petit élément de la résolution d'écran. Cette unité peut être utilisée pour toutes les propriétés acceptant les mentions de longueur.

Comment mettre en forme les polices de caractères ?

En parcourant les exemples des tutoriels précédents, vous avez dû vous apercevoir que l'on a souvent utilisé des propriétés telles que font-size, font-family, color etc. **Pour autant, on n'a pas encore expliqué ce que cela signifie exactement**. Dans les chapitres qui suivent, on essaiera de voir tout cela plus en détail.
Commençons d'abord par tout ce qui concerne la typographie : les fonts, ou polices de caractères. La mise en forme nous permet d'avoir des polices de caractère personnalisées à notre goût. Choix de la police
Le pouvoir de l'informatique nous permet d'avoir une multitude de polices de caractères qui ont chacune leur forme. Parmi les standards on distingue par exemple les times new roman, arial, etc Dans ce cas, il ne vous reste plus qu'à choisir les polices selon votre préférence. Ensuite, vous spécifierez votre choix dans la feuille de style en utilisant la propriété font-familly comme suit :
Code CSS :

```
p {
font-family: "Agency FB"; /* font fantaisiste ;) */
}
```

Cependant, il ne faut surtout pas oublier que vos pages web seront visitées. La police ne s'affichera pas forcément telle que vous souhaiteriez la voir apparaître sur l'écran de l'internaute. En effet, Il se peut que la police que vous avez spécifiée n'existe pas sur le système du visiteur. Vous avez donc la possibilité de lui proposer d'autres choix alternatifs.
Code CSS :

```
p {
font-family: "Agency FB", Calibri, "Californian FB", serif
}
```

Le navigateur essaiera d'abord de mettre la police en Agency FP. Si ne le visiteur ne l'a pas, il essaiera de mettre la Calibri et ainsi de suite. Il faut séparer par une virgule la liste des polices. Si le nom de la police contient un espace, il faut le mettre entre guillemets comme "Californian FB". En principe, on a intérêt à mettre en tout dernier une police standard. C'est-à-dire une police dont on est certain qu'elle existe partout. Comme ça on est sûr que l'affichage sera toujours correct, au moins dans cette police. Il ne faut néanmoins pas oublier de tester l'affichage dans cette police également. Il y a deux familles distinctes

- famille classique : arial, times new roman, etc.
- famille générique : serif, sans-serif,etc.

La plupart du temps, on met les polices de la famille générique au moins en dernier, particulièrement les serifs. Style de la police
Comme son nom l'indique, la propriété font-style permet de définir le style de la police. Grâce à cette propriété, on peut avoir des polices en italique ou normale.
Code CSS :

```
.normale {
font-style:normal
}

.italique {
font-style:italic
}
```

Code XHTML :

```
<p class="normale">Police normale</p>
<p class="italique">Police italique</p>
```

Mettre en gras en CSS

La mise en gras en CSS permet de mettre du poids sur les polices. **D'où le nom de la propriétéCSS font-weight** .Cette propriété définit l'épaisseur de la police. On l'utilise souvent

dans les titres, dans les paragraphes, etc Elle peut prendre la valeur bold pour gras ou normal pour le normal (pas gras)

Code CSS :

```
.normale {
font-weight:normal
}

.bold {
font-weight:bold
}
```

Code XHTML :

```
<p class="normale">pas gras </p>
<p class="bold">Police gras </p>
```

Il est possible également de définir un poids avec un attribut numérique : 100, 200... 900. 400 correspond alors à 'normal', alors que '700' correspond au 'gras' traditionnel. L'échelle va de 100 en 100.

Taille de la police avec différentes valeurs en CSS

En CSS, on a différentes façons de définir la valeur de la taille d'une propriété. Pour définir la taille de la police on utilise la propriété font-size.
Il y a deux façons de spécifier les valeurs de la taille de la police. On peut donner une valeur soit en la spécifiant par des mots clés, soit en indiquant sa valeur numérique à l'aide des unités de mesure que l'on a vues tout à l'heure.

Spécifier la taille par des mots clé

On peut donner une valeur relative de la taille par rapport à celle de l'élément parent. Les mots-clés utilisés sont :

- smaller
- medium
- larger

Sachant que sur Internet Explorer, la police par défaut est medium. Lorsque l'on spécifie ainsi la taille, la valeur de base est alors le navigateur de l'internaute.

Code XHTML :

```
<div>
<span style="font-size:smaller ;">Police small </span><br>
<span style="font-size: medium ;">Police medium</span><br>
<span style="font-size: larger ;"> Police large</span><br>
</div>
```

On peut spécifier de façon absolue la taille de la police tout en utilisant toujours des mots clés comme précédemment. Sauf que donner la valeur absolue consiste à juste dire si c'est petit, moyen ou grand sans spécifier la taille exacte et sans donner une référence. Les mots clés possibles par ordre croissant de la valeur sont :

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

Code CSS :

```
<div>
<span style="font-size: xx-small;">Police xx-small </span><br>
<span style="font-size: x-small;">Police x-small</span><br>
<span style="font-size: small;"> Police small</span><br>
<span style="font-size: medium;"> Police medium</span><br>
<span style="font-size: large;"> Police large</span><br>
<span style="font-size: x-large;"> Police x-large</span><br>
```

```
<span style="font-size: xx-large;"> Police xx-large</span><br>
</div>
```

En termes de taille, les polices fonctionnent comme les tee-shirts. Il y en a pour toutes les tailles, il y a juste à employer le mot clé convenable et vous trouvez facilement ce dont vous avez besoin.
Spécifier la taille de la police grâce aux valeurs numériques
La taille de la police peut aussi être indiquée en pourcentages par rapport à celle de l'élément parent. La taille de l'élément parent sera servira donc de référence pour le calcul de la valeur de la taille de l'élément. Pour cela, on indique la valeur numérique et après on met le signe % tout simplement.
Code CSS:

```
p.taille150{font-size: 150%}
p.taille130 {font-size: 130%}
p.taille100{font-size: 100%}
```

Code XHTML :

```
<p class="taille150">taille 15O pourcent</p>
<p class="taille130">taille 130 pourcent</p>
<p class="taille100">taille 100 pourcent</p>
<p>Taille normal</p>
```

On peut aussi spécifier la taille de police de façon absolue mais en utilisant des valeurs numériques exactes. Autrement dit, la taille spécifiée est indépendante de la table des tailles de police du visiteur.
Code CSS :

```
p.taille12pt{font-size: 12pt;}
p.taille50px {font-size: 50px;}
p.taille2mm{font-size: 2mm;}
```

Code XHTML :

```
<p class="taille12pt">taille 12 point</p>
<p class="taille50px">taille 50 pixel</p>
<p class="taille2mm">taille 4mm </p>
```

La syntaxe de définition est la même que celle la précédente sauf qu'à la place du signe % on met l'unité de mesure. Il suffit donc de donner la valeur numérique, suivie de l'unité de mesure em, ex, px, in, cm, mm, pc, pt (voir le cours sur l'unité de mesure).
En revanche, il ne faut pas mettre de valeurs négatives. Ne pas mettre non plus d'espace entre la valeur et l'unité (16 px ne marche pas toujours)

Mettre des textes en majuscules et minuscules

Les propriétés CSS permettent de mettre automatiquement des textes en majuscules dans un texte. Pour cela, il y a 2 sortes de propriétés.
Premièrement, voyons la propriété font-variant. Elle peut prendre 2 valeurs différentes :

- small-caps qui donnera un texte en petites majuscules
- normal qui est la valeur par défaut et donnera un texte normal

Code CSS :

```
p.normale {
font-variant: normal;
}

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

Code XHTML :

```
<p class="normale">texte normale<p>
<p class="petite">texte en capitale </p>
```

Il y a aussi une autre propriété CSS qui permet de forcer les majuscules: text-transform d'un texte. Elle peut prendre ces valeurs :

- none : ne met aucun effet sur le texte
- lowercase : met toutes les lettres de chaque mot en minuscules

- uppercase : met toutes les lettres de chaque mot en majuscules
- capitalize : met uniquement la première lettre de chaque mot en majuscule

Code CSS :

```
.aucun {
text-transform:none;
}

.minuscule {
text-transform: lowercase;
}

.majuscule {
text-transform: uppercase;
}

.capitale {
text-transform: capitalize;
}
```

Code XHTML :

```
<p class="aucun">None : ne met aucun effet sur le texte</p>
<p class="minuscule">Lowercase : met toutes les lettres de chaque mot en minuscules</p>
<p class="majuscule">Uppercase : met toutes les lettres de chaque mot en majuscules</p>
<p class="capitale">Capitalize : met uniquement la première lettre de chaque mot en majuscule</p>
```

Espacement entre les caractères : letter-spacing

La propriété letter-spacing permet de spécifier l'espacement entre les caractères du texte. Les règles d'héritage sont applicables pour cette propriété. Elle peut prendre 2 valeurs :
Normal :
La valeur "normale" est la valeur par défaut de la propriété letter-spacing. Elle correspond à l'espacement normal de la police en question.
Valeur de longueur :
C'est une valeur numérique qui détermine la quantité d'espacement entre les lettres. Elle s'ajoute à la valeur normal. Elle peut être exprimée avec des unités de mesure ou en pourcentage. Celle-ci peut prendre une valeur négative. Voici un exemple d'utilisation de la propriété letter-spacing.
Code CSS :

