Chapitre 5: Les feuilles de style CSS

Introduction

Le langage **CSS** (*Cascading Style Sheets*) est utilisé pour définir l'aspect futur des sites web, comme par exemple la couleur du fond de la page ou le type de police.

Plus concrètement, le **CSS** (ou *feuille de style*) est un document numérique qui va pouvoir spécifier toutes les caractéristiques de mise en forme du document lié à la balise à laquelle elle s'applique.

Exemple :

Dans un document d'une certaine importance, il arrive fréquemment que l'on attribue à certains éléments des caractéristiques de mise en forme identiques. Par exemple, les noms des chapitres seront mis en police Arial, en gras et en couleur bleu.

On peut imaginer qu'on peut donner à cette définition de mise en forme un nom soit "**titre**" et qu'à chaque nouveau chapitre, plutôt que d'écrire chaque fois le nom du titre et puis le mettre en Arial, gras, bleu, on peut affecter au nom du chapitre cette mise en forme, déjà défini, sous le nom de "**titre**".

On parle :

- de **feuilles de style** [*style sheets*] car le but est d'en définir plusieurs.
- de feuilles de style **en cascade** [*Cascading Style Sheets* ou *CSS*] car en cas de styles identiques, un ordre de priorité sera déterminé par le browser

I. Utilité et avantages

Les apports des feuilles de style peuvent être résumés comme suit :

- **Séparation** du contenu et de la mise en forme.
- **Cohésion** de la présentation tout au long du site avec les feuilles de style externes.
- **Modification** de l'aspect d'une page ou d'un site sans en modifier le contenu et cela en quelques lignes plutôt que de devoir changer un grand nombre de balises.
- **Ecriture** concise et nette par rapport au Html qui devient vite fouillis.
- **Réduction** du temps de chargement des pages.

- **Dépassement** des certaines insuffisances du langage Html (contrôle des polices, contrôle de la distance entre les lignes, contrôle des marges (sans devoir utiliser de tableaux...).
- **Permettre** le positionnement au pixel près du texte et/ou des images.

II. Définition

La syntaxe d'un style est toujours la même, elle précise la balise à laquelle le style va s'appliquer, et les différents attributs du style. Les attributs sont enfermés entre **deux accolades {}** ou deux guillemets (précédés de Style) selon les cas, et chacun des attributs est séparé par un point-virgule. On donne la valeur de l'attribut par la syntaxe suivante "**nom de l'attribut: valeur de l'attribut**"

```
balise {propriété de style : valeur ;  propriété de style : valeur}
```

Exemple :

```
h3 {font-family: Arial; font-style: italic}
```

Donc ici, la balise **h3** sera en **Arial** et en ***italique***. Et dans le document en question, toutes les balises <h3> auront comme style Arial et italique.

NB :

- Pour la lisibilité, on doit écrire les styles sur plusieurs lignes :

```
h3 {
        font-family: Arial;
        font-style: italic;
        font-color: green
}
```

- On peut attribuer **plusieurs valeurs à une même propriété**. Dans ce cas, on séparera les différentes valeurs par des virgules.

```
h3 {font-family: Arial, Helvetica, sans-serif}
```

- On peut attribuer un même style à plusieurs balises (séparées par des virgules).

```
h1, h2, h3 {font-family: Arial; font-style: italic}
```

Il existe trois manières d'utiliser et d'intégrer les feuilles de style dans un document **HTML**, **intra-lignes**, **globales**, ou **importées**. Chacune de ces manières a ses avantages et ses particularités.

1. CSS Intra-lignes : A côté des balises

Une feuille intra-ligne s'insère directement à côté de la balise qu'elle définit, elle ne constitue donc pas véritablement une feuille, simplement elle permet localement de fixer des attributs à une partie d'un document. Sa déclaration est un peu particulière, en voici une (notez la forme **Style=" "**):

```
<html>
<body>
        <h1 style="font-family: Arial; font-style: italic"> blabla </h1>
</body>
</html>
```

Signalons que:

- Le style Arial, italique n'affectera que cette seule balise h1.
- La syntaxe est légèrement différente de la précédente.
- L'écriture : **<style type="text/css">h1 {"font-family: Arial; font-style: italic" }</style>** fonctionne aussi.

2. CSS Globale

A l'intérieur des balises **<head></head>** :

Une feuille globale va se déclarer dans la tête du document, entre les balises « **head** ». Elle va s'appliquer globalement à toutes les balises du document qu'elle a défini.

```
<html>
<head>
        <style type="text/css">
                <!--
                        La ou les feuille(s) de style
                -->
        </style>
</head>
<body>
```

- La balise « **<style>** » avertit le navigateur que l'on va utiliser des feuilles de style.
- L'attribut **type="text/css"** informe que ce qui suit est du texte et qu'il s'agit de cascading style sheets (*css*).

