1.3 — À la découverte du CSS

Le langage **CSS** (pour *Cascading Style Sheets*, ie. feuilles de style en cascade) est un langage permettant de définir les propriétés graphiques des éléments HTML constituant une page Web.

Méthode 1 : L'attribut `style`

L'attribut `style` est autorisé dans tous les éléments (par exemple `p`, `span`, `h1`, `b`, `div`, etc.) et défini des **propriétés CSS** pour l'élément concerné.

Créer un fichier HTML vierge minimal appelé `premier_css.html`.

Copier le code HTML/CSS ce dessous dans la balise `<body>` du fichier minimal.

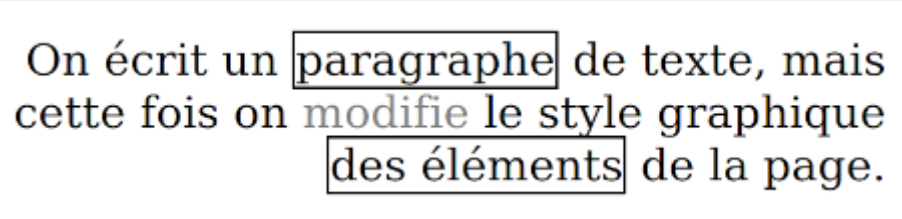
```
<p style="text-align: right;">
    On écrit un
    <span style="border: 1pt solid black;">
        paragraphe
    </span>
    de texte, mais cette fois on
    <span style="color: gray;">modifie</span>
    le style graphique
    <span style="border: 1pt solid black;">
        des éléments
    </span>
    de la page.
</p>
```

Observer le résultat obtenu.

Critiquer le code obtenu.

CORRECTION

Voici une capture d'écran du résultat attendu.



On remarque qu'avec l'attribut `style` nous venons de modifier :

- l'alignement du texte,
- l'encadrement du texte et
- la couleur du texte.

Le code obtenu est **peu lisible** et un style est dupliqué ce qui rend la **maintenance du code plus compliquée**.

Méthode 2 : les sélecteurs CSS

CSS propose une manière alternative de modifier le rendu graphique de la page par le biais de *sélecteur CSS*.

L'attribut `style` n'est présent que dans l'en-tête du document et tous les attributs `style` du corps de texte sont remplacés par des attributs `class` et `id`.

Comme `class` et `id` sont des balises neutres, elle ne modifient pas en elles même le rendu graphique. En revanche les *étiquettes* de classe ou d'identifiant qu'elles

donnent font références aux styles graphiques décrits par la balise `<style>` de l'en-tête.

Créer un fichier HTML minimal appelé `premier_css_selecteur.html` et y **copier** le code suivant :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Premier CSS avec sélecteurs</title>
    <style>
        p {text-align: right;}
        .bord {border: 1pt solid black;}
        #n42 {color: gray;}
    </style>
</head>
<body>
    <p>
        On écrit un
        <span class="bord">paragraphe</span>
        de texte, mais cette fois on
        <span id="n42">modifie</span>
        le style graphique
        <span class="bord">des éléments</span>
        de la page.
    </p>
</body>
</html>
```

Noter vos observations.

CORRECTION

Le document produit est exactement le même que celui de l'activité précédente. Il est un peu plus lisible et l'utilisation de l'attriut `class` permet d'éviter des duplications de code.

Dans l'activité :

- `p {text-align: right;}` : **toutes** les balises `<p>` du document sont alignées à droite ;
- `.bord {border: 1pt solid black;}` : **toutes** les balises ayant l'attribu `class="bord"` sont encadrées par un trait noir de 1pt d'épaisseur ;
- `#n42 {color: gray;}` : l'**unique** balise dont l'attribut `id` vaut `"n42"` est écrit en gris.

Cette méthode est plus propre que celle avec les nombreuses balises `<style>` car elle sépare la structure du document (HTML) de sa présentation graphique (CSS). Elle permet de *factoriser* les styles communs (grâce au nom des balises ou à l'attribut `class`).

ACTIVITÉ

Imaginons la situation suivante : *Votre document de présentation est composé d'une dizaine de fichiers HTML différents. Pour des raisons de cohérence visuelle, vous souhaitez que tous les fichiers aient le même rendu graphique.*

Comment s'y prendre pour homogénéiser un grand nombre de fichier

HTML différents ?

Critiquer la méthode proposée.

CORRECTION

Si on possède plusieurs fichiers HTML pour lesquels on souhaite appliqué le même style graphique, il faut dupliquer la balise `<style>` dans tous les fichiers.

Le problème est la maintenance du code. Chaque modification de style devient pénible avec un grand risque d'oubli et d'erreur. On souhaiterai ne pas avoir à dupliquer la balise `<style>`...

Méthode 3 : le fichier CSS

Il est possible de rajouter dans l'entête d'un fichier HTML une balise `<link>` qui pointe vers l'emplacement d'un fichier CSS. Comme un lien hypertexte, c'est au moyen de l'attribut `href` que l'emplacement du fichier est défini.

Le fichier CSS est écrit avec la même syntage que le contenu de la balise `<style>`.

Exemple

Voici un exemple de fichier HTML faisant référence au fichier CSS `style.css` :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
```

```
    <title>Premier CSS avec sélecteurs</title>
    <link href="style.css" rel="stylesheet" type="text/css"/>
</head>
<body>
    ...
</body>
</html>
```

