Css [cascading style sheets]

UNIVERSITE GASTON BERGER DE SAINT-LOUIS

U.F.R. S.A.T.

DIETEL

(Diplôme d'Ingénieur en Electronique et Télécommunication)

GETI

(Génie en Electronique, Télécommunication et Informatique)

Introduction :

La création de styles CSS (Cascading Style Sheets ou feuilles de style en cascade) est le complément indispensable du langage XHTML. Ce procédé correspond parfaitement à la séparation du contenu et de la présentation d'une page. D'une part, cette séparation permet d'alléger les pages en centralisant les définitions des styles en un point unique, une seule définition pouvant s'appliquer à un grand nombre d'éléments. D'autre part, elle facilite également la maintenance et l'évolution des sites par voie de conséquence. Elle apporte aussi une plus grande rigueur dans la conception des pages et peut permettre un travail collaboratif entre plusieurs programmeurs travaillant en parallèle, d'où une réduction des délais de fabrication. À l'attention de ceux pour qui ces points peuvent paraître marginaux, nous pouvons ajouter que les styles CSS apportent une bien plus grande richesse créative que ne le permettait le langage HTML utilisé sans CSS.

Créer des styles :

Les règles générales :

Une déclaration de style comporte plusieurs parties, selon l'ordre suivant :

• Un sélecteur qui va déterminer à quel élément et éventuellement dans quelles conditions va s'appliquer le style. Autant que les propriétés, c'est la variété des sélecteurs qui fait la richesse de CSS.

• La déclaration des propriétés que l'on veut voir appliquées à l'élément sélectionné. Elle doit être incluse entre des accolades ouvrante ({) et fermante (}).

• Dans ces accolades doivent apparaître une ou plusieurs propriétés déterminées chacune par un mot-clé propre à CSS suivi du caractère deux-points (:), puis de la valeur attribuée à cette propriété. Si nous définissons plusieurs propriétés dans le même style, il faut séparer chaque déclaration de la précédente par le caractère point-virgule (;). Les propriétés sont en nombre limité et font l'objet d'une recommandation du W3C . La version actuelle de CSS est la version 2.1 (au 13 juin 2005), dans laquelle ont été éliminées un certain nombre de propriétés qui n'étaient mises en application par aucun navigateur. C'est cette version que nous emploierons ici. À chaque propriété correspond un domaine de valeurs particulier constitué par exemple de mots-clés ou de nombres. Signalons en fin que l'utilisation d'une propriété ou d'une valeur erronée ne provoque pas d'erreur lors de l'exécution comme ce serait le cas dans un langage de programmation. Ces fausses définissons sont simplement ignorées.

```
sélecteur { propriété1 : valeur1;  propriété2 : valeur2; }
```

Syntaxe d'écriture d'un style

Sur ce modèle, nous pouvons par exemple écrire le style suivant :

```
div {color : red ; background-color :yellow ;}
```

dans lequel div est le sélecteur, color est la première propriété qui détermine la couleur du texte de l'élément, red la valeur attribuée à cette couleur, background-color qui désigne la couleur de fond est la seconde propriété et yellow sa valeur. Tous les éléments de la page dans laquelle se trouve cette déclaration ont donc un contenu écrit en rouge sur fond jaune.

Si nous écrivons par exemple le code suivant :

```
h1,div,p {color : black ; background-color : red;}

div {margin : 20px;}
```

l'élément va avoir à la fois un texte noir, un fond rouge et une marge de 20 pixels, car la propriété margin, définis uniquement pour l'élément , s'ajoute à celles déjà définies pour le sélecteur d'éléments h1,div,p.