- La balise Html de commentaires **<!– ... –>,** empêche les navigateurs qui ne connaissent pas les feuilles de style, d'interpréter ces instructions. Les informations à l'intérieur des tags de commentaires seront ignorées par ces navigateurs.
- Pour les commentaires à propos des feuilles de style, on utilisera la convention désormais classique **/* commentaires */** de (C, C++, Javascript, etc.).

3. CSS importée : Dans un document séparé

Une feuille liée va se déclarer à part, dans un fichier avec une extension **.css** qui sera mis à côté des autres documents dans le répertoire du site web. Cette feuille de style sera valable pour toutes les pages du site qui l'appelleront dans les balises **HEAD**. C'est un outil très puissant pour uniformiser la mise en pages d'un grand nombre de documents.

Principe:

On crée d'abord, dans le répertoire du site, un fichier avec l'extension **.css** soit **styles.css** qui contiendra toutes les feuilles de style.

```
td {font-family : Arial, Helvetica; font-size:10pt}
```

Ensuite, on crée une page soit **page.htm** avec dans la **<head>** un lien vers ce fichier **CSS** :

```
<html>
<head>
        <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Commentaires :

- La balise **<link>** avertit le navigateur qu'il faudra réaliser un lien.
- L'attribut **rel="stylesheet"** précise qu'il y trouvera une feuille de style externe.
- L'attribut **type="text/css"** précise que l'information est du texte et du genre cascading style sheets (*css*).
- L'attribut classique de lien **href=" ... "** donne le chemin d'accès et le nom du fichier à lier.

III. Les classes

1. Définition

Un sélecteur contextuel, ou classe, associe une règle particulière à un élément en fonction de sa situation. On appelle sélecteur simple les balises HTML classiques auxquelles on a attribué des

caractéristiques de style. Mais ces sélecteurs simples présentent une contrainte importante, définie de manière générale, il est difficile de changer momentanément certains de leurs attributs ou bien d'en rajouter. Pour résoudre ce problème, les concepteurs du CSS ont inventé les classes. Une classe est ce qu'on appelle un sélecteur contextuel. Il va pouvoir venir modifier contextuellement une caractéristique du document.

La définition d'un style était :

```
balise {propriété de style : valeur}
```

Elle devient :

```
balise.nom_de_classe {propriété de style : valeur}
```

Remarquez le **point** entre balise et nom_de_classe (**indispensable**). Ou, comme la mention de la balise est facultative,

```
.nom_de_classe {propriété de style : valeur}
```

Pour appeler l'effet de style dans le document, on ajoute le nom de la classe à la balise.

```
<balise class="nom_de-classe"> .... </balise>
```

Exemple :

On souhaite mettre ce qui est important dans le texte en gras et en bleu.

On crée la classe .Toto :

```
.Toto {font-weight: bold; font-color: #000080; }
```

Et dans le document Html, il suffit d'appeler la classe quand cela se révèle nécessaire :

```
<p class="Toto"> ... blabla ... </p>
<h1 class="Toto ">Titre 1</h1>
<table><tr><td class="Toto ">cellule</td></tr>...
```

2. Appliquer plusieurs classes au même élément

L'avantage de définir des classes abstraites est qu'elles peuvent s'appliquer à n'importe quel élément. Leur puissance peut être multipliée car nous pouvons appliquer plusieurs classes indépendantes à un même élément. Celui-ci a alors la combinaison des propriétés de chacune des classes. Pour utiliser plusieurs classes dans le même élément HTML, il faut donner à son attribut **class** la liste des noms des classes en les séparant par un espace comme ceci :

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

IV. Les identifiants (ID)

L'attribut « **id** » fonctionne exactement de la même manière que « **class** », à un détail près : il ne peut être utilisé qu'une fois dans le code. La syntaxe est :

```
#nom_de_ID {propriété de style : valeur}
```

Et pour l'appeler :

```
<balise id="nom_de_ID"> .... </balise>
```

Notons : on ne pourra effectuer qu'un seul appel à ***#nom_de_ID*** par page. Ainsi, Pour :

