Le langage CSS et l'utilisation des styles

Chap. 19

Table des matières

Les premières version du langage HTML mélangeaient les *éléments dans la page* (texte, image, ...) et la *mise en forme de ces éléments* (mettre en gras, définir un fond coloré, ...).

> *Aucune séparation entre le fond (l'information) et la forme (comment l'information est présentée) n'existait.*

Ce mélange est à éviter pour plusieurs raisons :

- Séparer le fond de la forme permet de se concentrer davantage sur « le message » délivré, *dans un premier temps*, et sur sa présentation, *dans un second temps* ;
- Une page HTML peut être lue depuis différents « lecteurs » : navigateur sur ordinateur, navigateur sur tablette ou téléphone, lecteur pour mal-voyant, ... La séparation entre la structure d'une page et son apparence permet d'adapter cette dernière au média utilisé (Un fond noir à l'écran peut embellir une page, il peut par contre devenir gênant si le lecteur essaie d'imprimer la page.).

Pour se convaincre de l'intérêt de dissocier le fond de la forme et apprécier la créativité dont certains sont capables, consulter cette référence : http://www.csszengarden.com.

Le langage de spécification du style des pages s'appelle « *Cascading Style Sheets* » ou en Français « *Feuilles de Style en Cascade* » (sigle CSS). Quelques uns de ses principes seront abordés dans ce chapitre.

1 Références

- http://www.w3schools.com (en anglais mais interactif !).
- https://developer.mozilla.org/fr/docs/Web/CSS.
- http://fr.openclassrooms.com/informatique/css/cours.

2 Comment définir les styles ?

Le langage CSS définit un ensemble de paramètres possédant chacun une ou plusieurs valeurs. Où doivent-ils être placés ?

2.1 Utilisation de l'attribut style dans une balise

On installe dans la balise ouvrante voulue un attribut de style avec pour valeur les spécifications que l'on souhaite définir, chacune étant terminée par un point-virgule `;`. Chaque spécification est de la forme : `paramètre:valeur`. Par exemple, `<p style="color:red; background-color:yellow;">` définit un paragraphe dont le fond est jaune et le texte rouge.

Ces spécifications s'appliquent *uniquement* au contenu de la balise concernée.

2.2 Utilisation de la balise `style` dans l'entête

On peut placer les règles entre les balises `<style>` et `</style>`, dans l'entête du fichier HTML. Entre ces balises, une règle s'écrit alors : `sélecteur {spécification}`.

Les sélecteurs indiquent à quoi s'appliquent les spécifications. *Pour le moment, il suffit de savoir qu'un sélecteur peut être un nom de balise.* Par exemple :

```
<style>
    p {color: green;}
</style>
```

déclare que le texte de *tous* les paragraphes (contenus des balises `<p>`) doit apparaître en vert.

2.3 Utilisation de fichiers CSS extérieurs – méthode à préférer

On place les règles dans un *fichier extérieur* au fichier HTML source. Ce fichier s'appelle alors *une feuille de style*.

1. Le fichier doit posséder l'extension `.css`. C'est aussi un fichier texte (il faut donc utiliser un éditeur de texte pour le créer).

2. Le contenu du fichier est identique au contenu de la balise `<style>`.
   Les balises `<style>` et `</style>` ne doivent pas être utilisées.

3. Une liaison doit être établie entre le fichier `HTML` et la feuille de style par une balise `<link ... />` dans l'entête du fichier `HTML`.
   Par exemple, si le fichier s'appelle `style1.css` et si il est situé dans le même répertoire que le fichier HTML, la balise s'écrit, dans l'entête (entre les balises `<head>` et `</head>`) du fichier HTML :

   ```
   <link rel="stylesheet" type="text/css" href="./style1.css" />
   ```

   Si le fichier CSS ne se trouve pas dans le même dossier que le fichier HTML, il est plus que conseillé d'utiliser son chemin relatif dans l'attribut `href`[1].

2.4 Avantages de l'utilisation d'un fichier CSS

Le fait de regrouper les spécifications de mise en forme dans un fichier extérieur a plusieurs avantages :

– On obéit ainsi parfaitement à la recommandation de séparer la *sémantique de la page* et sa *mise en forme*.

– Un même fichier CSS peut être *utilisé par plusieurs pages Web*, et même par toutes les pages d'un site. En plus de l'économie d'écriture évidente, cela permet d'assurer une *homogénéité de présentation des pages*, ce qui donne au site une plus grande cohérence.

1. Cf. le chapitre 18 pour la définition des liens relatifs.

2.5 Hiérarchie des spécifications

Le sigle CSS est l'abréviation de *Cascading Style Sheets* (feuilles de style en cascade). « Cascade » signifie que plusieurs spécifications peuvent s'appliquer à un certain contenu : une ou plusieurs feuilles de style, puis une balise `<style>` et enfin des spécifications dans la balise concernée. En principe, les différentes spécifications se complètent en adressant des dispositions différentes. Si elles se contredisent, une hiérarchie existe :

1. Les spécifications du fichier CSS extérieur s'appliquent en premier.
2. Si une balise `<style>` est présente *après la balise* `<link ... />` dans l'entête du fichier HTML, les dispositions contraires qu'elle contient prennent alors le pas.
3. Finalement, si la balise dont on veut mettre en forme le contenu possède des règles de style, ces dernières finissent par s'imposer.