Le sélecteur universel :

Pour appliquer un style à tous les éléments, nous utiliserons le sélecteur universel * avant la définition d'une ou plusieurs propriétés. Par exemple, pour que la couleur du fond de tous les éléments soit le jaune, nous écrirons :

```
*{background-color : yellow;}
```

Cela n'empêche pas de modifier cette couleur de fond pour un élément particulier, en la redéfinissant uniquement pour celui-ci, par exemple :

```
*{background-color : yellow;}

p{background-color : gray;}
```

Dans ce cas, tous les éléments ont un fond jaune, sauf <p> qui a un fond gris redéfini spécialement.

Les classes :

Les éléments XHTML/HTML possèdent l'attribut class. Ce dernier permet d'appliquer un style défini dans une classe à un élément dont l'attribut class se voit attribuer le nom de cette classe. Pour créer une classe, le sélecteur est constitué du nom choisi pour la classe précédé d'un point (.). Le nom de la classe peut être un mot quelconque, en évitant quand même les noms des propriétés CSS et des éléments XHTML car cela occasionnerait des confusions. Nous pouvons par exemple définir la classe nommée *evidence* en écrivant le code :

```
.evidence {color : red;}
```

À ce stade, la classe est abstraite et ne s'applique à aucun élément. Pour mettre en évidence un paragraphe précis de la page avec un texte rouge, nous devons alors écrire dans le code XHTML :

```
<p class="evidence">Texte contenu du paragraphe</p>
```

Nous pouvons également définir une classe en la déclarant applicable seulement à un élément en faisant précéder son nom de celui de l'élément. Nous pouvons écrire par exemple :

```
div.jaune {color : yellow;}
```

Dans ce cas, seules les divisions ayant un attribut class dont la valeur est jaune ont un texte jaune. Les autres éléments, et même s'ils ont le même attribut avec la même valeur, n'ont pas de style défini dans cette classe.

Il est possible de définir d'abord une classe abstraite, puis de la particulariser en ajoutant une autre propriété pour un élément qui utilisera la même classe. Dans le code CSS ci-après :

```
.rouge {color : red;}

div.rouge {background-color : blue;}
```

suivi du code XHTML dans la page,

```
<div class="rouge">Texte contenu de la division </div>
```

le texte contenu dans l'élément est affiché en rouge sur fond bleu.

On peut aussi utiliser plusieurs classe abstraites à la fois :

```
<div class="classe1 classe2"> Ceci est un texte avec la classe 1 et 2 </div>
```

Sélecteur d'identifiant id :

Pratiquement, chaque élément peut avoir un attribut id qui doit être unique dans une page donnée. Nous pouvons écrire un style qui ne sera applicable qu'à l'élément dont l'id a une valeur précise en donnant cette valeur au sélecteur (comme pour une classe) et en le faisant précéder du caractère dièse (#).

En écrivant le style suivant :

```
div {color: black;}

#bleu {color: white; background-color: blue;}
```

puis le code XHTML :

```
<div id="bleu">Texte en blanc sur bleu</div>

<div>Texte en noir </div>
```

Les sélecteurs d'attributs :

Il est également possible d'appliquer un style à un élément déterminé dès qu'il possède un attribut donné, quelle que soit la valeur de cet attribut. Pour appliquer ce sélecteur, le nom de l'élément doit être suivi du nom de l'attribut placé entre crochets ([) et (]).

```
acronym[title] {color: red; background-color: gray;}
```

tous les éléments <acronym> qui possèdent un attribut title, quelle que soit sa valeur, ont un contenu affiché en rouge sur fond gris.

De même, en définissant le style pour l'élément <img />

```
img[longdesc] {border–color: red; border–weight: 2px;}
```

toutes les images ayant un attribut longdesc ont une bordure rouge de deux pixels de large.

```
*[title] {background–color: yellow;}
```

ce sont tous les éléments ayant l'attribut title qui sont affichés avec un fond jaune.

```
h2[title][id]{background-color: yellow;}
```

seuls les titres <h2> ayant à la fois les attributs title et id ont une couleur de fond jaune.