```
#Toto { ... }
<p id="Toto"> est correct, mais si on le rencontre dans la même page :
<h1 id="Toto"> ce n'est plus correct !
```

V. Les sélecteurs avancés

En CSS, le plus difficile est de savoir cibler le texte dont on veut changer la forme. Pour cibler (on dit « sélectionner ») les éléments de la page à modifier, on utilise ce qu'on appelle des **sélecteurs**.

- Les éléments HTML (p, h1, img, etc)
- Les classes
- Les identifiants

Maintenant on va voire quelques sélecteurs avancés :

1. ***** : Sélecteur universel. Sélectionne toutes les balises sans exception.

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

2. **A, B** : Sélectionner A et B

```
h1, h2, h3 {color: aquamarine;}
```

3. **A B** : Sélectionner une balise contenue dans une autre

```
h3 em {color: aquamarine;}
<h3>Titre avec un <em>texte important</em></h3>
```

4. **A + B** : Sélectionner une balise qui en suit une autre

```
h3 + p {color: aquamarine;} /* Séléctionne la 1ère balise p après un titre h3 */
<h3>Titre</h3>
<p>Paragraphe</p>
```

5. **A[attribut]** : Sélectionner une balise qui possède un attribut

```
input[name] {background-color: aquamarine;}
```

```
<input type="text" name="nom" />
<input type="text" name="prenom" />
```

6. **A[attribut="valeur"]** : Sélectionner une balise, un attribut et une valeur exacte

```
input[name="nom"] {background-color: aquamarine;}
<input type="text" name="nom" />
<input type="text" name="prenom" />
```

7. **A[attribut*="valeur"]** : Sélectionner une balise, un attribut et une valeur

```
input[name*="nom"] {background-color: aquamarine;}
<input type="text" name="nom" />
<input type="text" name="prenom" />
```

8. **A:last-child** : Sélectionner le dernier enfant, indépendamment du type de son parent

```
tr:last-child {background-color: aquamarine;} /* Sélectionne la dernière ligne d'un tableau */
```

9. **A:first-child** : Sélectionner le premier enfant, indépendamment du type de son parent

```
tr:first-child {background-color: aquamarine;} /* Sélectionne la première ligne d'un tableau */
```

10. **A:nth-child()** : Sélectionner le nième enfant, indépendamment du type de son parent

```
tr:nth-child(2) {background-color: aquamarine;} /* Sélectionne la 2ème ligne d'un tableau */
```

11. **A>B** : Sélectionner tous les éléments B qui suivent directement l'élément A.

```
#contenu>.box {background-color: aquamarine; } /* Sélectionne box1 */
<div id="contenu">
     <div class="box">box1</div>
    <div>
            <div class="box">box2</div>
    </div>
 </div>
```

VI. Les pseudo-classes

Les pseudo-classes permettent d'affiner le style appliqué à un certain nombre de balises en définissant une réaction à un événement ou bien à la position relative de la balise au sein des autres balises.

1. Les pseudo-classes d'ancre : :link et :visited

En général, les navigateurs représentent différemment les liens qui n'ont pas été visités de ceux qui l'ont déjà été. CSS en fournit un équivalent au travers des pseudo-classes ':link' et ':visited'.

- La pseudo-classe **:link** s'applique aux liens qui n'ont pas été visités
- La pseudo-classe **:visited** s'applique lorsque le lien a été visité par l'utilisateur

Les deux déclarations CSS2 suivantes produisent le même effet :

```
a:link { color: red }
:link { color: red }
```

Exemple(s):

```
<a class="external" href="http://www.mon-lien">lien externe</a>
a.external:visited { color: blue }
```

Celle-ci va rendre bleue la couleur du lien visité

2. Les pseudo-classes dynamiques : :hover, :active et :focus

Les navigateurs interactifs changent parfois l'aspect du rendu en réponse aux actions de l'utilisateur. **CSS2** propose trois pseudo-classes pour un usage courant :

- La pseudo-classe **« :hover »,** permet d'affecter un style à la balise sélectionnée lors d'un survol par le curseur de la souris.

```
a:hover {text-decoration: none;}
```

- La pseudo-classe **« :active »,** permet de définir un style à la balise sélectionnée lorsque l'utilisateur clique sur l'élément (entre le moment où l'utilisateur clique sur le bouton de la souris et celui où il le relâche).

```
a:active {color: #FF0000;}
```

- La pseudo-classe **« :focus »,** permet de définir un style à la balise sélectionnée lorsque le focus lui est donné (par exemple lors d'un clic dans un élément de formulaire).