```
.espace1 {
letter-spacing: -2px;
}

.espace2 {
letter-spacing: 1cm;
}
```

Code XHTML :

```
<p class="espace1">ESPACE</p>
<p class="espace2">ESPACE</p>
<p>ESPACE</p>
```

Le mot ESPACE est écrit de façons différentes. Les caractères de la classe espace 1 sont beaucoup plus serrés entre eux que ceux de la normale.

Espacement entre les mots : word spacing

Si la propriété letter-spacing définit l'espace inter-lettrage, la propriété word-spacing quant à elle définit le comportement de l'espace entre les mots. Les valeurs applicables sont pareilles que celles de letter-spacing:

- **Normal** : La valeur normale correspond à l'espacement normal entre les mots pour une police bien déterminée.
- **Valeur de Longueur** : Cette valeur indique la quantité d'espacement qui s'ajoute à l'espace mot par défaut c'est-à-dire normal. Une valeur négative est acceptée.

Exemple d'espace entre les mots :
Code CSS :

```
.espacemot1 {
word-spacing:20px; /*espace de 20 px entre les mots*/
}

.espacemot2{
word-spacing:1cm;/*espace de 1 cm entre les mots*/
}
```

Code XHTML :

```
<p class="espacemot1">Je vais chez ce cher Serge si sage et si chaste.</p>
<p class="espacemot2">Je vais chez ce cher Serge si sage et si chaste.</p>
<p>Je vais chez ce cher Serge si sage et si chaste.</p>
```

Centrer, aligner et justifier des textes en CSS

L'alignement des textes en CSS est défini par la propriété 'text-align'.Cette propriété peut prendre différentes valeurs selon l'alignement souhaité : La valeur left correspond à l'alignement à gauche et c'est la valeur par défaut.
L'autre valeur right correspond à l'alignement à droite. Avec la valeur center, le texte sera centré. La valeur justify permet de justifier le texte. La justification consiste à faire en sorte que le texte occupe toute la largeur possible, et qu'il soit aligné aussi bien à droite qu'à gauche. Pour retenir facilement, voici en résumé les différentes valeurs et leur signification:

- left : aligner à gauche
- right : aligner à droite
- center : centrer
- justify : justifier

Voyons ces différents alignements dans cet exemple.
CSS :

```
h1 {
text-align: center; /* centré */
}

.justifie {
text-align: justify; /* justifiée */
}

.gauche {
text-align: right; /*aligné à gauche*/
}
```

Code XHTML :

```
<h1>Paragraphe sans justification</h1>
<p>La valeur left correspond à l'alignement à gauche et c'est la valeur par défaut. L'autre valeur right correspond à l'alignement à droite. Avec la valeur center, le texte sera centré.
La valeur justify permet de justifier le texte. La justification consiste à faire en sorte que le texte occupe toute la largeur possible.
</p>
<h1>Paragraphe avec justification</h1>
<p class="justifie">La valeur left correspond à l'alignement à gauche et c'est la valeur par défaut.
L'autre valeur right correspond à l'alignement à droite.
Avec la valeur center, le texte sera centré. La valeur justify permet de justifier le texte. La justification consiste à faire en sorte que le texte occupe toute la largeur possible.
</p>
<p class="gauche">
<a href="exemple.html">lire la suite...</a>
</p>
```

Il n'est pas possible de modifier l'alignement d'un texte d'une balise in line telles qu'em, strong, etc. L'alignement ne concerne que les types block. Cela n'a rien de vraiment étrange, c'est juste une question de logique. Modifier l'alignement d'un mot qui se trouve au milieu d'un paragraphe n'a aucun sens logique.

Alignement vertical

Comme son nom l'indique, la propriété vertical-align permet de définir le positionnement vertical d'un texte par rapport à un élément de type in-line. En général, elle ne bénéficie pas des règles d'héritage. On trouve l'intérêt de cette propriété particulièrement dans la manipulation des cellules d'un tableau. Cependant, la propriété pose un problème sur certains navigateurs comme IE6 par exemple. Différentes valeurs possibles sont proposées dans la liste suivante :

- **super** : place le texte en dessus de l'élément parent comme un exposant
- **top** : Le sommet du texte est aligné sur celui du plus haut de l'élément de référence
- **middle** : Le milieu du texte s'aligne au milieux de l'élément parent
- **baseline** : C'est la valeur par défaut et elle indique la position normale. Elle respecte l'alignement à gauche et place le texte au même niveau que l'élément in-line de référence
- **bottom** : La partie inférieure du texte est alignée sur celle du plus bas de l'élément de référence
- **sub** : Le texte est placé en dessous de l'élément parent comme un indice

Et il y a aussi deux autres valeurs : text-top et text-bottom. Les deux sont similaires respectivement à top et bottom et ils donnent des résultats très fins.

Code CSS :

```
p {
font-size:36px;
text-decoration:underline; /* souligné*/
}

.petit {
font-size:9px; }

.exposant {
vertical-align:super;
}

.haut {
vertical-align:top;
}

.milieu {
vertical-align:middle;
}

.normal {
vertical-align:baseline;
}

.bas {
vertical-align:bottom;
}

.indice {

vertical-align:sub;
}
```

Code XHTML :

```
<p>REFERENCE <span class="petit exposant "> super</span></p>
<p>REFERENCE <span class="petit haut "> top</span></p>
<p> REFERENCE <span class="petit milieu"> middle</span></p>
<p> REFERENCE <span class="petit normal"> baseline</span></p>
<p> REFERENCE <span class="petit bas"> bottom</span></p>
<p> REFERENCE <span class="petit indice"> sub</span></p>
```

Dans notre exemple, on a mis un texte de 36px comme référence, et un texte en plus petit pour montrer chaque position. Les propriétés définissent alors selon leur valeur chacune des positions du texte en petite taille par rapport au grand.

L'indentation

L'indentation est l'équivalent du retrait dans les traitements de texte. Il s'agit de mettre le début d'un paragraphe un peu plus à droite par rapport à la normale. La propriété text-indent est utilisée

pour cela. On attribue comme valeur à cette propriété la taille du retrait. La taille peut être indiquée en px, etc.

Code CSS :

```
.indentation {
text-indent: 40px; /* Les paragraphes commenceront 40 pixels de la droite */
font-size:12px;
}

<p>Paragraphe sans indentation</p>
<p class="indentation">Paragraphe avec indentation. </p>
```

Mise en forme des listes via CSS

En XHTML, on a appris les différents types de liste et les différentes balises pour les définir.

- Liste non numérotée
- Liste numérotée
- Liste alphabétique

Maintenant, nous allons nous pencher plutôt vers la mise en forme de ces listes.

Les listes à puces

Les listes à puces sont des listes non numérotées. A la place des numéros, on met des petits symboles appelés 'puces'. En CSS il est possible d'avoir différentes formes de puces grâce à la propriété list-style-type. Les formes varient en fonction de la valeur de la propriété, en voici quelques exemples :

- list-style-type: disc /* petit cercle plein */
- list-style-type: circle /* petit cercle vide */
- list-style-type: square /* petit carré plein */

La valeur par défaut de cette propriété est none.

Code CSS :

```
.disc {
list-style-type:disc
}

.circle
{ list-style-type:circle;
}

.square {
list-style-type:square;
}
```

Code XHTML :

```
<ul class="disc" >
  <li>liste 1 avec disc</li>
  <li>liste 2 avec disc</li>
</ul>

<ul class="circle" >
  <li>liste 1 avec circle</li>
  <li>liste 2 avec circle</li>
</ul>

<ul class="square" >
  <li>liste 1 avec square</li>
  <li>liste 2 avec square</li>
</ul>
```

Les listes numérotées et alphabétiques

Pour faire des listes numérotées et alphabétiques il suffit de remplacer < ul > par < ol> en XHTML. Toujours avec la propriété list-style-type on peut mettre des types de chiffre ou de caractère de notre choix comme des :

- Chiffres romains minuscules (list-style-type: lower-roman)
- Chiffres romains majuscules (list-style-type: upper-roman)
- Numérotation normale (list-style-type: " decimal type")
- Alphabet latin majuscule (list-style-type: upper-roman)
- Alphabet latin minuscule (list-style-type: lower-roman)

Code XHTML :

```
<ol style="list-style-type: lower-roman;" >
  <li>liste 1 avec chiffre romain en minuscules</li>
  <li>liste 2 avec chiffre romain en minuscules</li>
</ol>

<ol style="list-style-type: upper-roman;" >
  <li>liste 1 avec chiffre romain en majuscules</li>
  <li>liste 2 avec chiffre romain en majuscules</li>
</ol>
```

Listes avec images

A la place des puces et des numéros, il est aussi possible de mettre des images. Pour cela, utilisez la propriété list-style-image en mettant l'URL de l'accès local. Au cas où vous voulez désactiver l'affichage de l'image mettez none à la place de l'URL.

Code CSS :

```
.listeimage1 {
list-style-image: url("images/liste.gif");
}
```

Code XHTML :

```
<ul class="listeimage1">
  <li>liste 1 avec image en locale</li>
  <li>liste 2 avec image en locale</li>
</ul>
```

Le chemin de l'image peut aussi contenir des URL comme dans cet exemple :

Code CSS :

```
.listeimage2 {
list-style-image:url("http://www.votredomain.com/images/liste.gif" )
}
```

Code XHTML :

