

Chapitre : Les feuilles de style

I. Définition et avantages :

1. Définition :

Le **CSS** (**Cascading Style Sheets** en anglais, ou « **feuilles de style en cascade** ») est un langage informatique qui a été inventé **pour décrire la présentation des documents HTML** ; CSS permet de choisir la couleur du texte, la police utilisée, la taille du texte, les bordures, l'arrière-plan...et de faire la mise en page du site.

Aux débuts du Web, **CSS** n'existait pas, il n'y avait initialement que le langage **HTML**. Cependant, les pages **HTML** commençaient à devenir assez complexes : Il y avait de plus en plus de balises et le mélange entre le fond et la forme devient délicat, ce qui rendait la mise à jour des pages web de plus en plus difficile et embrouillée. C'est pour cela que l'on a créé le langage **CSS** avec un objectif majeur de séparer la structure d'un document de ses styles de présentation.

C'est ainsi que le langage **CSS** a été introduit en 1996 par Microsoft avec son Internet Explorer 3.0. Ces feuilles de style n'ont connu qu'un intérêt limité du fait que celles-ci n'étaient pas reconnues par Netscape Navigator 3.0. Plus tard, la norme **Html 4.0** en a repris le concept avec **CSS** version 1, tout en sachant que les standards définissant **CSS** sont publiés par le **W3C** (World Wide Web Consortium).

Remarques :

- On remarque que l'on parle de feuilles de style car l'objectif est d'en définir plusieurs.
- On parle aussi de feuilles de style en **cascade** car en cas de styles identiques, un ordre de priorité ou poids sera déterminé, pour chaque règle, par le navigateur.

L'ordre de préférence et d'application d'un style va dépendre de trois grands facteurs qui vont être :

- ✚ La présence ou non du mot clef **important** : une règle qui utilise la syntaxe **important** sera jugée comme prioritaire peu importe la précision du sélecteur ou sa place dans le code.
- ✚ Le degré de précision du sélecteur ;
- ✚ L'ordre de déclaration des règles dans le code.

A noter que ces trois facteurs vont être analysés dans l'ordre donné et que le premier va primer sur le deuxième qui va primer sur le dernier.

2. Avantages :

- La structure du document et la présentation peuvent être gérées dans des fichiers séparés ce qui rend le travail d'équipe plus facile.
- La conception d'un document se fait dans un premier temps sans se soucier de la présentation, ce qui permet d'être plus efficace.

- Dans le cas d'un site web, la présentation est uniformisée : les documents (pages HTML) font référence aux mêmes feuilles de styles. Cette caractéristique permet de plus une remise en forme rapide de l'aspect visuel.
- Le code HTML est considérablement réduit en taille et en complexité, puisqu'il ne contient plus de balises ni d'attributs de présentation.

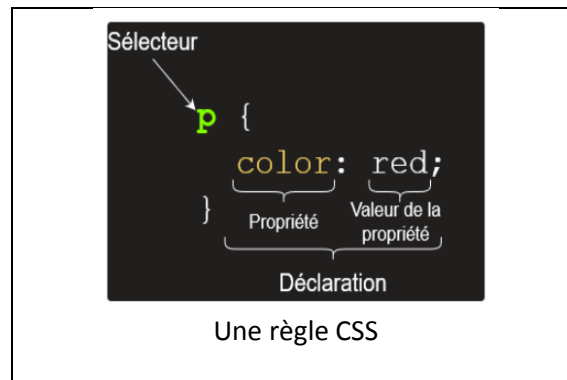
II. Syntaxe de CSS :

1. Définition d'un style :

CSS est un langage basé sur **des règles de styles** destinées à des éléments ou des groupes d'éléments particuliers dans la page.

Par exemple : « Je veux que le contenu de tous les textes des paragraphes de ma page s'affiche en rouge. ».