```
textarea:focus {color: #FF0000;}
```

VII. Les pseudo-éléments

Un pseudo-élément est un mot-clé ajouté à un sélecteur qui permet de mettre en forme certaines parties de l'élément ciblé par la règle

1. Le pseudo-élément :first-line

Le pseudo-élément **« :first-line »** produit un style particulier sur la première ligne formatée d'un paragraphe. Par exemple :

```
p:first-line { text-transform: uppercase }
```

La règle précédente signifie "**mettre les lettres de la première ligne de chaque paragraphe en majuscule**". Notons que la longueur de la 1ère ligne dépend de plusieurs facteurs, dont la largeur de la page, la taille de la police, etc. Ainsi un paragraphe quelconque en HTML tel que :

```
<P>Voici un paragraphe HTML plutôt long qui va être découpé en plusieurs lignes. La première ligne sera identifiée
à l'aide d'une séquence de balises fictives. Les autres lignes vont rester ordinaires dans la suite du paragraphe.</P>
```

Celui-ci va apparaître avec le découpage des lignes suivant :

```
VOICI UN PARAGRAPHE HTML PLUTÔT LONG
qui va être découpé en plusieurs lignes. La première ligne sera identifiée à
l'aide d'une séquence de balises fictives. Les autres lignes vont rester ordinaires
dans la suite du paragraphe.
```

2. Le pseudo-élément :first-letter

Le pseudo-élément « **:first-letter** » peut être employé pour faire des capitales initiales et des **lettrines**, ce sont des effets typographiques courants.

Cette feuille de style produit une lettrine qui s'étend sur deux lignes :

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Lettrine</title>
<style type="text/css">
        p { font-size: 12pt; line-height: 12pt }
        p:first-letter {
                font-size: 300%; font-style: italic;
                font-weight: bold;
                float: left
        }
</style>
</head>
<body>
<p>Les premiers mots parmi ceux d'un article de journal.</p>
</body>
</html>
```

Ce qui pourrait apparaître de cette manière :

3. Les pseudo-éléments ::after et ::before

Les pseudo-éléments **::after** et **::before** permettent d'ajouter du contenu directement après ou directement avant un élément.

Les pseudo-éléments doivent être utilisés avec la propriété **content**. Cette propriété permet de spécifier le contenu qui est inséré. Par défaut, ce contenu est de type « inline».

```
h1::before{
        content: "Titre: " ;
}
h3::after{
        content: " →" ;
}
```

4. Le pseudo-élément ::selection

Le pseudo-élément **::selection** permet d'appliquer des règles CSS à une portion du document qui a été sélectionnée par l'utilisateur (via la souris ou un autre dispositif de pointage).

```
::selection {
        background: cyan;
}
```

5. Le pseudo-élément ::placeholder

Le pseudo-élément **::placeholder** représente le texte de substitution pour un élément de formulaire. Cela permet aux développeurs web de personnaliser l'apparence de ce texte..

```
input::placeholder {
        color: blue;
        font-size: 1.5em;
}
```

NB : cette fonction ne fonctionne pas dans tous les navigateurs

VIII. Les unités dans les feuilles de style

Grâce aux feuilles de style il est possible de définir des valeurs numériques pour les propriétés de style de plusieurs façons :

- de façon ***absolue***, c'est-à-dire dans une unité indépendante du format de sortie (en centimètres par exemple) ;
- de façon ***relative***, c'est-à-dire dans une unité relative à un élément ;

Les valeurs des feuilles de style sont soit des nombres entiers, soit des nombres réels, c'est-à-dire des chiffres ayant une partie entière et une partie décimale.

D'une manière générale il est à noter l'utilisation du point (« . ») dans les notations décimales en lieu et place de la virgule (« 8.5 cm« et non « 8,5 cm«).

Les valeurs peuvent par ailleurs dans certains cas être négatives (précédées du signe « - »).

1. Unités absolues

Les unités absolues proposées par le standard CSS sont récapitulées dans le tableau suivant :

Unité	Description
cm	Le centimètre
in	Le pouce (en anglais « inch ») correspondant à 2,54 cm
mm	Le millimètre
pt	Le point
pc	Le pica (correspondant à 12 pt)

2. Unités relatives

Les unités relatives proposées par le standard CSS sont récapitulées dans le tableau suivant:

Unité	Description
em	Unité relative à la taille de police de l'élément sélectionné. Seule exception à cette règle : lorsque la propriété font-size est définie, elle se rapporte à la taille de la police de l'élément parent.
ex	Unité relative à la hauteur de la minuscule de l'élément sélectionné. Seule exception à cette règle : lorsque la propriété font-size est définie, elle se rapporte à la hauteur de la minuscule de l'élément parent.
px	Le pixel. Il s'agit d'une unité dont le rendu dépend de la résolution du périphérique d'affichage.
%	Le pourcentage est une unité relative à la taille de l'élément ou de son parent.