```
<ul class="listeimage2">
  <li>liste 1 avec image en ligne</li>
  <li>liste 2 avec image en ligne</li>
</ul>
```

Les navigateurs n'acceptent pas tous la propriété list-style-image. Afin de préserver l'affichage de la liste malgré tout, on peut ajouter un autre style liste qui servira de "style de secours". Ainsi, on ajoute dans le code, en bas de l'insertion d'image cette ligne : list-style-type: circle ; Des petits cercles s'afficheront à la place des images sur les navigateurs ne supportant pas la propriété list-style-image

Code CSS :

```
ul {
font-size:36px;
list-style-image: url (liste.gif);
list-style-type: circle ;
list-style-position: inside;
}
```

Position de la liste

list-style-position est la propriété pour spécifier l'emplacement de la liste par rapport à la bordure et au texte. Il peut y avoir deux valeurs possibles :

- outside qui est la valeur par défaut
- inside

Code CSS :

```
.bordure {
border: medium double red;
}

ul.inside {
font-size:36px;
list-style-type: circle ;
list-style-position: inside;
}
ul.outside {

font-size:36px;
list-style-type: circle;
list-style-position: outside;
}
```

Code XHTML :

```
<div class="bordure">
<ul class="inside" >
  <li>liste 1 inside</li>
  <li>liste 2 inside</li>
</ul>

<ul class="outside" >
  <li>liste 1 outside</li>
  <li>liste 2 outside</li>
</ul>
</div>
```

Observez dans l'exemple suivant que les listes avec list-style-position: inside; sont un peu plus décalées de la bordure gauche du div que les listes avec list-style-position: outside ;

Gestion des couleurs en CSS

Faisons encore quelque chose de plus joli avec le CSS, utilisons les couleurs. Déjà, il vous vient très certainement à l'esprit, l'idée de vouloir colorer des textes. Comment faire ? Il y a juste à suivre une syntaxe très simple. Utiliser la propriété color et indiquer la valeur de la couleur.

Code CSS :

```
h3 {
color : blue ;
}
```

Seulement, il y a différentes façons d'indiquer les couleurs en CSS. Il nous sera très utile de savoir comment ça marche alors allons-y. Voyons tout de suite la méthode la plus simple : la définition par mots clés.

Couleur et mots-clés

Il existe 17 couleurs qui peuvent être nommées directement par leur nom anglais. Ce sont ces noms là qu'on appellera ici mots-clés. Ces mots-clés correspondent à une couleur bien précise, désignée entre parenthèses :

- maroon (#800000)
- red (#ff0000)
- orange (#ffA500)
- yellow (#ffff00)
- olive (#808000)
- purple (#800080)
- fuchsia (#ff00ff)
- white (#ffffff)
- lime (#00ff00)
- green (#008000)
- navy (#000080)
- blue (#0000ff)
- aqua (#00ffff)
- teal (#008080)

- black (#000000)
- silver (#c0c0c0)
- gray (#808080)

Code CSS :

```
h1 {
color : red ;
}
```

Voyons maintenant comment définir les autres couleurs qui ne font pas partie de ce groupe.

Notation hexadécimale des couleurs

Dans la liste ci-dessus, on voit à côté de chaque mot-clé, la notation hexadécimale des couleurs. Ces valeurs là sont utilisées aussi pour définir des couleurs. Et surtout pour les couleurs qui ne peuvent pas être spécifiées par leur nom anglais.
Il s'agit de la notation hexadécimale. **C'est une combinaison de lettres et de chiffres qui indiquent une couleur**. Elle est composée de 6 caractères qui contiennent des lettres de 0 à 9 ou des chiffres de A à F. Ces caractères sont toujours précédés du signe #.
Ces lettres ou chiffres fonctionnent deux par deux. Les 2 premiers indiquent une quantité de rouge, les 2 suivants une quantité de vert, et les 2 derniers une quantité de bleu. En effet, toute couleur est issue de la combinaison des 3 couleurs de base : rouge, vert et bleu. En mélangeant une quantité de chacune de ces composantes on peut obtenir la couleur que l'on désire.
Code CSS :

```
h1 {
color : #ffA500;
}
```

Ceci permet d'avoir un titre de couleur orange.

Code couleur RGB

Il y a aussi une autre annotation des couleurs qui est le RGB. Cela consiste à définir les couleurs avec des valeurs décimales comprises entre 0 et 255. Sachant que R signifie Red, G signifie Green et B signifie Blue (respectivement 'rouge', 'vert', 'bleu'). La première valeur dans la notation RGB est celle des composantes rouges de la couleur.<.p>
La seconde valeur spécifie la teneur des couleurs vertes dans la couleur désirée. Et la troisième, est la valeur des composantes bleues dans la couleur en question.
Si vous voulez que le texte de l'élément p soit en bleu, vous noterez :
Code CSS :

```
p {
color : rgb(0,0,255) ;
}
```

On donne comme valeur à la propriété color, rgb avec les 3 valeurs décimales qui seront tous les 3 mises entre parenthèses. Les trois valeurs contenues dans les parenthèses sont séparées par une virgule.

Utiliser les pourcentages

On peut aussi indiquer en pourcentages la valeur des composantes rouges, vertes et bleues d'une couleur. Il suffit juste de copier ces valeurs dans vos codes CSS comme ceci :
CSS

```
p {
color : rgb(0%,0%,100%) ;
}
```

L'intérêt de la notation RGB c'est que dans certain logiciel comme paint ou photoshop par exemple, il y a une palette de couleur où on peut spécifier les valeurs RGB. Mieux encore, le navigateur Mozilla Firefox possède une extension qui s'appelle colorZilla.
Si vous avez ce navigateur, vous pouvez bien sûr l'installer gratuitement. **Ainsi, lorsque vous surfez sur le net et que vous êtes impressionné par une couleur que vous venez de voir, il vous suffit de pointer le curseur sur la couleur et la valeur RGB s'affiche en bas de la page**. Notez-la, et insérez-la dans vos codes CSS quand vous voulez l'utiliser.

Gérer le fond (background) en CSS

En CSS, il est possible non seulement de mettre un fond sur une page web mais aussi de mettre des fonds sur une partie de la page (les titres, les paragraphes, etc.) Les propriétés pour les fonds sont généralement précédées du mot background. On peut mettre comme fond une couleur mais aussi une image.

La couleur de fond

La propriété background-color sert à mettre une couleur de fond. Pour mettre un fond vert sur toute la page web par exemple, on utilise des styles dans la balise body.

Code CSS :

```
body {
background-color: green; /* fond de page en vert
color: black; /* couleur du texte en noir */
}
```

Si on ne désire mettre du fond que sur une partie, il suffit de créer une classe .fondcouleur par exemple. Et de l'appliquer sur l'élément.

Code CSS :

```
.fondcouleur {
background-color: yellow;
color: green ;
}
```

Code XHTML :

```
<h1 class="fondcouleur">Titre </h1>
Notre titre en h1 sera surligné en jaune, alors que le texte sera vert.
```

Les images de fond

La propriété background-image nous permet de placer une image en fond de la page ou en fond de texte d'un ou plusieurs éléments. **Il suffit de spécifier une image d'arrière-plan en indiquant son chemin**. Pour une image fond.jpg se trouvant dans un répertoire image du site, on aura donc :

Code CSS :

```
body {
background-image: url("image/fond.jpg");
}
```

On donne le chemin de l'image comme valeur de la propriété background-image. Il est nécessaire de bien formater les textes et les données des couleurs pour qu'ils soient compatibles avec l'arrière plan. Sinon ils risquent de ne pas être visibles.

Image de fond : fixe ou pas

Lors du défilement du texte quand on clique sur la barre de défilement, il est possible de rendre l'image de fond immobile. C'est-à-dire que quel que soit le mouvement de la scrollbar, seuls les textes bougent. L'image de fond reste fixe. Le nom de la propriété CSS pour avoir ce fond fixe est background-attachment et il peut prendre 2 valeurs :

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

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

Mettez des paragraphes très longs dans votre code XHTML pour bien voir le résultat. Essayez de faire défiler les barres de défilement. Vous verrez que le fond ne bouge pas avec le texte.

Répétition de l'image de fond

Parmi les propriétés CSS pour les fonds, il y en a une qui est très particulière parce qu'elle permet de gérer la répétition de l'image de fond. Il s'agit de la propriété background-repeat. Ses valeurs possibles sont les suivantes :

- **no-repeat** : le fond ne sera pas répété
- **repeat-x** : le fond sera répété uniquement sur la première ligne, horizontalement.
- **repeat-y** : le fond sera répété uniquement sur la première colonne, verticalement.
- **repeat** : le fond sera répété, horizontalement et verticalement. (valeur par défaut).

