Introduction au CSS3

NSI Première

Académie de Versailles

M. Boehm, C. Poulmaire, P. Remy

Sommaire

Le langage HTML est à la fois un langage permettant de décrire la structure du texte, mais aussi de spécifier son apparence. Pour cette dernière, on utilise des *commandes CSS*. Ces commandes peuvent être placées directement dans la page HTML au moyen de la balise <style> du conteneur <head> ou dans un fichier séparé que la page HTML référence. C'est cette dernière utilisation qui est préconisée dans le HTML5.

1 Référencement d'un fichier CSS

Pour référencer un fichier CSS dans une page HTML, par exemple le fichier *style.css* situé dans le dossier courant, il faut placer la balise

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

dans le conteneur <head> du document HTML. La structure de base d'un fichier HTML devient alors :

```
<!DOCTYPE html>
<html lang='fr'>
     <head>
          <meta charset='UTF-8'>
          <link rel='stylesheet' href='chemin_fichier_de_style' type='text/css'>
          <title>Titre pour l'onglet de navigation</title>
     </head>
     <body>
          Texte à afficher sur la page web
     </body>
</html>
```

2 Création d'un fichier CSS

2.1 Règle de base

Pour créer un fichier CSS, il suffit de préciser la balise ainsi que le ou les attributs que l'on désire modifier. On utilise pour cela la syntaxe suivante :

```
nom_balise{attribut 1: valeur; attribut 2: valeur; ... attribut n: valeur;}
```

Remarque 2.1 Les balises n'ont pas toutes les mêmes attributs. Différents exemples sont donnés au paragraphe 4 ci-dessous et on consultera internet pour plus de détails.

2.2 Utilisation des identifiants et des classes

Lorsque l'on utilise la structure ci-dessus, toutes les balises de même nom sont mises en forme de la même façon. Si l'on veut modifier uniquement l'aspect d'une balise particulière ou d'un groupe de balises spécifique, on utlise les identifiants id et les classes class. La syntaxe générale devient alors :

- Pour un identifiant :

```
#nom_identifiant{attribut 1: valeur; attribut 2: valeur; ... attribut n: valeur;}
```

- Pour une classe :

```
.nom_classe{attribut 1: valeur; attribut 2: valeur; ... attribut n: valeur;}
```

3 Validation des feuilles de style CSS

De même qu'il existe un validateur permettant de vérifier la syntaxe HTML d'un document, le W3C fournit un validateur pour les fichiers CSS qui permet d'en vérifier la syntaxe et, en cas d'erreur, d'en être informé.
On trouvera ce validateur à l'adresse suivante : http://jigsaw.w3.org/css-validator/

4 Exemples de mises en forme

4.1 Mise en forme générale des éléments.

4.1.1 Marges

Pour chaque élément HTML, on peut donc définir l'espacement qui le séparera des autres éléments. On utilise pour cela les attributs suivants :

- ⋆ `margin` : permet de modifier les marges extérieures.
- ⋆ `padding` : permet de modifier les marges intérieures.

Les valeurs attribuées à ces deux attributs peuvent être fixes (valeurs en `px`, en `em`, etc...) ou proportionnelles (en `%`).

Remarque 4.1 On peut détailler les tailles des marges à l'aide des suffixes `-top`, `-right`, `-bottom` et `-left`.

Exemple 4.2 L'instruction

```
p{margin:3px; padding-left:5px;}
```

indiquent que, pour toutes les balises `<p>`, les marges extérieures sont de 3 pixels et que les marges intérieures gauches sont de 5 pixels (les autres marges intérieures ont des valeurs par défaut).

4.1.2 Bordures

- ⋆ `border-width` : permet de modifier la taille de la bordure ; l'usage est d'utiliser les pixels `px`.
- ⋆ `border-style` : permet de modifier le style de la bordure ; prend les valeurs `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset` ou `none`.
- ⋆ `border-color` : permet de modifier la couleur de la bordure.

Remarque 4.3 Les trois attributs doivent être présents pour afficher une bordure, mais on peut les synthétiser à l'aide de l'attribut `border` qui prend trois valeurs : la taille, le style et la couleur.