Dans le code suivant, une règle CSS élémentaire réalise cette mise en forme :



Cette structure s'appelle « **une règle CSS** ». Une feuille de style CSS est constituée d'une succession de telles règles où chacune est constituée de différentes parties qui se nomment comme suit :

| | |
|-------------------------------|---|
| Sélecteur | <p>La règle commence par un sélecteur, soit le nom de l'élément HTML mis en forme.</p> <p>Un sélecteur est un motif qui permet d'appliquer un style aux éléments du document HTML.</p> |
| Déclaration | <p>Par la suite, une paire d'accolades '{ }' est placée, à l'intérieur desquelles on trouve une ou plusieurs déclarations, sous la forme d'une paire propriété : valeur qui détermine les propriétés de l'élément que l'on veut mettre en forme.</p> |
| Propriétés | <p>Les propriétés sont les différentes façons ou aspects ou styles dont on peut mettre en forme un élément HTML sélectionné. Ce sont des identifiants lisibles par l'homme qui indiquent les caractéristiques stylistiques (par exemple font-size, width, background-color,..) qu'on souhaite modifier.</p> |
| Valeur de la propriété | <p>Une valeur est attribuée à chaque propriété spécifiée permettant ainsi de choisir une mise en forme parmi d'autres.</p> |

Les éléments importants de la syntaxe à respecter sont les suivants :

- Chaque ensemble de règles, à l'exception du sélecteur, doit être entre accolades ({}).
- Pour chaque déclaration, il faut utiliser deux points (:) pour séparer la propriété de ses valeurs.
- Si on veut modifier plusieurs propriétés d'un coup, on peut utiliser plusieurs déclarations dans une seule règle. Par la suite, il faut utiliser un point-virgule (;) pour séparer les déclarations entre elles.
- Une ou plusieurs valeurs possibles peuvent définir le comportement d'une même propriété définie par **CSS**. Dans ce cas, on séparera les différentes valeurs par des virgules (,). **Par exemple :** `p {color: red, pink}.`
- On peut attribuer une même règle à plusieurs types d'éléments. Ainsi, il suffit de placer plusieurs sélecteurs, séparés par des virgules.
Par exemple : `H1, H2, H3 {font-family: Arial; font-style: italic}.`

2. Commentaires et indentation en CSS :

- Il est essentiel de bien organiser et de bien commenter son code CSS afin de ne pas faire d'erreur en appliquant par exemple deux styles différents à un même élément.
- **Les commentaires mono-ligne et multi-lignes commencent par les caractères "/*" et se terminent par "*/" .**
 - Indenter en CSS est également très important afin de conserver le plus de clarté possible dans son code.
 - Pour plus de lisibilité, on retourne également à la ligne après chaque déclaration. Cela va augmenter de façon très minime le temps d'exécution du code et donc le temps d'affichage de la page.

III. Les différents types de sélecteurs :

Les sélecteurs sont l'un des éléments fondamentaux du CSS et sont utilisés pour cibler des contenus HTML et leur appliquer les styles souhaités.

Il y a différents types de sélecteurs. Voici quelques-uns des types de sélecteur les plus fréquents :

| Nom du sélecteur | Ce qu'il sélectionne | Exemple |
|---|--|---|
| Sélecteur d'élément (parfois appelé « sélecteur de balise » ou « sélecteur de type ») | Tous les éléments HTML d'un type donné. | <code>p</code> sélectionne tous les <code><p></code> . |
| Sélecteur d'ID | L'élément d'une page qui possède l'ID fourni (pour une page HTML donné, on ne peut avoir qu'un seul élément pour un ID donné). | <code>#my-id</code> sélectionne <code><p id="my-id"></code> ou <code></code> |

| | | |
|-----------------------------------|--|---|
| Sélecteur de classe | Les éléments d'une page qui sont de la classe donnée (pour une page donnée, il est possible d'avoir plusieurs éléments qui partagent une même classe). | .my-class sélectionne <p class="my-class"> et |
| Sélecteur d'attribut | Les éléments de la page qui possèdent l'attribut donné. | img[src] sélectionne mais pas |
| Sélecteur de pseudo-classe | Les éléments donnés mais uniquement dans un certain état (par exemple quand on passe la souris dessus). | a:hover sélectionne <a> mais uniquement quand le pointeur de la souris est au- dessus du lien. |

IV. Déclaration du CSS :

On peut écrire du code en langage CSS à **trois endroits différents** :

1. feuille de style externe :

Ecrire du code CSS dans un fichier séparé qui porte l'extension '**.css**' : On parle alors de **feuille de style externe**.