```
body {
background-image: ("/images/fond.jpg");
background-repeat: repeat-y;
}
```

Ce code CSS donnera une page avec un fond répété verticalement.

Positionnement d'une image d'arrière-plan

On peut aussi avoir différentes façons de positionner l'image d'arrière-plan. Ce qui peut s'avérer très intéressant lorsqu'on n'a qu'une image unique sur la page. C'est à dire lorsqu'on a mis un fond qui ne se répète pas "background-repeat: no-repeat;" **La propriété background-position est utilisée si on veut indiquer la position du fond par rapport au coin supérieur gauche de l'élément**. L'élément correspond à la page si le fond est appliqué sur la balise body. Mais il peut aussi s'agir d'un paragraphe.
Pour mettre un fond placé à 40 pixels de la gauche et 50 pixels du haut, la propriété aura comme valeur : background-position:40px 50px; Il est aussi possible d'indiquer les valeurs à l'aide de mots clés suivants :

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

Ensuite, il suffit de combiner les mots comme on veut. Par exemple, background-position: top right ; permet d' aligner une image en haut à droite.
Mettre une image en fond rend la page plus jolie, mais il faut le faire avec modération. En fait, si l'image de fond est très lourde, le temps de chargement de votre page risque d'augmenter. **Plus la page met de temps à se charger, plus le visiteur risque de s'en aller avant la fin du chargement.**
Notez qu'il y a différentes façons de traiter les images pour qu'elles soient le moins lourdes possible. Enfin, mettez une image qui convienne à la couleur de votre texte, afin que ce dernier reste bien lisible.

Les styles de bordure

On utilise la propriété border-style pour définir le style de bordure. Il y a différentes valeurs possibles :

- **none** : pas de bordure
- **solid** : un trait simple
- **dotted** : pointillés
- **dashed** : tirets
- **double** : bordure double
- **groove** : en relief
- **ridge** : effet 3D
- **inset** : Effet 3D mais le block est vu comme un creux
- **outset** : Effet 3D mais le block a l'air d'être surélevé)

La plus simple et la plus courante des bordures est de style solid. C'est juste pour mettre un trait simple.
Code CSS :

```
.borduresimple {
border-style : solid ;
}
```

Code XHTML :

```
<p class=" borduresimple ">
J'aime le css et le XHTML. C'est vraiment génial de pouvoir modifier le design du site comme on veut.
</p>
```

Après vous avez le choix. Une bordure avec des pointillés par exemple peut vous être utile.
Code CSS :

```
.pointille {
border-style : solid ;
}
```

Code XHTML :

```
<p class=" pointille">
J'aime le CSS et le XHTML. C'est vraiment génial de pouvoir modifier le design du site comme on veut.
</p>
```

Couleur de bordure

On a vu plus haut des bordures en noir, couleur par défaut. En donnant une valeur à la propriété border-color, on peut avoir d'autres couleurs. Après, il suffit de spécifier une couleur soit par son mot clé ("green", "red"...) , soit par sa valeur hexadécimale (#FF0000), soit par une valeur rgb (rgb(0, 21, 7), comme l'on a vu dans les autres tutoriels. Pour colorer la bordure en vert par exemple, on met comme mot clé 'green'.

Code CSS :

```
.couleur {
border-style : solid;
border-color : green;
}
```

Code XHTML :

```
<p class=" couleur"> Voici un paragraphe encadré en vert.
</p>
```

Epaisseur de la bordure

Vous avez aussi la possibilité de spécifier l'épaisseur de la bordure. Si vous désirez avoir une bordure plus épaisse (20px par exemple) ou plus fine (2px par exemple), l'exemple suivant vous montrera comment faire.

Code CSS :

```
.fine {
border-width : 1px;
border-color : green;
border-style : solid;
}

.epais{
border-width : 20px;
border-color : green;
border-style : solid;
}
```

Code XHTML :

```
<p class=" fine">
Voici un paragraphe avec une bordure fine.
</p>
<p class=" epais">
Voici un paragraphe avec une bordure épaisse.
</p>
```

Si vous ne voulez pas vous casser la tête avec les chiffres, pour avez la possibilité de mettre juste des mots-clés comme valeurs :

- **thin** : pour avoir une bordure fine
- **medium** : pour avoir une bordure moyenne
- **thick** : pour avoir une bordure épaisse

Bordure juste sur les 2 côtés

D'habitude quand on met une bordure, c'est pour avoir un cadre de 4 côtés. Mais le contexte des bordures ne se limite pas à cela. **On peut avoir un autre style en mettant juste sur 2 côtés par exemple**. Pour mettre une bordure en haut, utilisez border-top, alors que pour le bas, utilisez border-bottom. En suivant la logique border-left, c'est pour les bordures de gauche et border-right pour celle de la droite.

Code CSS :

```
.hautbas {
border-top:1px solid #FF9900 ;
border-bottom:1px solid #00CC00 ;
}

.gauche {
border-left:5px double #333333 ;
}

.droitegauche {
border-right:4px groove #006699 ;
border-left:4px groove #FF0000;
}
```

Code XHTML :

```
<p class="hautbas">
J'aime le CSS et le XHTML. C'est vraiment génial de pouvoir modifier le design du site comme on
veut.<br /> Voici un paragraphe avec une bordure orange en haut et verte en bas.
</p>
<p class="gauche">
J'aime le CSS et le XHTML. C'est vraiment génial de pouvoir modifier le design du site comme on veut.
<br /> Voici un paragraphe avec une bordure grise à gauche.
</p>
<p class="droitegauche">
J'aime le CSS et le XHTML. C'est vraiment génial de pouvoir modifier le design du site comme on veut.
<br /> Voici un paragraphe avec une bordure rouge à droite et bleue à gauche.
</p>
```

Toutes ces propriétés fonctionnent comme border mais la seule différence c'est que chacune d'elles ne s'applique qu'à un seul côté.

Enlever la bordure

La valeur par défaut de la propriété border-style est none. Donc si vous omettez cette propriété dans la définition du style, vous ne verrez pas une bordure.
A l'inverse, cette valeur peut être très utile dans certains cas. **Quand on a parlé des liens enXHTML, on a vu que lorsqu'une image est transformée en lien, elle sera encadrée automatiquement par une bordure bleue**. En mettant la valeur none pour la propriété border de toutes les images cliquables, on peut enlever cette bordure bleue.

Code CSS :

```
/* Enlever les bordures de toutes les images cliquables */
a img {
border: none;
}
```

Code XHTML :

```
<p>
<a href="#"><img src="bordure.jpg" alt="suivant" title="Cliquer ici" /></a>
</p>
```

Relooker des liens grâce à CSS

Les styles de cette page de tutoriel n'ayant pas encore été mis à jour, les exemples ne sont pas fonctionnels
Avec le CSS, on peut présenter des nouveaux effets magiques sur les liens. Ceci peut aller du soulignement au non soulignement jusqu'au changement de couleur au survol du pointeur de la souris. C'est ce que nous allons illustrer dans cette partie.

La propriété text-decoration

Comme son nom l'indique la propriété text-decoration est faite pour décorer un texte. Grâce à elle, on a différentes manières de souligner le texte et on peut aussi choisir de ne pas le souligner. Les valeurs applicables sont :

- overline = le texte est souligné en dessus
- underline = le texte est souligné dessous
- line-through = le texte est barré
- blink = le texte clignote (valable que sur Netscape et Firefox)
- none = soulignement annulé et valeur par défaut

La valeur BLINK n'est pas encore applicable sur Internet Explorer donc mettons-le de côté pour l'instant.

Code CSS :

```
p {
font-family: arial, verdana, sans-serif;
font-size: 14pt;
}

.dessus {
text-decoration: overline ;
}

.barre {
text-decoration: line-through;
}

.dessous {
text-decoration: underline ;
}
```

Code XHTML :

```
<p class="dessus"> DESSUS</p>
<p class="barre"> BARRE</p>
<p class="dessous"> DESSOUS</p>
```

Des liens non soulignés via CSS

Habituellement, les liens sont soulignés et en bleu en XHTML. Le pouvoir du CSS nous permet de changer le monde.

C'est là qu'intervient la valeur none de text-décoration parce qu'elle permet d'annuler le soulignement automatique du lien. Mais ce n'est pas tout. Désormais on sait comment colorer les textes avec les styles alors pourquoi ne pas colorer les liens ? Essayons toute suite un exemple pour que cela ne soit pas que des illusions.

Code CSS :

```
a {
text-decoration: none;
color: red;
}
```

Code XHTML :

```
<a href="#">texte de lien magique</a>
```

Les liens ne sont plus en bleu, ils sont en rouge et non soulignés.

Les pseudo-classes CSS et les liens

Maintenant que l'on a acquis quelques expériences, nous pouvons passer à des choses plus complexes. **En utilisant les pseudo-classes ou pseudo-formats avec les liens hypertexte, il est possible de changer l'aspect visuel des textes servant de support à ces liens**. Ils sont toujours associés au sélecteur "a". Autrement dit, on peut faire ce qu'on appelle des liens dynamiques. On dit lien dynamique parce qu'on peut changer leur aspect dès qu'un événement se passe.

Afin de voir cela plus clairement, on a subdivisé l'utilisation du pseudo-format en 2 parties pour les liens :

Les Pseudo-format d'ancre

Le pseudo-format d'ancre permet de savoir si les pages associées aux liens ont été visitées ou pas. Cette notion est très pratique dans un sommaire par exemple. **Par défaut, le navigateur colore le lien en violet pour dire que la page cible a été vue**. On n'est pas obligé de se limiter à cette couleur. En utilisant le pseudo-format a:visited, on peut appliquer les couleurs que l'on veut aux liens qui ont été visités. A l'inverse a:link spécifie la couleur des liens qui n'ont pas été encore visités.

Exemple : Cliquez, et je deviendrai rouge Ne cliquez pas et je resterai vert

Les Pseudo-classes dynamiques avec hover, active et focus

Au passage de la souris

La pseudo classe a:hover change l'apparence du lien quand l'utilisateur passe le pointeur de sa souris au-dessus du lien sans cliquer dessus.