Exemple 4.4 L'instruction

```
table,th,td{border:1px solid black;}
```

indiquent que les balises `<table>`, `<th>` et `<td>` auront toutes une bordure solide et noire de 1 pixel. Ceci permet en particulier de délimiter les cases des tables.

Remarque 4.5 Les couleurs peuvent être définies de l'une des façons suivantes :

- ⋆ par des valeurs nommées prédéfinies (`black`, `yellow`, `blue`, etc...),
- ⋆ par un triplet RVB en notation hexadécimale (`#6A0888`, `#8BF1EF`, etc...),
- ⋆ par un triplet RVB en notation décimale (`rgb(255,255,0)`, `rgb(24,96,37)`, etc...).

On consultera internet pour plus de détails sur les valeurs de telle ou telle couleur en utilisant des palettes.

4.1.3 Tailles

Pour certains éléments comme les tables, les éléments sectionnants, etc..., il est souvent utile de modifier ou de spécifier leurs tailles. Pour cela, on utilise les attributs suivants :

- ⋆ `width` : permet de spécifier la largeur d'un élément.
- ⋆ `height` : permet de spécifier la hauteur d'un élément.

Les valeurs attribuées à ces deux attributs peuvent être fixes (valeurs en `px`, en `em`, etc...) ou proportionnelles (en `%`).

Exemple 4.6 L'instruction

```
#intro{width:50%; height:500px;}
```

indiquent que la balise d'identifiant `intro` aura une largeur occupant la moitié de la fenêtre du navigateur et une hauteur de 500 pixels.

4.1.4 Couleurs

- ⋆ `background-color` : permet de modifier la couleur de l'arrière-plan.
- ⋆ `color` : permet de modifier la couleur du texte.

Exemple 4.7 L'instruction

```
span{background-color:blue; color:red;}
```

indiquent que, pour toutes les balises `<span>`, l'arrière-plan de l'élément sera en bleu et le texte écrit sera en rouge.

4.2 Les types « bloc » et « en ligne »

- ⋆ `display` : permet de modifier le type d'affichage ; prend les valeurs `block`, `inline` ou `none`.
  Les deux premières valeurs permettent de permuter le type d'un élément entre « bloc » et « en ligne ». La troisième valeur permet « d'effacer » l'élément ciblé sans laisser de trace : les éléments précédents et suivants sont positionnés comme si l'élément visé n'existait pas.
- ⋆ `visibility` : permet de cacher un élément tout en conservant occupé l'espace dans lequel il aurait été visible ; prend les valeurs `visible` (par défaut) ou `hidden`.

Exemple 4.8 L'instruction

```
.cache{visibility:hidden;}
```

indique que les éléments de classe `cache` sont cachés, mais que l'espace qu'ils auraient occupés est conservé.

4.3 Les textes

- ⋆ `font-family` : permet de modifier la police utilisée.
- ⋆ `font-size` : permet de modifier la taille de la police.
- ⋆ `font-style` : permet de modifier le style de la police (prend les valeurs `italic`, `oblique` ou `normal`).
- ⋆ `font-weight` : permet de modifier le gras de la police (prend les valeurs `bold` ou `normal`).
- ⋆ `text-align` : permet de modifier l'alignement du texte (prend les valeurs `left`, `right`, `center` ou `justify`).
- ⋆ `text-decoration` : permet de modifier la décoration du texte (prend les valeurs `underline`, `overline`, `line-through` ou `none`).

Exemple 4.9 L'instruction

```
h1{text-align:center; font-style:italic; text-decoration:underline;}
```

indiquent que, pour toutes les balises `<h1>`, les textes seront centrés, soulignés et écrits en italique.