Par exemple, avec la définition suivante, appliquée à une cellule de tableau :

```
td {font–size: 12px;}
```

```
td [title="nom"] [align="center"] {font–size: 14px; color: red;}
```

seules les cellules ayant un attribut title avec la valeur nom et un attribut align ayant la valeur center sont affichés avec un texte rouge et avec une taille de fonte de 14 pixels, alors que le texte des autres cellules a une taille de 12 pixels.

En écrivant par exemple le style suivant :

```
td[id ~="nom"]{background-color: #222;color: white;}
```

toutes les cellules du tableau dont l'attribut id contient la valeur nom seront affichées avec un fond gris foncé et en caractères blancs.

Note :

Internet Explorer ignore tout de ces sélecteurs.

Les sélecteurs contextuels parent-descendant :

Plutôt que de définir un style pour toutes les occurrences d'un élément, nous pouvons souhaiter ne l'appliquer qu'en fonction de sa position relative par rapport à un autre dans la hiérarchie des éléments de la page. Ce type de sélecteur est dit contextuel. Nous pouvons par exemple définir un style général pour l'élément et vouloir lui en appliquer un autre quand il se trouve inclus dans un élément . Pour cela, il faut utiliser la syntaxe suivante :

element_parent element_enfant {Définition des styles;}

En écrivant par exemple le style suivant :

p {color: blue;}

div p{color: red;}

et en l'appliquant au code XHTML ci-après :

Texte de la division

Texte du paragraphe inclus dans div

Texte d'un paragraphe non inclus dans div

Seuls les contenus des éléments inclus dans sont de couleur rouge, tous les autres étant bleus, et le texte inclus directement dans à la couleur par défaut qui est le noir.

Remarque :

Il est aussi possible de préciser la hiérarchie en appliquant plus de deux éléments en les séparant tous par un espace.

Note :

Internet Explorer ignore tout de ces sélecteurs.

Pseudo-classes et pseudo-éléments :

Les sélecteurs précédents permettent d'attribuer un style à un ou plusieurs éléments bien définis dans la hiérarchie d'un document XHTML/HTML. Les pseudo-classes et les pseudo-éléments permettent d'attribuer un style à une partie abstraite d'un document non identifiable dans cette hiérarchie, par exemple le premier caractère ou la première ligne d'un paragraphe. D'autres pseudo-classes permettent d'attribuer un style à un document en fonction des actions prévisibles

mais non déterminées de l'utilisateur final, par exemple le fait de placer son curseur sur un lien ou un composant de formulaire.

Les pseudo-classes applicables aux liens :

Deux pseudo-classes spécifiques aux éléments possèdent un attribut href faisant référence à un document externe (lien vers une autre page) ou interne (ancre vers une partie du même document). Il s'agit des pseudo-classes suivantes :

• :link, qui permet d'attribuer un style à un lien qui pointe vers un document non encore vu. C'est l'état normal de tous les liens à l'ouverture de la page.

• :visited, pour attribuer un style à un lien qui pointe vers un document déjà vu, après un retour sur la page d'origine.

Exemple :

a:link {color: blue;} lien non encore visité

a:visited {color: red;} lien visité

Les pseudo-éléments :

Leur nom vient de ce qu'ils permettent d'agir sur une partie du contenu d'un élément comme s'il était contenu dans un nouvel élément fictif. On dénombre les quatre pseudo-éléments suivants :

• :first-letter, qui permet d'affecter un style à la première lettre du contenu d'un élément indiqué avant ce sélecteur. On l'utilise classiquement pour créer des effets de lettrines en définissant pour ce sélecteur une taille de caractères très supérieure à la taille de l'élément. En écrivant par exemple le style suivant :

p:first-letter {fonize: 300%; color: blue;}

la première lettre de chaque paragraphe sera trois fois plus grande que les autres et decouleur bleue. Le pseudo-élément :first-letter n'admet que les propriétés suivantes :

font, font-size, font-family, font-style, font-weight, color, background, margin,

padding, border, text-decoration, vertical-align, text-transform, line-height,

float, letter-spacing, word-spacing, clear.