Cette manière d'associer une page à un fichier de style est la plus propre. Elle respecte intégralement le principe de la séparation entre présentation des données et structuration du contenu.

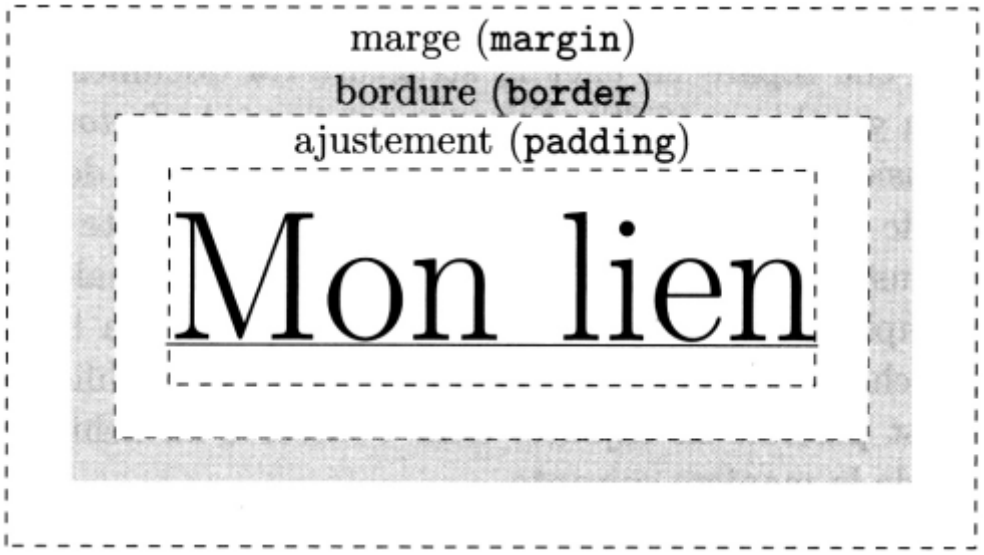
Quelques propriétés CSS

Le format CSS est standardisé. Voici un aperçu de quelques propriétés.

Modèle de boite Tout élément HTML possède une boîte rectangulaire qui délimite son contenu. Par exemple :

```
<a href="https://www.monsite.org">Mon lien</a>
```

Lorsque le navigateur dessine ce lien, il définit une boîte possédant plusieurs zones distinctes :



- `padding` définit l'espace entre la zone interne et la bordude ;
- `border` définit la taille de la bordure ;
- `margin` définit l'espace au delà de la bordure.