4.4 Les liens hypertextes

Pour modifier la mise en forme d'un lien hypertexte (couleur, police, etc...), on procède comme précédemment en agissant sur les attributs de la balise `<a>`. Toutefois, il existe des pseudo-classes associées à cette balise qui permettent de gérer la mise en forme du lien suivant l'action que l'on effectue (survol du lien, clic, etc...) :

- ⋆ `a:link` : représente l'état d'un lien par défaut.
- ⋆ `a:visited` : représente l'état d'un lien pointant sur une page déjà visitée et présente dans l'historique du navigateur.
- ⋆ `a:hover` : représente l'état d'un lien lorsque l'on passe la souris dessus.
- ⋆ `a:focus` : représente l'état d'un lien qui a reçu « l'attention », par exemple lorsqu'on y accède grâce au clavier.
- ⋆ `a:active` : représente l'état d'un lien lorsque l'on clique dessus.

Exemple 4.10 Les instructions

```
a:link{color:green;}
a:visited{color:red; font-style:italic;}
a:hover{font-size:18pt; color:blue;}
a:active{color:maroon;}
```

indiquent que les liens non visités sont par défaut en vert, que les liens visités sont rouges et en italique, que les liens survolés sont bleus et écrits en taille 18 points et que les liens sur lesquels on clique sont en marron.

Remarque 4.11 Les pseudo-classes `:link` et `:visited` sont réservées aux liens hypertextes. En revanche, les trois autres pseudo-classes `:hover`, `:focus` et `:active` peuvent s'appliquer à d'autres éléments. On consultera internet pour plus de détails.

4.5 Les images

4.5.1 Images de fond

- ⋆ `background-image` : permet d'insérer une image en fond.
- ⋆ `background-attachment` : permet de rendre fixe le fond de la page (utiliser la valeur `fixed`).
- ⋆ `background-repeat` : permet d'empêcher que l'image soit insérée en mosaïque (utiliser la valeur `no-repeat`, l'image se place alors par défaut en haut à gauche de l'écran).
- ⋆ `background-position` : permet de positionner l'image ; prend deux valeurs séparées par un espace (la première valeur est `right`, `center`, `left`, une valeur fixe ou une valeur proportionnelle ; la seconde valeur est `top`, `center`, `bottom`, une valeur fixe ou une valeur proportionnelle).
- ⋆ `background-size` : permet de redimensionner l'image de fond ; prend deux valeurs en `px` ou `%` séparées par un espace (la première valeur est la largeur et la seconde la hauteur de l'image).

Exemple 4.12 L'instruction

```
body{background-image:url(MonImage.png); background-repeat:no-repeat;}
```

indique que l'image `MonImage.png` (situé dans le dossier courant) est mise en fond et n'est pas en mosaïque.

Remarque 4.13 L'attribut `background-image` admet également pour valeur `linear-gradient` et `radial-gradient` qui permettent de réaliser des dégradés de couleurs. On renvoie à internet pour plus de détails.

4.5.2 Images dans la page

- ⋆ `width` : permet de fixer la largeur de l'image (en `px` ou `%`).
- ⋆ `height` : permet de fixer la hauteur de l'image (en `px` ou `%`).

Remarque 4.14 Pour garder le ratio d'une image lors d'un redimensionnement, il suffit d'indiquer la largeur ou la hauteur voulue, puis d'attribuer la valeur `auto` à l'autre dimension.

Exemple 4.15 L'instruction

```
img#image1{width:100px; height:auto;}
```

indique que l'image d'identifiant `image1` a pour largeur 100 pixels et que sa hauteur est calculée automatiquement pour garder le ratio initial.

Remarque 4.16 Pour centrer une image, il faut transformer l'image en une structure de type « bloc » puis modifier les marges à gauche et à droite de l'image. Pour ce faire, on utilise simultanément les trois instructions suivantes : `display:block; margin-left:auto; margin-right:auto;`.

4.6 Les listes

- ⋆ `list-style-type` : permet de modifier le type de puces utilisé dans les listes ordonnées et non ordonnées.
  - Pour les listes ordonnées, on peut utiliser par exemple les valeurs `decimal` (par défaut), `lower-roman` (i, ii, ...), `upper-roman` (I, II, ...), `lower-greek` (α, β, ...), `lower-latin` (a, b, ...), `upper-latin` (A, B, ...), etc...
  - Pour les listes non ordonnées, on peut utiliser les valeurs `disc`, `circle` ou `square`.