C'est la méthode la plus recommandée et la plus utile vu ses nombreux avantages qui sont principalement :

- une meilleure maintenabilité du code grâce à la séparation des différents langages.
- une meilleure lisibilité.
- Réutilisabilité : Appliquer des styles à plusieurs pages HTML en même temps, d'un seul coup.

En pratique :

1. nous allons commencer par créer un nouveau fichier qui contiendra toutes les feuilles de style dans notre éditeur, soit par exemple **styles.css**.
2. Nous allons enregistrer ce fichier et le placer dans le même dossier que notre page HTML, pour plus de simplicité, soit par exemple **exemple.html**.
3. Ensuite, nous devons « **lier** » notre fichier **HTML** à notre fichier **CSS** pour indiquer au navigateur qu'il doit appliquer les styles déjà définies dans le fichier **styles.css** au fichier **exemple.html**.

Pour cela, nous allons placer un nouvel élément HTML **<link>** (« lien », en français). au sein de l'élément **<head>** de notre fichier HTML.

Le code HTML ci-dessous illustre cette technique :

| | |
|--|--------------------------|
| exemple.html | styles.css |
| <html> <head> <title>style externe</title> | h1{ color: blue; } |

| | |
|--|---|
| <pre> <link rel="stylesheet" type="text/css" href="styles.css"/> </head> <body> <h1>Un titre de niveau 1 (en bleu)</h1> <p>Un paragraphe (en rouge, et en gras)</p> </body> </html> </pre> | <pre> p{ color:red; font-weight: bold; } </pre> |
| <p>L'aperçu de « exemple.html » avec Google chrome est :</p> <div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: fit-content;"> <p>Un titre de niveau 1 (en bleu)</p> <p>Un paragraphe (en rouge, et en gras)</p> </div> | |

- La balise **<LINK>** avertit le browser qu'il faudra réaliser un lien.
- L'attribut **rel** précise le type de ressource que l'on souhaite lier à notre fichier HTML. Dans notre cas, nous indiquerons la valeur **stylesheet** pour « feuille de style ».
- L'attribut **type="text/css"** précise que l'information est du texte et du genre CSS.
- L'attribut classique de lien **href** donne le chemin d'accès au fichier CSS à lier.

2. Feuille de style interne :

On peut écrire le code CSS à l'intérieur même d'un fichier HTML, dans l'entête, au sein de l'élément **<head>** , dans un élément **<style>** : On parle alors de **feuille de style interne**.

Ces styles s'appliqueront à toutes les balises du document HTML dans lequel ils sont écrits.

Le code HTML ci-dessous illustre cette technique :

```

exemple.html :
<html>
  <head>
    <title>style interne</title>
    <style>
      body{
        background-color: pink;
      }
      p{
        color: blue;
        font-size: 12px;
      }
      h1 {
        color:purple;
        font-style: italic;
      }
    </style>
  
```

```

</head>
<body>
  <h1>Un titre de niveau 1</h1>
  <p>Un paragraphe</p>
  <p>Un deuxième paragraphe</p>
</body>
</html>

```

L'aperçu de « exemple.html » avec Google chrome est :

Un titre de niveau 1

Un paragraphe

Un deuxième paragraphe

3. Feuille de style en ligne :

On peut déclarer le **CSS** à l'intérieur même d'un fichier HTML, dans le corps, au sein de l'élément **<body>**, directement via **des attributs style** qu'on va ajouter à l'intérieur des balises ouvrantes des éléments HTML pour lesquels on souhaite modifier les styles : On parle **de feuille de style en ligne**.

Ce sont des déclarations CSS qui n'affectent qu'un seul élément en particulier.

Le code HTML ci-dessous illustre cette technique :

exemple.html :

```

<html>
  <head>
    <title>style en ligne</title>
  </head>
  <body style="background-color:yellow;">
    <h1 style="color:purple; font-style: italic;">Un titre de niveau 1</h1>
    <p style="color: blue; font-size: 20px;">Un paragraphe</p>
    <p>Un deuxième paragraphe</p>
  </body>
</html>

```

Aperçu de « exemple.html » avec Google chrome est :



On peut aussi définir le style comme suit :

exemple.html :

```
<html>
<head>
  <title>style en ligne</title>
</head>
<body style="background-color:yellow;">
  <STYLE>
    H2 {font-family: Arial; font-style: italic; color: orange}
  </STYLE>

  <h1 style="color:purple; font-style: italic;">Un titre de niveau 1</h1>
  <h2>Un titre de niveau 2</h2>
  <p style="color: blue;font-size: 20px;">Un paragraphe</p>
  <h2>Un titre de niveau 2</h2>
  <p>Un deuxième paragraphe</p>
  <h2 style="color:red; font-weight:bold">Un titre de niveau 2</h2>
</body>
</html>
```

Aperçu de « exemple.html » avec Google chrome est :



Dans ce cas, ce style affectera toutes les balises <h2> à l'exception du dernier titre de niveau 2 auquel on a attribué une autre couleur de façon plus spécifique.