La boîte peut avoir un comportement de *balise de structure* ou de *balise de texte*. Pour cela il faut changer la valeur de la propriété `display` :

```
display : block ;
```

La boîte se comporte comme une balise de structure. Comme `<p>` ou `<ul>`, la boîte impose un retour à la ligne.

```
display : inline ;
```

La boîte se comporte comme une balise de texte. Comme `<a>` ou `<b>`, la boîte est affichée dans le flot du texte.

```
display : none ;`
```

La boîte est masquée !

Longueur De nombreuses propriétés CSS indiquent des longueurs. Leur valeur est toujours un nombre suivi d'une unité :

- `px` : pour le pixel
- `pt` : pour le point (même unité que dans les traitements de texte)
- `%` : pourcentage de la taille de l'élément contenant

Couleur Il est possible de définir les couleurs par

- leurs noms (en anglais, par exemple `red`, `green`, `blue`, `LightGreen`, ettc.) ou
- la notation *hexadécimale* : $\#rrvvbb$ qui représente une couleur obtenue par un mélange d'une certaine quantité de rouge (rr), de vert (vv) et de bleu (bb) :
  - noir : si rr, vv et bb sont au minimum, alors la couleur obtenue est noire ;
  - blanc : si rr, vv et bb sont au maximum, alors la couleur obtenue est blanche.
  - r, v ou b sont des caractères qui représentent un nombre écrit en base 16. Ce nombre vaut 0 (valeur minimale), $1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e$ ou f (valeur maximale).

Exemple

La valeur minimale pour rr, vv ou bb est : `00`. Ainsi, la couleur noire est obtenue par le codage `#000000`.

La valeur maximale pour rr, vv ou bb est : `ff` (et oui !!! il faut savoir que ff en base 16 représente 255 en base 10). Ainsi, la couleur blanche est obtenue par le codage `#ffffff`.

ACTIVITÉ

Déterminer le codage hexadécimal permettant d'obtenir la couleur violette (mélange de rouge et de bleu uniquement).

CORRECTION

Le violet peut s'obtenir par `#ff00ff`.

Un violet plus foncé s'obtiendra en *se rapprochant* du noir : `#aa00aa` ou encore plus foncé `#550055`.

Un violet clair en *se rapprochant* du blanc : `#ff99ff` ou `#ffccff`.

Exemple

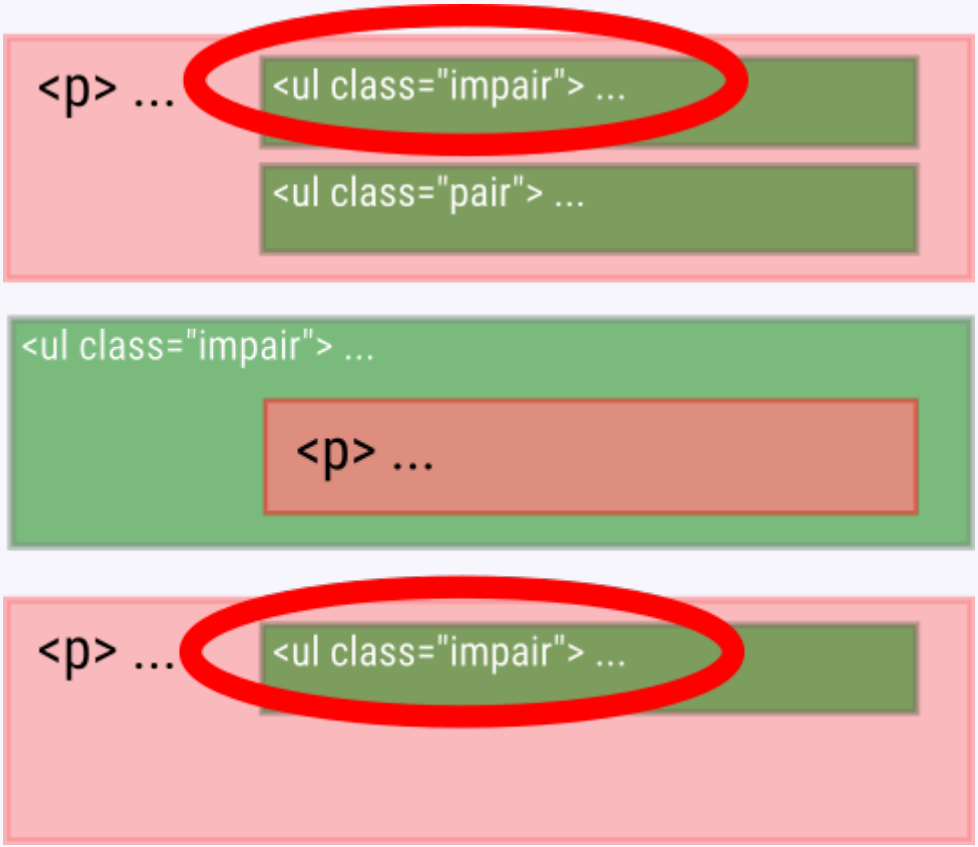
propriété	valeur	description
`display`	`none`, `block` ou `inline`	mode d'affichage de la boîte
`background`	couleur	couleur de fond de la boîte
`color`	couleur	couleur de texte
`border`	`none` ou `taille` `motif` `couleur`	Si `none` alors bordure masquée. Sinon on donne 3 propriétés séparées par des espaces : la *taille* (par ex `2pt`), le *motif* (`solid`, `dotted` ou `dashed`) et la *couleur* (par ex `green` ou `#ff00ff`).
`margin`	longueur	taille des marges