Exemple 4.17 L'instruction

```
ol.normal{list-style-type:upper-latin;}
```

indique que la liste ordonnée de classe `normal` sera numérotée avec les valeurs A, B, C, etc...

4.7 Les tables

- ⋆ Les balises `table`, `th` et `td` supportent tous les attributs sur les textes, marges, bordures, couleurs et images donnés ci-dessus. Elles supportent également l'attribut `width` permettant de spécifier des largeurs.
- ⋆ La balise `table` admet également l'attribut `border-collapse` permettant de dissocier les cellules avec la valeur `separate` (valeur par défaut) ou de coller les cellules avec la valeur `collapse`.
- ⋆ Pour positionner le titre au-dessus ou au-dessous du tableau, on utilise l'attribut `caption-side` de la balise `caption` qui prend les valeurs `top` (par défaut) ou `bottom`.
- ⋆ Pour centrer une table, il faut l'inclure dans une balise `div` puis combiner les instructions `text-align:center;` pour `div` et `margin:auto;` pour `table`.

Exemple 4.18 Les instructions

```
table,th,td{border:1px solid red;}
td{background-color:blue;}
```

indique que la table et les cellules d'entête et de données ont une bordure de 1 pixel, solide et rouge et que les cellules de données ont un arrière-plan bleu.

4.8 Le pointeur de la souris

- ⋆ `cursor` : indique la forme du curseur de la souris lorsqu'il survole un élément (ne nécessite pas d'utiliser la pseudo-classe `:hover`). On a par exemple les valeurs suivantes :
  - `auto` : pointeur défini par le navigateur en fonction de l'endroit où se trouve la souris,
  - `default` : pointeur du système d'exploitation (en général une flèche),
  - `help` : pointeur de l'aide,
  - `pointer` : une main indiquant que l'on peut cliquer,
  - `text` : indique une saisie de texte,
  - `wait` : indique qu'une tâche est en cours de traitement (en général un sablier).

Remarque 4.19 Il existe plusieurs autres valeurs pour personnaliser le curseur. On renvoie à internet pour plus de détails.

Exemple 4.20 L'instruction

```
.cliquable{cursor:pointer;}
```

indique que le pointeur de la souris est représenté sous la forme d'une main lorsqu'il survole les éléments de classe `cliquable`.

4.9 Les transformations 2D

- ⋆ `transform:rotate(`*`val`*`deg)` : effectue une rotation de `val` degrés ; la rotation est effectuée par rapport au centre de l'élément. La valeur `val` peut être négative.
- ⋆ `transform:scale(`*`val`*`)` : effectue un agrandissement de l'élément si `val` est > 1 et une réduction si `val` est < 1. On peut également utiliser les syntaxes suivantes pour réaliser un agrandissement/réduction uniquement sur l'un des axes :
  - `transform:scaleX(`*`val`*`)` : effectue un agrandissement/réduction de l'élément uniquement sur l'axe des x.
  - `transform:scaleY(`*`val`*`)` : effectue un agrandissement/réduction de l'élément uniquement sur l'axe des y.
- ⋆ `transform:translate(`*`val1`*`,`*`val2`*`)` : effectue une translation de l'élément de `val1` sur l'axe des x et `val2` sur l'axe des y (les valeurs sont généralement en pixels et peuvent être négatives). On peut également utiliser les syntaxes suivantes pour réaliser une translation uniquement par rapport à l'un des axes :
  - `transform:translateX(`*`val`*`)` : effectue une translation de l'élément de `val` sur l'axe des x.
  - `transform:translateY(`*`val`*`)` : effectue une translation de l'élément de `val` sur l'axe des y.
- ⋆ `transform:skewX(`*`val`*`deg)` et `transform:skewY(`*`val`*`deg)` : effectue une transformation sur l'élément de façon à ce qu'il devienne un losange. La valeur `val` est en degré et peut être négative.

Remarque 4.21 Ces transformations peuvent être combinées, mais attention à l'ordre. D'autre part, pour le navigateur `Safari`, il est nécessaire d'écrire `-webkit-transform` au lieu de `transform`.