• :first-line, qui permet d'affecter un style à la première ligne du contenu de l'élément indiqué. Cet affichage permet d'attirer l'attention sur un texte. En écrivant le style suivant :

```
div:first-line{font-size: 150%; font-weight: bold;}
```

la première ligne de chaque division sera affichée en gras et dans une taille 1,5 fois plus grande que la police en cours. Le pseudo-élément :first-line n'admet que les propriétés suivantes :

font, font-size, font-family, font-style, font-weight, color, background,

word-spacing, letter-spacing, text-decoration, vertical-align, text-transform,

line-height.

•:before, qui permet d'insérer un contenu doté d'un style particulier avant le contenu réelde l'élément précisé, en l'associant avec la propriété content. En écrivant le style suivant :

```
cite:beforeontent:"<<"; font-weight: bold;}
```

chaque contenu d'une citation <cite> sera précédé des caractères << en gras.

•:after, qui joue un rôle similaire au précédent mais définit un contenu doté d'un style à la fin du contenu de l'élément utilisé. En écrivant :

```
cite:after {cnt: ">>";font-weight: bold;}
```

chaque citation contenu dans l'élément <cite> sera suivie des caractères >> en gras.

La déclaration !important :

Chaque déclaration de style peut revêtir un caractère de plus grande importance par rapport à une autre déclaration concernant le même élément et la même propriété qui comporte une valeur différente. Ces deux déclarations peuvent entrer en conflit au moment de la création de la présentation par le navigateur. Pour donner cette importance à un style, il faut insérer la déclaration d'importance à l'aide du mot-clé !important en le plaçant entre la valeur attribuée à la propriété et le point-virgule qui termine la déclaration. Dans l'exemple suivant :

```
*{color: black !important; background-color: yellow;}
div{color: blue; background-color: white;}
```

Les couleurs de texte et de fond des sélecteurs * et div sont en conflit, mais comme la propriété color définie dans le sélecteur universel * est marquée !important, le texte de la division figure en noir. En

revanche, le fond de la division est de couleur blanche car la valeur yellow n'est pas marquée !important et que la déclaration faite dans div est spécifique.

Écrire des feuilles de style :

Nous allons envisager maintenant les différentes méthodes d'écriture des style CSS et la façon dont on peut les lier à un document XHTML.

Dans l'élément <style> :

Défini dans la première partie de ce livre, l'élément <style> a pour vocation de renfermer les définitions des styles CSS utilisables dans la page qui le contient. Rappelons qu'il doit toujours être inclus dans l'élément <head> et qu'il ne peut contenir que des définitions de styles CSS et des commentaires XHTML délimités par <!-- et --> ou des commentaires CSS délimités par /* et */.

Dans l'éventualité où toutes les pages d'un site ont en commun un certain nombre de styles et que chaque page possède quelques styles propres, les styles communs peuvent être écrits dans un fichier externe (voir la section suivante) et inclus dans l'élément <style> au moyen de la directive @import selon la syntaxe suivante :

```
@import url(fichier.css);
```

L'URL du fichier peut être relative ou absolue et elle peut être suivie de la désignation du média auquel les styles importés doivent s'appliquer spécifiquement.

```
<style type= "text/css">
 @import url(commun.css)all;
 @import url(ecran.css)screen;
 @import url(imprimante.css)print;
 div,p {font-style: italic;}
 h1,h2 {color: red;}
</style>
```

Un fichier externe peut inclure les styles d'un autre fichier externe en faisant appel à la directive @import de la même façon que nous avons définie plus haut pour les éléments <style>.