```
a {
text-decoration: none; /* Les liens ne seront plus soulignés */
color: red; /* Les liens seront en rouge au lieu de bleu */
font-size:24px }
```

Ci-dessous, le texte du lien sera surligné en jaune, et coloré en vert quand on pointera dessus.

```
a:hover {
background-color: #FFFF00;
color: green;
}
```

Sur clic

On peut aussi changer l'apparence du lien quand l'utilisateur clique sur le lien. C'est-à-dire entre le moment où l'utilisateur presse le bouton de la souris et le relâche, à l'aide de la pseudo classe a:active.

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

Avec a:focus, le style s'applique lorsque le lien a le focus, c'est-à-dire lorsqu'il est sélectionné. Pour sélectionner un lien, il faut appuyer sur la touche 'TAB' jusqu'à ce qu'il soit sélectionné. A noter que ce style présente des problèmes d'interprétation.

Les pseudo-éléments CSS

Les pseudo-formats ont encore différentes applications possibles notamment au niveau des paragraphes. Sachant que ceux-ci sont valables aussi pour les autres balises comme les titres, etc. Essayons d'utiliser un style pour changer l'apparence de la première lettre et de la première ligne du texte d'un paragraphe à l'aide des pseudos formats.

Première lettre et première ligne d'un paragraphe

Comme son nom l'indique first-letter ou 'première lettre' est un pseudo-format permettant d'appliquer un style à la première lettre d'un groupe de texte. Créons une sorte d'attraction par exemple, qui permettra au visiteur de distinguer facilement la question aux différentes réponses. On marquera le début de la question en vert et le début d'une réponse en rouge.

Code CSS :

```
/* La première lettre de la question */
.question:first-letter {
font-weight: bold; /* En gras */
font-size: 4em;
color: green;
}

/* La première lettre de chaque réponse */
.reponse:first-letter {
font-weight: bold;
font-size: 4em;
color: red; }
p {
text-indent: 20px; }
```

Code XHTML :

```
<p class="question">
Question : Qu'est-ce que le XHTML ?
</p>
<p class="reponse">
Réponse : C'est un langage hybride entre le XML et le HTML
</p>
```

Styliser la première ligne : first-line

On peut faire pareil pour la première ligne d'un paragraphe mais cette fois-ci avec le pseudo-format : first-line. Cela permet de rendre le paragraphe un peu plus attrayant en mettant en majuscules par exemple la première ligne.

Code CSS :

```
p:first-line {
color: #000099; /*bleu*/
```

```
font-variant: small-caps ; /*majuscule*/
}
```

Code XHTML :

```
<p>
Vous pouvez utiliser le pseudo-élément : first-line pour ajouter un effet spécial à la première ligne d'un paragraphe.
</p>
```

Pseudo éléments before et after

Les pseudo-éléments suivants pourront aussi être utiles à certain d'entre nous.

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

Ces pseudo-éléments nous permettent d'éviter les répétitions. En cas de répétition de plusieurs textes ou d'image au début et à la fin d'un texte par exemple, on peut utiliser ces pseudo-éléments sans problème. Dans cet exemple d'interview, afin d'éviter de répéter le nom des participants chaque fois qu'ils prennent la parole, on met le nom dans le content d'un pseudo élément : before. Pour pimenter un peu la sauce, un petit coup de crayon de couleur et un léger réajustement de taille seront plus classes.

Code CSS :

```
.thomas:before {
content: "Thomas Brown:";
text-decoration:underline;
font-size:2em;
color:green; }

.valerie:before {
content: "Valérie Cornel:";
text-decoration:underline;
font-size:2em;
color:red
}
```

L'attribut content permet de spécifier le contenu répété.

Code XHTML :

```
<p class="thomas">Comment vous-vivez votre vie de Star ? </p>
<p class="valerie">J'ai commencé à chanter depuis l'an 1998....</p>
<p class="thomas">N'y a-t-il pas un impact sur votre vie privée? </p>
<p class="valerie">C'est très difficile mais j'essaie de bien m'organiser....</p>
```

De la même manière on peut utiliser le pseudo élément after en cas de besoin. On peut même combiner les deux.

Les boîtes en CSS

Ceux qui ont étudié le XHTML savent très bien qu'il y a deux sortes de balises : les balises de type inline et les balises de type block. En CSS, les éléments de type bloc comme div, ul, form, blockquote, pre, servent souvent à la mise en page générale de la page. **On peut donc créer de grands rectangles qu'on appellera désormais les boîtes**. Ils peuvent contenir soit d'autres éléments du même type, soit des éléments de type "en-ligne"
Les éléments de type "en-ligne" sont juste des conteneurs plus petits que bloc, avec un comportement spécifique. Parmi ces éléments on distingue par exemple A, B, IMG, INPUT, U, SPAN
Grosso modo, les types bloc peuvent contenir des types inline. Par contre, les types inline ne doivent pas contenir des types bloc.

Les balises Div et Span

Les balises div et span sont généralement conçues pour séparer la mise en forme du contenu. Grâce à elles on peut profiter de toute la gamme de propriétés qu'offre CSS pour la mise en forme.

La balise Div

La balise div est un élément de type bloc. Elle est utilisée pour faire des blocs de structure ou boîte. Ces blocs de structure peuvent être par exemple un ensemble de paragraphes. **Avec cette balise on va pouvoir modifier l'aspect d'un ou plusieurs paragraphes et les mettre dans une sorte de boîte**. Ainsi, souvent quand on parle de bloc ou boîte, on fait référence à un div.

Les div servent généralement à positionner un élément. C'est pourquoi on donne souvent à un div une identité unique à l'aide de l'attribut id. Mais on peut aussi lui assigner des classes si l'on veut utiliser plusieurs fois le même type de bloc dans une même page.

Code CSS :

```
.boitenoire{
background-color:#000000;
color: #FFFFFF;
font-size:24px;
width:6cm;
height:4cm;
}
```

Code XHTML :

```
<div id="boitenoire">boîte noire</div>
```

La balise Span

Si la balise div permet de spécifier un bloc de texte en entier, la balise span s'applique pour une lettre, un mot ou quelques mots seulement. Autrement dit, les balises ... servent à modifier un morceau de texte contenu dans des div.

C'est une balise de type inline et on l'utilise souvent pour souligner l'importance d'un mot ou d'un morceau de texte. C'est vrai qu'il y a des balises comme strong qui servent particulièrement à cela. Cependant, avec les span, il nous sera plus facile de modifier à la fois la couleur, la taille, la police d'écriture du texte. Il suffit d'utiliser des classes à l'intérieur de la balise.

Par exemple, pour surligner en vert un mot dans le div ci-dessous, voici ce qu'on doit faire :

Code CSS :

```
.boitespan {
border: solid 2px #990000 ;
}

.stylespan {
background-color:#00FF00;
text-decoration: underline;
font-weight: bold;
}
```

Code XHTML :

```
<div class="boitespan">Ce <span class="stylespan">mot</span> sera surligné en vert, mis en gras et souligné pour marquer son importance.</div>
```

Les marges

Avant de commencer, notons bien tout d'abord que deux notions de marges seront abordées dans ce chapitre :

- La marge externe définit l'espace entre le bloc et son extérieur
- La marge interne définit l'espace entre le bloc et son contenu

Notons également qu'un bloc désigne un élément de type bloc comme le paragraphe, les div, etc.

Les marges externes avec margin

La "marge externe" est la zone "margin". D'où le nom de la propriété : MARGIN. **Cette dernière spécifie d'un coup la valeur de la marge pour les quatre côtés à la fois**. On peut mettre comme valeur une longueur pour définir une largeur fixe. Il est aussi possible de mettre un pourcentage.

Dans ce cas, on mettra le pourcentage calculé par rapport à la taille du bloc conteneur. Et une dernière possibilité sera de mettre auto qui correspond à une valeur automatique définie par le navigateur. Si on ne spécifie qu'une valeur dans la propriété margin, celle-ci s'applique à tous les côtés.

Code CSS :

```
.margin {margin: 4cm;}
```

Code XHTML :

```
<p>Ceci est un paragraphe sans marge </p>
<p class="marge">Ceci est un paragraphe avec marge </p>
<p>Ceci est un paragraphe sans marge </p>
```

Celui-ci va provoquer un retrait de 4cm par rapport à l'extérieur dans tous les côtés du bloc.
S'il y a deux valeurs, alors la première valeur s'applique pour la marge du haut et celle du bas, et la seconde pour la marge droite et celle de gauche.
Code CSS :

```
/*retrait haut+bas et droite+gauche*/
.margin{ margin : 3cm 10cm ; }
```

En cas de 3 valeurs, la première est celle de la marge du haut. La deuxième est celle des marges gauche et droite. La marge du bas est définie par la troisième.
Code CSS :

```
.marge { margin: 2px 10cm 4px ; }
```

Si les quatre valeurs sont spécifiées, celles-ci s'appliquent respectivement aux marges du haut, de droite, du bas et de gauche.
Code CSS :

```
.marge { margin: 2cm 10cm 4cm 6cm;}
```

Il est parfois possible (sauf pour width) d'employer des valeurs négatives. **Seulement la définition de la marge doit être utilisée avec précaution sinon on aura des problèmes au niveau de l'affichage**. En mettant une valeur margin= 0 dans la balise body par exemple, on risque de nuire à la lisibilité de la page. En fait cela coïncide avec la bordure de la fenêtre du navigateur et donc le contenu sera collé sur le bord.

Les marges internes avec padding

Le principe reste le même que tout à l'heure sauf qu'à la place de margin, on utilise padding. Le padding est une zone, appelée aussi "marge interne" ou "intervalle". Elle est utilisée pour entourer de marge le "contenu" à l'aide de 4 valeurs. Concrètement, ça sert à définir une distance entre le contenu d'une boîte et la bordure de ce dernier.
Code CSS :