`padding`	longueur	taille des ajustements
`text-decoration`	`none`, `underline`, `overline` ou `line-through`	décoration du texte
`text-align`	`left`, `right`, `center` ou `justify`	justification du texte
`font-family`	`fixed`, `serif` ou `sans-serif`	nom de la police
`font-weight`	`normal`, `light`, `bold` ou `bolder`	graisse de la police
`font-style`	`normal` ou `italic`	style de la police
`font-size`	longueur ou `xx-small`, `x-small`, `small`, `normal`, `large`, `x-large` ou `xx-large`	taille de la police

Pour aller plus loin

Exemple

Par exemple, on souhaite modifier la couleur du contenu de toutes les balises `<ul>` de classe `impair` qui appartiennent aux paragraphes `<p>`.



Pour cela, on raisonne par inclusion : de la plus grande boîte à la plus petite.

Ici :

1. on sélectionne **toutes** les boîtes `<p>`,
2. puis **dans la sélection de l'étape précédente**, on sélectionne **toutes** les boîtes `<ul class="impair">`
3. puis **dans la sélection de l'étape précédente**, on modifie la propriété `color`.

Ce qui se traduit par le sélecteur suivant :

```
p ul.impair {color: ... ;}
```

Attention, l'ordre des sélecteurs est important !

Sélecteurs imbriqués De façon générale, une règle CSS est donnée sous la forme suivante :

```
p1 p2 ... pn { prop1 : valeur1 ;
    prop2 : valeur2 ;
    ...
    propk : valeurk ; }
```

avec ici n sélecteurs et k propriétés.

Comparer l'effet des sélecteurs suivant :

```
div#menu li a { color: pink ; background: yellow; }
```

et

```
div #menu li a { color: pink ; background: yellow; }
```

CORRECTION

1. On souhaite trouver tous les liens (`<a>`) qui sont à l'intérieur des éléments `<li>` qui sont eux-mêmes à l'intérieur de l'unique balise `div` dont l'identifiant est `menu`.
2. On cherche l'unique élément dont l'identifiant est `menu` s'il est contenu dans une balise `div` quelconque. Cet élément n'est pas forcément une balise `div`

Ce qui donne :

1. sélectionne l'élément tel quel :

```
<div id="menu">Contenu sélectionné</div>
```

2. sélectionne l'élément `p` :

```
<div>
    ...
    <p id="menu">Contenu sélectionné</p>
    ...
</div>
```

Cascade ? Enfin, si plusieurs style s'appliquent successivement à un même élément :

- les effets **se cumulent**
- et en cas de conflit c'est une règle compliquée... Pour les cas simples, c'est le dernier style rencontré qui est appliqué.