Exemple 4.22 L'instruction

```
#photo{transform:scale(0.7) rotate(15deg) translate(50px,75px);}
```

indique que l'élément d'identifiant `photo` est réduit de 30%, puis tourné de 15 degrés par rapport à son centre et enfin translaté de 50 pixels en abscisse et de 75 pixels en ordonnées.

Remarque 4.23 Les instructions précédentes peuvent être généralisées à des transformations 3D en ajoutant une coordonnée.

4.10 Les transitions et les animations

4.10.1 Les transitions

Celles-ci permettent de transformer un élément d'un état A en un état B.

- ⋆ `transition-property` : indique quels attributs `CSS` sont animés lors de la transition ; par défaut, tous mais on peut préciser ceux que l'on veut en les séparant par des virgules.
- ⋆ `transition-duration` : indique combien de temps dure la transition.
- ⋆ `transition-delay` : indique au bout de combien de temps commence la transition (0 par défaut).
- ⋆ `transition-timing-function` : indique quel type de transition on peut voir durant la transition. Les valeurs prédéfinies possibles sont :
  - `ease` : rapide au début et ralenti à la fin,
  - `linear` : vitesse constante (par défaut),
  - `ease-in` : lent au début et accélère de plus en plus à la fin,
  - `ease-out` : rapide au début et de plus en plus lent à la fin,
  - `ease-in-out` : lent au départ et à la fin.

  On peut également utiliser la valeur `cubic-bezier()` pour définir des courbes afin de paramétrer finement la fluidité de la transition. On renvoie à internet pour plus de détails.

Remarque 4.24 Pour définir une transition, il faut au minimum indiquer la valeur de `transition-duration`. Une transition peut s'activer lors d'un `:hover`, `:active` ou `:focus`, et également lors d'un événement `JavaScript`.

Exemple 4.25 Les instructions

```
div{width:200px; height:150px; background-color:blue;
    transition-property:background-color,width;
    transition-duration:1s;
    transition-timing-function:ease-in-out;}
div:hover{width:500px; height:500px; background-color:red;}
```

indiquent que les éléments `div` ont initialement une largeur de 200 pixels et une hauteur de 150 pixels et qu'ils ont un arrière-plan bleu ; puis, en les survolant qu'ils deviennent carrés de côté 500 pixels avec un arrière-plan rouge, les transitions visibles étant sur la largeur et l'arrière-plan, durant 1 seconde et étant rapide au début, puis ralenti à la fin.

Remarque 4.26 On peut également utiliser l'attribut `transition` qui permet de regrouper les quatre attributs précédents. Celui-ci prend entre deux et quatre valeurs :

- ⋆ la première valeur est obligatoire et vaut soit `all` soit le nom de l'attribut sur lequel on veut faire agir la transition,
- ⋆ la deuxième valeur est obligatoire et définie la durée de la transition,
- ⋆ la troisième valeur est optionnelle et définie le délai d'attente pour lancer la transition,
- ⋆ la quatrième valeur est optionnelle et indique le type de transition.

Il est possible d'avoir plusieurs effets de transition en parallèle avec des paramètres différents. Pour cela, il suffit de les séparer par des virgules.

Exemple 4.27 Les instructions

```
div{width:200px; height:150px;
   transition:transform 1s, background-color 1.5s 1s ease-out;}
div:hover{transform:translate(200px,200px) scale(2) rotate(45deg);
          background-color:blue;}
```

indiquent que les éléments `div` sont initialement de taille 200×150 ; puis, en les survolant qu'ils sont translatés de 200 pixels en abscisse et en ordonnée, doublés en dimension et tournés de 45 degré (ceci en 1 seconde) et que leur arrière-plan devient bleu en $1,5$ seconde après un délai de 1 seconde (pour avoir l'impression que les deux transitions s'enchaînent) et avec une transition rapide au début et lente à la fin.

4.10.2 Les animations

Celles-ci permettent d'enchaîner plusieurs transitions, c'est-à-dire de transformer un élément d'un état A à un état B en passant par plusieurs états intermédiaires.