```
.padding {
border: solid 2px #990000;
padding: 50px;
width: 350px;
text-align: justify;
}
```

En faisant un petit rappel sur les bordures, on a vu deux cas. Il était possible de définir la bordure sur les quatre côtés ou la définir séparément sur chacun des côtés.
Il en est de même pour les marges internes. **On a pu définir à la fois le retrait en haut, en bas, à gauche et à droite grâce à la propriété margin ou padding**. Une autre façon serait aussi de les définir séparément. Il suffit de rajouter les mots clé 'top, bottom, left, right' aux margin ou aux padding.

Liste des propriétés de margin

- **margin-top** : marge extérieure en haut
- **margin-bottom** : marge extérieure en bas
- **margin-left** : marge extérieure à gauche
- **margin-right** : marge extérieure à droite

Liste des propriétés de padding

- **padding-top** : marge intérieure en haut
- **padding-bottom** : marge intérieure en bas
- **padding-left** : marge intérieure à gauche
- **padding-right** : marge intérieure à droite

Gestion des dimensions des boîtes en CSS

Généralement quand on parle de bloc, on fait référence à un div. **Mais tout ce qui suit est aussi valable pour d'autres balises de type bloc comme les paragraphes par exemple**. Parfois on veut avoir des petites boîtes, parfois on en veut des grandes. Serait-il possible de fixer la dimension des boites ? La réponse est oui. Un bloc a des dimensions précises que l'on peut spécifier.

Il a une largeur et une hauteur définis par 2 propriétés CSS :

- **width** : pour spécifier la largeur du bloc
- **height** : pour spécifier la hauteur du bloc

Par défaut, un bloc prend 100% de la largeur disponible. Voyons un exemple de div qui n'utilise aucun style:
Code XHTML :

```
<div>
Généralement quand on parle de bloc, on fait référence à un div. Mais tout ce qui suit est aussi valable pour d'autres balises de type bloc comme les paragraphes par exemple. Parfois on veut avoir des petites boîtes, dès fois on en veut des grandes. Serait-il possible de fixer la dimension des boîtes ? La réponse est oui. Un bloc a des dimensions précises que l'on peut spécifier. </div>
```

Maintenant, avec le même code on ajoute de la largeur et de la bordure pour bien voir le changement.
Code CSS :

```
div {
border: solid 2px #000099;
}

.largeur {
width: 60%;
text-align: justify;
}
```

Code XHTML :

```
<div class="largeur"> Généralement quand on parle de bloc, on fait référence à un div. Mais tout ce qui suit est aussi valable pour d'autres balises de type bloc comme les paragraphes par exemple. Parfois on veut avoir des petites boîtes, parfois on en veut des grandes. Serait-il possible de fixer la dimension des boîes ? La réponse est oui. Un bloc a des dimensions précises que l'on peut spécifier. </div>
```

Dans l'exemple ci-dessus, le div avec une largeur de 60% n'occupe plus que 60 % de la page. On a utilisé le pourcentage mais on peut mettre aussi des tailles exactes en px, etc... comme l'on a fait dans les tutoriels précédents.

Overflow pour les balises de type bloc

Parfois il vous arrive de fixer une taille pour la boîte, mais vous n'avez pas le résultat que vous attendiez. **Dans le cas où le div contient des textes très longs par exemple, la boîte sera automatiquement élargie pour que le texte soit visible, quelle que soit la taille que vous avez spécifié**.
Cependant, si vous voulez quand même que la boîte respecte la dimension prescrite, même si le texte n'est pas affiché en totalité, vous pouvez utiliser la propriété overflow. Cette propriété peut prendre plusieurs valeurs :

- **visible** : C'est la valeur par défaut. Cela signifie que si le texte dépasse la taille spécifiée, la boîte s'élargit de manière à ce que tout soit visible.
- **hidden** : Cela permet de garder la dimension de la boîte et de masquer la partie de texte qui dépasse. En hidden, on ne peut pas voir tout le texte.
- **scroll** : Le texte sera caché s'il dépasse les limites. Sauf que le navigateur mettra en place des barres de défilement pour qu'on puisse voir tout le texte. Il met des barres de défilement verticales et horizontales même si on n'en a pas forcément besoin.
- **auto** : Le navigateur mettra les barres de défilement seulement s'il juge cela nécessaire.

Code CSS :

```
p {
width: 200px;
height: 100px;
text-align: justify;
}

.couper{
```

```
overflow: hidden;
}
```

Code XHTML :

```
<p class="couper">
Généralement quand on parle de bloc, on fait référence à un div. Mais tout ce qui suit est aussi valable pour d'autres balises de type bloc comme les paragraphes par exemple. Parfois fois on veut avoir des petites boîtes, parfois on en veut des grandes. Serait-il possible de fixer la dimension des boites ? La réponse est oui. Un bloc a des dimensions précises qu'on peut spécifier. </p>
```

Le positionnement en CSS

Nous avons appris qu'il y a deux types d'élément : in-line et block. Nous savons aussi que les inline peuvent être contenus dans des blocks alors que l'inverse n'est pas possible. **Jusqu'à présent, leCSS n'a jamais cessé de nous apporter des nouvelles astuces**. Cette fois-ci, il nous montre un autre moyen de différencier les deux types d'élément. Il s'agit de ce que l'on appelle le flux.
Le flux désigne l'ordre d'affichage par défaut des éléments. Par défaut, le navigateur affiche les éléments selon leur ordre d'apparition dans le code source. **Une des propriétés remarquables d'une balise de type inline c'est le fait de ne pas créer un retour à la ligne.** Par conséquent, les éléments du type inline s'affichent par défaut les uns à la suite des autres sur la même ligne. D'où leur désignation en anglais : inline qui signifie en ligne.
Code CSS (commun aux deux types d'affichage) :

```
.vert {
background-color:green; /* vert */
}

.rouge {
background-color:red; /* rouge */
}

.jaune {
background-color:yellow; /* jaune */
}
```

Code XHTML :

```
<span class="vert">tout sur</span> <span class="rouge">la même</span> <span class="jaune">ligne</span>
```

A l'inverse les types block s'affichent les uns en dessous des autres comme les mots japonais. Avant et après l'affichage d'un élément du type block, il se crée automatiquement un retour à la ligne. Exemple :
Code XHTML :

```
<div class="vert"> block1 ligne1</div>
<div class="rouge"> block2 ligne2</div>
<div class="jaune "> block3 ligne3</div>
```

Différentes propriétés CSS nous permettent de manipuler cet ordre de positionnement. C'est ce que nous allons voir dans la suite.

Changer le type

Une des possibilités de manipulation des positionnements est par exemple celle de changer les types des éléments. La propriété CSS pour modifier les types des éléments est display (='affiche').
Pour changer un élément de type block en inline il suffit de mettre
Code CSS (commun aux deux types d'affichage) :

```
Display :inline;
```

Et pour faire l'inverse, il suffit de mettre
Code CSS (commun aux deux types d'affichage) :

```
Display :block;
```

Généralement on voit que les titres s'affichent les uns à la suite des autres comme dans l'exemple suivant.
CSS commun aux deux types d'affichage :
Code CSS (commun aux deux types d'affichage) :

```
.vert {
background-color:green; /* vert */
}
```

```
.rouge {
background-color:red; /* rouge */
}

.jaune {
background-color:yellow; /* jaune */
}
```

Code XHTML :

```
<h1 class="vert">Titre vert </h1>
<h1 class="rouge">Titre rouge </h1>
<h1 class="jaune">Titre jaune </h1>
```

Ceci est parfaitement normal puisque la balise p est de type block.
Comme on est partisan du surréalisme, cherchons toujours l'extraordinaire avec le CSS. **Appliquons maintenant la propriété display à ces mêmes titres pour voir ce que ça donne quand on change de type**. On a juste à rajouter display : inline à la balise p. Et on aura les 3 paragraphes vert rouge jaune sur la même ligne
Code CSS (commun aux deux types d'affichage) :

```
.vert {
background-color:green; /* vert */
}

.rouge {
background-color:red; /* rouge */
}

.jaune {
background-color:yellow; /* jaune */
}

/* à rajouter*/
h1{
display: inline;
}
```

Code XHTML :

```
<h1 class="vert">Titre vert </h1>
<h1 class="rouge">Titre rouge </h1>
<h1 class="jaune">Titre jaune </h1>
```

Positionnement absolu, fixe et relatif

La propriété CSS position permet de changer le comportement d'affichage des blocs. La valeur par défaut de cette propriété est static qui définit l'affichage normal. Mais nous nous pencherons plutôt vers les autres propriétés qui s'avèrent intéressantes :

- position:relative;
- position:absolute;
- position:fixed;

Positionnement relatif

On a dit que le flux définit l'ordre d'affichage par défaut des éléments. **En positionnant de façon relative un élément, on ne le retire pas du flux**. Il suit toujours la règle normale des positionnements. En mettant position:relative ; l'élément se déplace simplement par rapport à sa position normale.
Code CSS :