V. Sélecteurs de classe :

Il peut s'avérer intéressant d'affecter des styles différents à des mêmes balises. Pour cela les spécifications CSS ont introduit **le concept de classe**.

Les **sélecteurs de classe** CSS permettent de cibler des éléments d'un document HTML en fonction du contenu d'un **attribut class** qu'on peut attribuer à chaque élément.

En plus de rendre le code plus propre et plus concis (puisque'il n'est pas nécessaire d'écrire le code individuellement), cet attribut permet de maintenir une cohérence stylistique et visuelle entre les différents éléments et de simplifier le développement web.

Syntaxe :

La définition d'un style dans un fichier **' .css '** était :

balise {propriété de style : valeur ;}

Elle devient :

balise.nom_classe {propriété de style : valeur ;}

Ou, comme la mention de la balise est facultative, on peut écrire :

.nom_classe {propriété de style : valeur ;}

L'emploi du point « . » devant le nom de la classe est **obligatoire**.

Pour appeler l'effet de style dans le document **HTML**, il suffit de rajouter un **attribut class** dans la balise concernée dans le fichier **HTML** :

<balise class="nom_classe"> ... </balise>

Exemple :

On souhaite mettre le texte d'un paragraphe en rouge, en italique et au centre. Donc, on crée les classes demandées dans un fichier externe d'extension **' .css '**, et dans le document **HTML**, il suffit d'appeler les classes en cas de besoin.

Soit les classes **.rouge** et **.center** à appliquer à la balise **<p>** :

/* Cible tous les éléments <p> ayant la classe ".rouge" */

```
p.rouge {  
  color: red;  
  font-style: italic;  
}
```


/* Cible tous les éléments ayant la classe ".center" */

```
.center {text-align: center;}
```



```
/* Cible tous les éléments <p> ayant une classe qui contient à la fois les classes ".rouge" et
".center" */
p.rouge.center{
    background-color: green;
}
```

L'appel à ces classes dans le code se fera de la façon suivante :

 `<p class="rouge center">` Texte à mettre en rouge, en italique et au centre `</p>`

Ici, l'attribut **class** est une liste de termes séparés par des espaces, il est nécessaire que chacun de ces termes corresponde exactement au nom utilisé dans le sélecteur pour que l'élément soit ciblé.

VI. Les éléments HTML `<div>` et `` (conteneurs génériques) :

1. Utilité :

Les éléments `<div>` et `` sont très spécifiques puisqu'ils ne possèdent aucune valeur sémantique, c'est-à-dire qu'ils ne servent pas à préciser la nature d'un contenu. Ils n'ont aucun effet sur le contenu ou la mise en page tant qu'ils ne sont pas mis en forme d'une manière quelconque à l'aide de **CSS**. Ils vont plutôt nous servir de conteneurs génériques en HTML ayant en particulier un intérêt pour l'application de certains styles CSS.

Notons que `` et `<DIV>` s'utilisent toujours avec **les classes**.

2. L'élément HTML `<div>` :

L'élément HTML `<div>` (ou division) est un conteneur générique de bloc (**block**) utilisé comme conteneur pour différents éléments placés ensemble. Par la suite, on va pouvoir leur appliquer les mêmes styles CSS en appliquant un style spécifique directement au `<div>`.

Exemple :

| | |
|---|---|
| <p>Fichier.html</p> <pre><div> <p>Ajouter progressivement l'huile d'olive en faisant tourner le mixeur lentement. </p> </div></pre> | <p>Fichier.css</p> <pre>div{ background-color: pink; } span.ingredient { color: blue; }</pre> |
|---|---|

Aperçu de « Fichier.html » avec Google chrome est :

Ajouter progressivement l'huile d'olive en faisant tourner le mixeur lentement.