Dans l'attribut style :

Nous pouvons écrire par exemple :

 Le langage XHTML

représente la dernière évolution du

 HTML

- Un exemple de cas pour la bilise body :

body{color: black; background-color:#FFF; background-image:url(fondbleu2.gif);

background-repeat: repeat-y;}

- Un exemple de cas pour la bilise h2:

h2{background-color:yellow; background-image:url(fondjaune.gif);

background-repeat:repeat-y;}

Sélecteurs d'états :

• :link, qui s'applique au lien non encore visité avant toute action du visiteur. Elle permet de définir les styles du lien, généralement textuel.

• :visited, qui s'applique au lien qui a déjà été visité au moins une fois. Elle permet par exemple de changer la couleur du lien.

• :active, qui s'applique au lien au moment précis où l'utilisateur maintient le bouton de la souris enfoncé, quand le curseur est positionné sur le texte du lien. Cette pseudo-classe est mal prise en compte par les navigateurs et ne présente pas réellement un intérêt fondamental aujourd'hui.

Nous pouvons également ajouter les pseudo-classes :focus et :hover qui ne sont pas spécifiques aux liens mais que nous pouvons utiliser :

• :focus, qui permet de modifier le style d'un lien quand il reçoit le focus à l'aide de la touche Tab ou d'un raccourci clavier.

• :hover, qui permet de donner un style particulier quand le curseur est positionné sur le lien. L'effet réalisé, comme un changement de couleur par exemple, est réversible instantanément quand le curseur quitte la zone du lien.

Exemple :

```
a:link{text-decoration: underline;color: navy;}

a:focus{font-size:200%;border: yellow 2px double;}

ul li a:hover{font-size: 25px; background-color: red; color: #FFF; border: yellow 2px double;}

li a:visited{background-color: #DDD; color: #700;}

li a:active{background-color: #red; color: #567;}
```

Image en fond de page avec la propriété background :

background-color

La couleur peut être:

- Un nom de couleur, comme white, blue, etc.
- Un code comme #002244.
- Le mot transparent qui n'affiche aucune couleur de fond. C'est la valeur par défaut

Exemple :

```
div{background-color: transparent }
```

background-image

On spécifie l'URL d'une image sans guillemets. Ce peut être

- une URL locale comme url(img.jpg) ou url(images/img.jpg),
- une URL relative ou
- une URL absolue comme url(http://www.xul.fr/img.jpg).

Exemple :

```
div{background-image:url(fond1.gif);}
```

Spécifier à la fois une couleur et une image est utile quand la position de l'image est décalée.

background-position

Définit le décalage de l'image dans le conteneur. Ne s'applique pas à la couleur de fond.
Elle peut être indiquée par des valeurs absolues, par des pourcentages, ou par des mots-clés.

- 10px 20px donne une marge de 10 pixel à gauche, et en haut.
- 5% 10% crée une marge à gauche de 5% de la largeur et en haut de 10% de la hauteur.
- *left* aligne l'image à gauche (par défaut).
- *top* aligne l'image en haut (par défaut).
- *right* équivaut à 100% (moins la largeur de l'image), l'image est alignée à droite.
- *bottom* aligne l'image en bas du conteneur.

Noter que si l'on donne une marge et que l'image est répétée, la marge sera remplie par l'image du fait qu'une texture remplit le conteneur.

background-attachment

Précise si l'image est déroulée avec l'ensemble de la page (du conteneur) ou si elle est fixe, auquel cas le contenu se déplace sur l'image.
Par défaut, le fond accompagne le contenu.

- *fixed*: Le fond est fixe, le contenu est déplacé devant.
- *scroll*: Le fond est déroulé avec le contenu.
- *inherit*: Comme le conteneur parent

background-repeat

Par défaut, l'image de fond est répétée horizontalement et verticalement, et occupe tout l'espace disponible de l'élément. Pour modifier cette caractéristique, nous pouvons limiter le type de répétition à l'aide de la propriété background-repeat dont la syntaxe est la suivante :

```
background-repeat:repeat | repeat-x | repeat-y | no-repeat | inherit
```

Exemple :