```
.relatif{
background-color: yellow;
border: 2px solid black;
position: relative;
left: 50px;
top:10px;
}
```

Code XHTML :

```
<p> Ceci est un paragraphe normal.<span class="relatif"> Block relatif</span> Pourquoi ce texte "block relatif" est-il décalé? parce que c'est relatif; </p>
```

Positionnement absolu

Avec le positionnement absolu, l'élément est détaché complètement du flux. Sa position n'est plus définie par rapport à la normale. Si la page est considérée comme une sorte de repère, l'élément sera placé par rapport à ce repère suivant les coordonnées définies. Les propriétés bottom, left, right ou top définiront ses positions comme suit :

- **left** : par rapport à la gauche de la page
- **right** : par rapport à la droite de la page
- **top** : par rapport au haut de la page
- **bottom** : par rapport au bas de la page

Le positionnement absolu nous donne alors une possibilité plus étendue au niveau du placement des blocks. Grosso modo, il nous permet de placer un block à n'importe quel endroit de la page que ce soit en haut à droite, ou en bas à gauche, ou etc.

Positionnement Fixed

La propriété position:fixed pourrait nous être avantageuse dans certains cas. Un petit rappel, comme d'habitude : quand nous avons étudié les fond, **le background-attachment:fixed nous a permis de ne pas bouger le fond même si on fait défiler la page.** L'idée du positionnement fixe est identique.
On place un élément sur la page, il y reste visible même si on fait défiler la page. Un cas très pratique est le cas des menus. Si on veut par exemple qu'un menu reste visible quand le visiteur parcourt la page, on le met en position fixe sur une partie de la page.
Cependant, il y a quand même un petit souci. **Certains navigateurs non conformes aux standardsCSS2 ne prennent pas en compte cette propriété**. A tout problème il y a une solution. Mais ce thème ne sera pas abordé dans ce tutoriel. Si vous êtes confronté à ce problème, on vous laisse le soin de trouver la solution en question.
La dernière propriété c'est la position:inherit (CSS2) qui permet de dire que l'élément en question hérite de la propriété position de son parent. Si la position du parent est de type relatif, l'élément concerné sera aussi positionné de façon relative.

Flottement et superposition

Le positionnement nous offre encore d'autres possibilités afin d'organiser notre page. Après avoir vu le positionnement absolu, fixe et relatif dans le tutoriel précédent, voyons maintenant le positionnement flottant.

Les flottants

Maintenant nous allons apprendre comment faire flotter des objets. Vous avez par exemple une page pour présenter des personnes et vous voulez mettre leur photo d'identité. Sans faire flotter l'image vous auriez quelque chose comme ceci :
Il y a donc un espace blanc à côté de la photo. **Si vous voulez mettre du texte dans cet espace blanc pour entourer la photo comme dans les journaux, il vous faut mettre la propriété float**. Dans ce cas on dit que l'image 'flotte' (propriété : float) .
Code CSS :

```
.imageflottante {
float: right;
}
```

Puis qu'on parle de journaux, ça nous fait penser aux lettrines. **Pour ceux qui ne savent pas encore ce qu'est une lettrine, c'est la première lettre d'un paragraphe écrite en plus grande taille**. Ceux qui parmi vous ont suivi les cours sur les pseudo-classes savent très bien qu'il nous faut utiliser p:first-letter. En voilà bien une révision et application à la fois !
Code CSS :

```
p:first-letter {
float: left;
font-size: 3em;
font-weight: bold;
margin-right: 5px;
}
```

Effet de superposition Z index

Toujours sur les manipulations du positionnement, on veut maintenant avoir des effets de superposition pour des blocs qui se chevauchent. Pour vous donner une petite idée, c'est à peu près comme les histoires de calques sur logiciels de traitement d'images. Il faut surtout apprendre à gérer les superpositions, et les écrasements qui en résultent.
Pour cela, nous devons utiliser la propriété z-index. Elle va donner un numéro à chacun des éléments. Chaque élément se retrouvera écrasé par les éléments numéros suivants.
Code CSS :

```
.numero1 {
background-color:red;
position:absolute;
top:25px;
left:25px;
width:50px;
height:50px;
z-index:1;
}

numero2 {
background-color:green;
position:absolute;
top:50px;
left:50px;
width:50px;
height:50px;
z-index:2;
}

numero3 {
background-color:yellow;
position:absolute;
top:75px;
left:75px;
width:50px;
height:50px;
z-index:3;
}
```

Code XHTML :

```
<div id="numero1"></div>
<div id="numero2"></div>
<div id="numero3"></div>
```

Le numéro 1 (z-index=1) qui est en rouge est écrasé par le numéro 2 (z-index=2), en vert, le numéro 2 par le 3 (z-index=3) et ainsi de suite.

Des jolis formulaires et des styles

Les styles de cette page seront prochainement mis à jour afin de pouvoir consulter les résultats des codes proposés directement depuis ce tutoriel.
Bien que l'on ait déjà appris l'essentiel pour tout ce qui est formulaire, il ne faut surtout pas s'arrêter là. Autant continuer à approfondir le sujet, en essayant d'améliorer notre formulaire, afin de lui donner un peu plus de design.

Formulaire organisé et facilement accessible

Parfois vous êtes confrontés à une longue liste de saisie sur un seul formulaire (un questionnaire par exemple). **Si vous n'essayez pas de mettre un peu d'ordre dans le formulaire en question, le visiteur va se sentir un peu perdu**. Il vous faut alors connaître la balise XHTML qui permet de grouper les champs qui ont les mêmes thèmes.
Il s'agit de la balise <fieldset></fieldset>. **Le principe n'est pas aussi compliqué que ça : à l'intérieur de cette balise vous groupez tous les champs du même thème**. Ensuite vous donnez un titre à ce groupe de champ. Pour cela, il y a la balise <legend></legend> qui a pour rôle de spécifier le nom du groupe.
Code XHTML :

```
<form method="post" action="questionnaire.php">
<fieldset>
  <legend>Coordonnées</legend>
   <label for="titre">Titre : </label>
```

```
    <select name="titre">
     <option value="Mme" selected="selected">Madame</option>
     <option value="Mlle">Mademoiselle</option>
     <option value="Mr">Monsieur</option>
    </select>
    <br />
    <label for="nom">Nom:</label>
     <input type="text" name="nom" id="nom" />
    <br />
    <label for="prenom">Prénom:</label>
     <input type="text" name="prenom" id="prenom" />
    <br />
    <label for="email">Email:</label>
     <input type="text" name="email" id="email" />
    <br />
    <label for="tel">Téléphone:</label>
     <input type="text" name="tel" id="tel" /><br />
    <label for="adresse">Adresse:</label>
     <input type="adresse" name="adresse" id="adresse" />
    <br />
</fieldset>
<fieldset>
  <legend>Situation familiale:</legend>
  <p>
   Quel est votre statut?<br />
   <input type="radio" name="celibat" value="celibat" id="celibat" /> <label
for="celibat">Célibataire</label><br />
   <input type="radio" name="marie" value="marie" id="marie" /> <label
for="marie">Marié</label><br />
   <input type="radio" name="divorce" value="divorce" id="divorce" /> <label
for="divorce">Divorcé</label><br />
  </p>
  <p>
   Si vous avez des enfants, combien en avez vous? <br />
   <input type="radio" name="deux" value="deux" id="deux" /> <label for="deux">1-2
enfants</label><br />
   <input type="radio" name="quatre" value="quatre" id="quatre" /> <label for="quatre">3-4
enfants</label><br />
   <input type="radio" name="cinq" value="cinq" id="cind" /> <label for="cinq">5 enfants ou
plus</label><br /> </p>
  </fieldset>
   <fieldset>
  <legend>Centres d'intérêt:</legend>
  <p>
   Quels sont vos centres d'intérêt?<br />
   <input type="checkbox" name="musique" id="musique" /> <label
for="musique">musique</label><br />
   <input type="checkbox" name="politique" id="politique" /> <label
for="politique">politique</label><br />
   <input type="checkbox" name="informatique" id="informatique" /> <label
for="informatique">informatique</label><br />
   <input type="checkbox" name="cuisine" id="cuisine" /> <label for="cuisine">cuisine</label><br />
  </p>
</fieldset>
</form>
```

Vous pouvez voir à partir de cet exemple que même si le formulaire paraît long, il y a quand même un certain ordre et une certaine clarté au niveau de la présentation des informations. On a mis trois groupes de champ : Coordonnées, situation familiale et centre d'intérêt.

Décorer le formulaire avec des styles CSS

Le formulaire de tout à l'heure paraissait déjà bien organisé mais on peut faire plus. Avec un petit coup de mise en forme CSS, on pourra avoir un résultat plus remarquable.

Code CSS :

```
/* Police du texte tapé l'intérieur des champs et mettre une marge à gauche */ input, textarea, select {
font-family: "Times New Roman", Times, serif;
font-size:12px;
```

```
margin-left: 30px
}

label{
color: #663399; /* Colorer en bleu tous les labels (bah oui, pourquoi pas en bleu ?) */
font-weight: bold ;
}