3. L'élément HTML :

L'élément HTML est un conteneur générique en ligne (*inline*) pour les contenus phrasés. Il va servir de conteneur interne à un élément plutôt que de conteneur pour plusieurs éléments : On va par exemple pouvoir placer une certaine partie du texte d'un titre ou d'un paragraphe dans un élément pour ensuite pouvoir lui appliquer un style CSS particulier.

Exemple :

Fichier.html

```
<p>
Ajouter le <span
class="ingredient">basilic</span>, <span
class="ingredient">les pignons de
pin</span> et <span
class="ingredient">l'ail</span> dans un
mixeur et mélanger jusqu'à obtenir une
pâte.
</p>

<p>Ajouter progressivement <span
class="ingredient">l'huile d'olive</span> en
faisant tourner le mixeur lentement.
</p>
```

Fichier.css

```
span.ingredient {
  color: blue;
}
```

Aperçu de « Fichier.html » avec Google chrome est :

Ajouter le basilic, les pignons de pin et l'ail dans un mixeur et mélanger jusqu'à obtenir une pâte.

Ajouter progressivement l'huile d'olive en faisant tourner le mixeur lentement.

VII. Positionner avec CSS :

Le positionnement permet de modifier le cours classique de la mise en page pour produire des effets intéressants par le fait de sortir les éléments du flux normal de la composition du document et de les faire se comporter différemment.

Exemples :

- Placer un élément sur un autre.
- Occuper toujours la même place dans la zone d'affichage du navigateur.

- Modifier légèrement le placement de certains éléments html par rapport à leur position par défaut dans la mise en page.
-

Il existe **différents types de positionnement** que vous pouvez appliquer à des éléments **HTML**. Nous allons pouvoir gérer et modifier le type de positionnement grâce à la **propriété CSS position** conjointement avec **les quatre propriétés top, left, bottom et right** qui déterminent l'emplacement final de l'élément positionné en indiquant où le coin supérieur gauche de la boîte représentant un élément doit être positionné par rapport à un certain point de référence.

La **propriété position** ne va pas nous permettre de positionner un élément en soi dans une page mais simplement de définir un type de positionnement grâce aux valeurs suivantes que nous allons tenter de comprendre leurs implications :

Position : static

C'est le comportement normal (par défaut). L'élément HTML est alors positionné de manière **static** selon le flux normal de la page.
Par la suite, les propriétés **top, left, bottom et right** ne s'appliquent pas.

position : relative

un élément positionné de manière **relative** avec **position : relative** va pouvoir être décalé par rapport à sa position de départ grâce aux propriétés **top, left, bottom et right** qui vont prendre comme origine la position initiale de l'élément.

position : absolute

Un élément positionné de manière **absolue** ne fait plus partie du cours normal de la mise en page et aucun espace n'est créé pour lui sur la page. Il se trouve plutôt sur un plan qui lui est propre, séparé de tout le reste, et il va ainsi pouvoir se placer par-dessus d'autres éléments. Ainsi, au lieu de positionner l'élément en fonction de sa position relative dans la mise en page du document, les propriétés **top, left, bottom et right** vont définir la distance à laquelle l'élément doit se situer par rapport aux côtés de l'élément contenant le plus proche.

position : fixed

Le positionnement fixe est très proche du **positionnement absolu**. En effet, un élément positionné avec **position: fixed** va également être retiré du flux normal de la page et l'espace qui lui était attribué va également pouvoir être utilisé par d'autres éléments. En dehors de certains cas particuliers, un élément positionné avec **position: fixed** sera figé et apparaîtra toujours à la même place dans la fenêtre si on descend ou on monte dans la page : il sera fixe par rapport à la fenêtre du navigateur elle-même.

position : sticky

La valeur **sticky** est une valeur récente de la propriété position et est à ce titre toujours en développement.

Bibliographie :

https://developer.mozilla.org/fr/docs/Learn/Getting_started_with_the_web/CSS_basics

<http://tvaira.free.fr/web/cours-css.pdf>

<https://www.pierre-giraud.com/html-css-apprendre-coder-cours/selecteur-propriete/>

<https://web.maths.unsw.edu.au/~lafaye/CCM/css/cssclass.htm>