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

   "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">

 <head>
```

```
<title> Les images de fond </title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<style type="text/css" >
body{background-image: url(fondbleu.gif);background-color:blue;}
p{background-color:yellow;}
h1{background-image:url(fond2.gif);}
div{background-image:url(fond1.gif);}
i{background-color:orange;background-image:url(fondjaune.gif);}
.herit{color:red;background-image:inherit;}
</style>
</head>
<body>
<h1>XHTML 1.1 &amp;amp; CSS 2.1</h1>
<div>
<h2>XHTML 1.1</h2>
<p>Pour <acronym>XHTML</acronym>, le respect des recommandations du <i>W3C
? </i>s'impose à tous les webmestres comme une nécessité...</p><br />
</div><hr />
<h2 class="herit">CSS 2.1 :<br /> feuilles de style en cascade</h2>

</body>
</html>
```

Créer des bordures, marges, espacements et contours :

Les styles de bordures :

• dotted : bordure en pointillés courts

• dashed : bordure en tirets longs

• solid : bordure pleine continue

• double : bordure constituée de deux traits parallèles continus. Si la largeur de bordure est insuffisante, un seul trait apparaît (pour définir la largeur, voir la propriété border-width) ;

• groove : bordure en creux. L'effet de creux est créé par l'emploi de couleurs différentes pour les côtés

Exemple :

```
h1.dotted{border-style: dotted;}

h1.dashed{border-style: dashed;}

h1.solid{border-style: solid; border-width: thick;}

h1.double{border-style: double; border-width: thick;}

h1.groove{border-style: groove; border-width: thick;}

h1.ridge{border-style: ridge; border-width: thick;}

h1.inset{border-style: inset; border-width: thick;}

h1.outset{border-style: outset; border-width: thick;}
```

avec le code html :

```
<h1 class="dotted">XHTML et CSS</h1>

<h1 class="dashed">XHTML et CSS</h1>

<h1 class="solid">XHTML et CSS</h1>

<h1 class="double">XHTML et CSS</h1>

<h1 class="groove">XHTML et CSS</h1>
```

```
<h1 class="ridge">XHTML et CSS</h1>

<h1 class="inset">XHTML et CSS</h1>

<h1 class="outset">XHTML et CSS</h1>
```

Nous pouvons définir individuellement chacune des bordures d'un élément en utilisant les propriétés suivantes :

• border-top-style : définit le style de la bordure haute ;

• border-right-style : définit le style de la bordure droite ;

• border-bottom-style : définit le style de la bordure basse ;

• border-left-style : définit le style de la bordure gauche.

Exemple :