/* Mettre les titres du groupe en caractères plus grands, en gras et spécifier une couleur*/
legend {

color: #663399;
font-weight: bold;
font-size:24px;
}

/* Mettre une séparation entre les fieldset et colorer l'arrière plan */
fieldset {
margin-bottom: 20px;
background-color: #FEDAD3;
}
```

Bien sûr, ce n'est qu'un exemple mais vous pouvez faire beaucoup mieux en exploitant toutes les propriétés CSS que l'on a vues jusqu'ici.
Jusque là, on avait appris à faire des champs de saisie, à décorer, etc. Mais un autre problème se pose : comment récupérer et traiter les valeurs saisies ? Rendez- vous dans le **cours de PHP** pour ceux qui veulent en savoir plus.

Les tableaux et les styles CSS

Là encore, on verra qu'il y a une étroite relation entre XHTML et CSS. En appliquant des styles CSSà des tableaux XHTML, on peut avoir des tableaux plus personnalisés, plus perfectionnés.
Pour vous rafraîchir la mémoire, prenons comme exemple le même tableau utilisé dans le **tutorielXHTML pour les tableaux**. Il s'agit du tableau des notes des élèves de la classe de 6ème. **On prendra la dernière version du tableau, on essaiera de lui mettre des styles comme les bordures, des couleurs des cellules, etc**. Allons-y, c'est parti !

Petit rappel et complément de XHTML

Avant toute chose, il est important que nous nous rappelions des principes que l'on a appris sur les tableaux en XHTML. On avait appris à faire des tableaux simples, à fusionner des cellules, à compartimenter un tableau, à lui mettre des légendes et à le rendre 'accessible' comme le principe de l'XHTML l'exige.
Toutefois on n'a pas encore appris comment rendre les tableaux plus jolis par exemple. Certes, en utilisant les CSS mais dans quel sens ?
Pour commencer apprenons à grouper les colonnes pour y appliquer ensuite des styles.
La balise col est contenue dans la balise colgroup. **Comme leur nom l'indique, elles servent toutes deux à grouper des colonnes afin d'y appliquer un style**. Il faut placer les balises après la balise caption si celle-ci est renseignée. Pour être plus exact, on les place avant tout le reste pour que tous les éléments du tableau bénéficient des styles que l'on va définir.
Code XHTML avec style en local :

```
<table summary="Notes des élèves de la classe de sixième année 2008-2009">
  <caption>Notes des élèves de la classe de sixième</caption>
   <colgroup>
    <col span="1" width="200" />
    <col span="3" style="background-color: #0099CC" />
    <col span="3" style="background-color: #FF9900" />
    <col span="3" style="background-color: #FFFF00" />
   </colgroup>
  <thead>
   <tr>
    <th rowspan="2">Nom</th>
    <th colspan="3" abbr="math">Mathématiques</th>
    <th colspan="3" abbr="sn">Sciences naturelles</th>
    <th colspan="3" abbr="phys">Physique </th>
   </tr>
   <tr>
    <th>Examen</th>
```

```
    <th>Devoir</th>
    <th>Moyenne</th>
    <th>Examen</th>
    <th>Devoir</th>
    <th>Moyenne</th>
    <th>Examen</th>
    <th>Devoir</th>
    <th>Moyenne</th>
   </tr>
  </thead>
  <tfoot>
   <tr>
    <th colspan="10">Résultats publiés le 01/09/2008</th>
   </tr>
  </tfoot>
 <tbody>
   <tr>
    <td>Assan Tessan</td>
    <td>10</td>
    <td>14</td>
    <td>12</td>
    <td>12</td>
    <td>18</td>
    <td>15</td>
    <td>13</td>
    <td>11</td>
    <td>12</td>
   </tr>
   <tr>
    <td>Bernard Chateau</td>
    <td>8</td>
    <td>10</td>
    <td>9</td>
    <td>8</td>
    <td>9</td>
    <td>8,5</td>
    <td>13</td>
    <td>6</td>
    <td>9,5</td>
   </tr>
   <tr>
    <td>Helmut Shröder</td>
    <td>12</td>
    <td>18</td>
    <td>15</td>
    <td>8</td>
    <td>9</td>
    <td>8,5</td>
    <td>13</td>
    <td>6</td>
    <td>9,5</td>
   </tr>
   <tr>
    <td>Willy Jons</td>
    <td>13</td>
    <td>6</td>
    <td>9,5</td>
    <td>8</td>
    <td>9</td>
    <td>8,5</td>
    <td>8</td>
    <td>10</td>
    <td>9</td>
   </tr>
  </tbody>
 </table>
```

La balise col contient un certain nombre d'attributs, particulièrement l'attribut span. Cet attribut permet de spécifier le nombre de colonnes sur lesquelles le style doit être appliqué. Ici, on a mis 3 comme valeur de l'attribut span pour que chacun des styles soit appliqué aux trois colonnes (puisqu'il y a 3 notes pour chaque matière).
Nous avons distingué ainsi les colonnes de chaque matière : les colonnes des notes de mathématiques seront en bleu, celles des notes de sciences naturelles en orange, et celles des notes de physique en jaune. **Sachant que pour mettre une couleur de fond dans une cellule, on utilise la propriété background-color**. Quant à la colonne des noms, on n'a pas mis de couleur mais on a juste spécifié sa largeur à 200. L'attribut scope
L'attribut scope sert à établir une relation entre la balise th et les cellules du tableau.
Il y a 4 valeurs applicables :

- **col** : la cellule sur laquelle est appliquée la propriété scope se rapporte à une colonne
- **row** : la cellule sur laquelle est appliquée la propriété scope se rapporte à une ligne
- **colgroup** : la cellule sur laquelle est appliquée la propriété scope se rapporte à tout le colgrou
- **rowgroup** : la cellule sur laquelle est appliquée la propriété scope se rapporte à tout le rowgroup

On le met dans la balise th comme suit :
Code XHTML :

```
<th colspan="3" abbr="math" scope="col" >Mathématiques</th>
```

Les bordures

L'attribut border sert à mettre une bordure externe à un tableau. La syntaxe est la même que celle de toutes les bordures : Pour une bordure bleue de 2px on a :
Code CSS :

```
.bordure {
border: #0099CC 2px solid;
}
```

Code XHTML :

```
<table summary="Notes des élèves de la classe de sixième année 2009" class="bordure">
```

Coller les bordures entre elles

On a vu une bordure à l'extérieur du tableau. Maintenant parlons de l'espacement des bords des cellules. La propriété border-collapse permet de dire si les cellules sont collées entre elles ou non. Cette propriété peut prendre 2 valeurs :

- **separate** : Les bordures seront dissociées par défaut.
- **collapse** : Cette valeur sous entend que les bordures seront collées entre elles. C'est-à-dire qu'il n'y a aucun espace entre les limites de chaque cellule.

En cas de cellules séparées (border-collapse: separate), il est aussi possible de définir une valeur exacte à l'espacement entre les cellules. En effet, la propriété border-spacing permet de spécifier la taille de l'espacement horizontal et vertical.
Par exemple pour border-spacing: 10px 5px, l'espacement horizontal entre les cellules est de 10 px, et l'espacement vertical entre les cellules est de 5px. Voici une exemple d'application de toutes ces propriétés :
Code CSS :

```
.bordure, td {
border: #0099CC 2px solid; /*bordure externe*/
border-collapse: separate;/*cellules séparées*/
border-spacing: 10px 5px; /*espacement horizontal : 10px et espacement vertical : 5px*/
}
```

Code XHTML :

```
<table summary="Notes des élèves de la classe de sixième année 2009 " class="bordure">
<caption>Notes des élèves de la classe de sixième</caption>
<colgroup>
   <col span="1" width="200"/>
   <col span="3" style="background-color: #00CCCC" class="bordure" />
   <col span="3" style="background-color: #FF9900" />
   <col span="3" style="background-color: #FFFF00" />
```

```
</colgroup>

<thead>
 <tr>
   <th rowspan="2">Nom</th>
   <th colspan="3" abbr="math">Mathématiques</th>
   <th colspan="3" abbr="sn">Sciences naturelles</th>
   <th colspan="3" abbr="phys">Physique </th>
  </tr>
  <tr>
   <th scope="row">Examen</th>
   <th>Devoir</th>
   <th>Moyenne</th>
   <th>Examen</th>
   <th>Devoir</th>
   <th>Moyenne</th>
   <th>Examen</th>
   <th>Devoir</th>
   <th>Moyenne</th>
  </tr>
</thead>

<tfoot>
  <tr>
   <th colspan="10">Résultats publiés le 01/09/2008</th>
  </tr>
</tfoot>
<tbody>
  <tr>
   <td>Assan Tessan</td>
   <td>10</td>
   <td>14</td>
   <td>12</td>
   <td>12</td>
   <td>18</td>
   <td>15</td>
   <td>13</td>
   <td>11</td>
   <td>12</td>
  </tr>
  <tr>
   <td>Bernard Chateau</td>
   <td>8</td>
   <td>10</td>
   <td>9</td>
   <td>8</td>
   <td>9</td>
   <td>8,5</td>
   <td>13</td>
   <td>6</td>
   <td>9,5</td>
  </tr>
  <tr>
   <td>Helmut Shröder</td>
   <td>12</td>
   <td>18</td>
   <td>15</td>
   <td>8</td>
   <td>9</td>
   <td>8,5</td>
   <td>13</td>
   <td>6</td>
   <td>9,5</td>
  </tr>
  <tr>
   <td>Willy Jons</td>
   <td>13</td>
```

```
      <td>6</td>
      <td>9,5</td>
      <td>8</td>
      <td>9</td>
      <td>8,5</td>
      <td>8</td>
      <td>10</td>
      <td>9</td>
    </tr>
  </tbody>

  </table>
```