- ⋆ `@keyframes` *nom_anim*{*étapes_animation*} : permet de définir les étapes de l'animation dont le nom est *nom_anim*. Pour définir une étape de l'animation, on utilise la syntaxe *val*%{*définition_étape*}, où *val* est un nombre indiquant le pourcentage de l'animation qui a été effectuée et où *définition_étape* représente ce que l'on veut faire dans l'animation.

  Exemple 4.28 Les instructions

```
@keyframes MonAnimation{
25%{transform:translate(100px,100px);}
50%{transform:translate(100px,100px) scale(2);}
75%{transform:translate(100px,100px) scale(2) rotate(45deg);
    background-color:green;}
100%{transform:translate(100px,100px) scale(2) rotate(45deg) scaleX(0.5);
     background-color:blue;}
                        }
```

  définissent l'animation `MonAnimation` où, à 25%, on translate l'élément de 100 pixels en abscisse et en ordonnée, à 50%, on double la taille de l'élément, à 75%, on effectue une rotation de l'élément de 45 degré et on met son arrière-plan en vert et où à la fin (100%), on fait une dilatation en abscisse de $0,5$ et on met l'arrière-plan en bleu. Remarquer qu'il faut répéter les instructions déjà effectuées à une étape précédente (notamment les déplacements) pour avoir une impression de fluidité.

- ⋆ `animation-name` : indique le nom de l'animation à utiliser.
- ⋆ `animation-duration` : indique la durée de l'animation.
- ⋆ `animation-delay` : indique au bout de combien de temps commence l'animation (0 par défaut).
- ⋆ `animation-timing-function` : indique quel type d'animation on peut voir. Les valeurs prédéfinies possibles sont :
  - `ease` : rapide au début et ralenti à la fin,
  - `linear` : vitesse constante (par défaut),
  - `ease-in` : lent au début et accélère de plus en plus,
  - `ease-out` : rapide au début et de plus en plus lent,
  - `ease-in-out` : lent au départ et à la fin.

  On peut également utiliser la valeur `cubic-bezier()` pour définir des courbes afin de paramétrer finement la fluidité de la transition et la valeur `step` pour définir le nombre d'étapes à effectuer. On renvoie à internet pour plus de détails.
- ⋆ `animation-iteration-count` : indique le nombre de répétitions de l'animation (par défaut 1). Avec la valeur `infinite`, l'animation est jouée en continue.
- ⋆ `animation-direction` : indique comment se déroule l'animation. Les valeurs prédéfinies possibles sont :
  - `normal` : l'animation se déroule dans l'ordre et, une fois terminée, l'élément revient à sa position initiale (par défaut),
  - `reverse` : l'animation se déroule dans le sens inverse et, une fois terminée, l'élément revient à sa position de départ (donc la fin ici),
  - `alternate` : l'animation se déroule dans l'ordre puis l'élément revient à sa position initiale en réalisant l'animation dans le sens inverse.
- ⋆ `animation-fill-mode` : indique comment l'élément doit être affiché une fois l'animation terminée. Les valeurs prédéfinies possibles sont :
  - `none` : la position est gérée par le navigateur (par défaut),
  - `forwards` : indique que l'élément reste à sa position finale,
  - `backwards` : indique que l'éléement revient à sa position initiale.

Remarque 4.29 Comme pour les transitions, on peut utiliser l'attribut `animation` qui permet de regrouper tous les attributs précédents. Celui-ci prend entre deux et sept valeurs :

- ⋆ la première valeur est obligatoire et indique le nom de l'animation à utiliser,
- ⋆ la deuxième valeur est obligatoire et indique la durée de l'animation,
- ⋆ les cinq autres valeurs sont optionnelles et indiquent le délai d'attente, le type d'animation, le nombre de fois où l'animation est jouée, le déroulement de l'animation et la position de l'élément une fois l'animation terminée.

Exemple 4.30 On reprend l'animation de l'exemple 4.28. Les instructions

```
div{width:200px; height:150px; background-color:red;}
div:hover{animation:MonAnimation 4s infinite alternate;}
```

indiquent que, lors du survol d'un élément `div`, l'animation `MonAnimation` est jouée en boucle, dans l'ordre donné puis en sens contraire et que chaque séquence dure 4 secondes.