```
h1{border-top-style: dotted; border-right-style: dashed;
 border-bottom-style: double; border-left-style: solid;}
p{border-left-style: inset; border-right-style: dashed;}
p.hover{border-top-style: dotted ; border-bottom-style:
double;}
```

On a aussi :

• border-top-width : pour la bordure haute ;

• border-right-width : pour la bordure droite ;

• border-bottom-width : pour la bordure basse ;

• border-left-width : pour la bordure gauche.

De même que :

border-top-color : pour la couleur de la bordure haute ;

border-right-color : pour la couleur de la bordure droite ;

border-bottom-color : pour la couleur de la bordure basse ;

border-left-color : pour la couleur de la bordure gauche.

Exemple :

border-color: red blue yellow haut : rouge, droite et gauche : bleu, bas : jaune

La déclaration globale :

h1{border: 5px double blue;}

est équivaut aux trois définitions suivantes :

h1{border-width: 5px border-style: double; border-color: blue;}

Les marges :

Afin d'aérer le contenu d'une page et en particulier l'espace entre le rendu d'un élément et ses voisins dans la page, nous pouvons définir une marge autour de chaque élément.

margin: <large>{1,4} | inherit

Le paramètre large est un nombre entier suivi d'une unité (px, ex, em, %, mm, cm, in, pc, pt). Les marges peuvent être négatives, et dans ce cas la boîte d'un élément sort de celle de son parent. Si la largeur de la marge est donnée en pourcentage, elle est calculée par rapport à celle du bloc parent. La notation {1,4} permet ici encore de définir de une à quatre marges dans le sens des aiguilles d'une montre (haut, droit, bas, gauche), et ce avec les mêmes conditions d'affectation si nous ne définissons qu'une, deux ou trois valeurs. La valeur inherit applique la marge de l'élément parent.

• margin-top : définit la marge haute ;

• margin-right : définit la marge droite ;

• margin-bottom : définit la marge basse ;

• margin-left : définit la marge haute.

Exemple :

h1{margin: 40px 10%; background-color: yellow;}

div{margin-left: 5% ; background-color: #CCC;}

p{margin: 10px -1em 15px; background-color: #EEE;}

Les espacements :

La syntaxe de la propriété padding est similaire à celle de margin :

padding: <large>{1,4} | inherit

- padding-top : définit l'espacement haut ;
- padding-right : définit l'espacement droit ;
- padding-bottom : définit l'espacement bas ;
- padding-left : définit l'espacement gauche.

Exemple :

p.retrait1{padding: 20px; background-color: #EEE;}

p.retrait2{padding:2em; background-color: #EEE;}

p.retrait3{padding-left: 10%; background-color: #EEE;}

p.herit{padding: inherit; background-color: #EEE;}

/* classe applicable aux balise p uniquement */

h1{ padding-top: 0.5em; background-color :yellow; border-style: dotted;}

Le style du contour :

Le style du contour est la première des propriétés à définir pour obtenir un affichage, les autres ayants des valeurs par défaut. Il est créé grâce à la propriété outline-style, dont la syntaxe est similaire à celle de border-style :

outline-style: none | <style> | inherit

les valeurs sont similaires à celles de style ; on peut de même définir un contour de façon individuelle.

Ouline-color : couleur.

outline-width: <long> | thin | medium | thick | inherit

exemple :

input:focus{outline-style: dotted; outline-color: orange;}

Pour la police, on a :

p#arial{font-family:Arial sans-serif;}

Pour la taille, on a :

div{background-color: transparent; font-size :125px} ou en poucentage

Grâce à la propriété text-transform, il est possible d'agir sur la casse du texte d'un élément.
Sa syntaxe est la suivante :

text-transform : uppercase | lowercase | capitalize | none | inherit

uppercase : le texte est mis en majuscules ;

lowercase : le texte est mis en minuscules ;

capitalize : seule la lettre initiale de chaque mot est mise en majuscules.

Pour le dimensionnement du texte, on peut utiliser :

p{width:80%;height:500px; background-color: blue}

Quand un élément est déclaré flottant, la boîte qui correspond à l'élément est déplacée vers la gauche ou la droite de son conteneur selon la valeur choisie pour le flottement.
L'alignement vertical est tel que la boîte est déplacée sur le haut de la ligne qui contient l'élément, ou sur le bas du bloc qui le précède. On procède à la définition du flottement d'un élément au moyen de la propriété float qui peut s'appliquer à tous les éléments XHTML et dont la syntaxe est la suivante :

float: left|right|none|inherit

Les valeurs qu'elle peut prendre ont la signification suivante :

• left : la boîte de l'élément est déplacée en haut et à gauche de son conteneur ;

• right : la boîte de l'élément est déplacée en haut et à droite ;

• none : la boîte ne flotte pas (c'est la valeur par défaut) ;

• inherit : la boîte hérite du comportement de son élément parent (ce qui n'est pas le cas par défaut).

CONCLUDION :

Le CSS est un outil très puissant dans la gestion d'application web, en particulier dans la mise en page. Ainsi, couplé avec javascript et php, le webmaster se prône maître du web.

Abdourahmane FALL

fallprofessionnel@hotmail.fr

+221 77 274 46 71