3 Validation d'une feuille de style

Le langage CSS définit des sélecteurs, des paramètres, des valeurs pour ces paramètres de façon très rigoureuse. L'ensemble est normalisé par le W3C. Il est nécessaire de vérifier la correction de chacune des feuilles de style auprès de cet organisme (à l'instar des fichiers HTML).

L'adresse à consulter est : http://jigsaw.w3.org/css-validator/.

4 Les sélecteurs

– Dans un fichier CSS ou dans une balise `<style>`, une règle prend la forme :

```
selecteur {paramètre1:valeur1; paramètre2:valeur2; ...}
```

ou, si on souhaite faciliter la lecture et la compréhension du code :

```
selecteur {
          paramètre1:valeur1;
          paramètre2:valeur2;
          ...
          }
```

– Comme attribut d'une balise, les spécifications s'écrivent :

```
<balise style="paramètre1:valeur; paramètre2:valeur; ...">
```

– Dans un fichier CSS les sélecteurs peuvent être groupés ou séparés par des virgules `,`.

Un même élément peut intervenir dans plusieurs règles. Si ces règles font intervenir des *paramètres différents, elles se complètent*. Si elles concernent le *même paramètre et si elles se contredisent, c'est la dernière lue qui l'emporte*.

4.1 Choisir une balise comme sélecteur

Si l'on souhaite qu'une règle de style s'applique à **toutes les occurrences d'une balise**, on définit une règle unique *en utilisant le nom de cette balise dans le fichier CSS*. Ainsi,

```
h1, h2 {font-family: arial;}
```

indique que la police Arial sera utilisée pour *toutes les balises* `<h1>` *et* `<h2>`.

4.2 Choisir un identifiant comme sélecteur

Si l'on souhaite qu'une règle de style s'applique à seulement **une occurrence d'une balise**, on utilise un **identifiant** à l'aide de l'attribut `id`, dans la balise. Par exemple,

```
<div id="intro"> ... </div>
```

ajoute l'identifiant *intro* à la balise `<div>` concernée, dans le fichier HTML.

Cet **identifiant** peut servir à attribuer un style à la balise dans laquelle il apparaît. Par exemple,

```
#intro {background-color: silver;}
```

dans le fichier CSS, impose un fond gris à tout le contenu de la balise `<div>`.

Remarque. Le nom de l'identifiant doit donc être précédé du symbole `#` dans la feuille de style.

4.3 Choisir une classe comme sélecteur

Si l'on souhaite qu'une règle de style s'applique à seulement **quelques fois (pour des balises peut-être différentes)**, on utilise une **classe** à l'aide de l'attribut `class`, dans la balise. Par exemple,

```
<div class="important"> ... </div>
```

ajoute la **classe** *important* à la balise `<div>` concernée, dans le fichier HTML.

Cette **classe** peut servir à attribuer un style à la balise l'incorporant. Par exemple,

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

dans le fichier CSS, indique que *toutes les balises* qui possèdent l'attribut `class` avec la valeur *important* auront des caractères écrits en rouge. De même,

```
a.important {font-size: 1.2em;}
```

dans le fichier CSS, indique que la taille des caractères des contenus de *toutes les balises* `<a>` qui possèdent l'attribut `class` avec la valeur *important* sera 1,2 fois plus grand que la largeur d'un `m` standard dans le reste du document.

Remarque. Le nom de la **classe** doit donc être précédé du symbole `.` dans la feuille de style ou du nom de la balise.

4.4 Sélection par ancêtre

EF signifie « *tout élément F qui est un descendant d'un élément E* ».

L'ancêtre peut être défini par *classe* :

```
.important li
```

ou par *balise* :

```
p strong
```

Exemple.

Fichier CSS :

```
li {
    list-style-type: disc;
   }
li li {
       list-style-type: circle;
      }
```

Fichier HTML :

```
<ul>
  <li>
    <div>Élément 1</div>
    <ul>
      <li>Sous-élément A</li>
      <li>Sous-élément B</li>
    </ul>
  </li>
  <li>
    <div>Élément 2</div>
    <ul>
      <li>Sous-élément A</li>
      <li>Sous-élément B</li>
    </ul>
  </li>
</ul>
```

Résultat :

- Élément 1
  - Sous-élément A
  - Sous-élément B
- Élément 2
  - Sous-élément A
  - Sous-élément B

4.5 Sélection par parent

$E > F$ signifie « *tout élément F qui un enfant direct d'un élément E* ».

Ainsi,

```
#intro>strong { ... }
```

s'applique à toutes les balises `<strong>` contenues *en première position* dans la division d'identifiant `intro`. Dans le code HTML ci-dessous :

```
<div id="intro">
     <p><strong> Introduction </strong></p>
     <p>Il est très <strong>important</strong> de ... </p>
</div>
```

le mot « Introduction» est concerné, mais pas le mot « important».

4.6 Choix parmi les « dépendants »

E: first-child signifie tout élément E premier enfant d'un autre élément.

Ainsi `#suite p:first-child` concerne *le premier paragraphe contenu dans un conteneur possédant l'attribut* `suite`.

```
<div id="suite">
     <!-- paragraphe concerné -->
     <p> ... </p>
     <!-- paragraphe non concerné -->
     <p> ... </p>
 </div>
```

4.7 Sélection par « frère » précédent

$E+F$: tout élément F directement précédé d'un élément E.

Ainsi `#suite a+li { ... }` concerne une balise `<a>` contenue dans un conteneur possédant l'attribut `id="suite"` à condition qu'elle soit immédiatement précédée par une balise `<li>`.

4.8 Sélection d'une partie d'élément

4.8.1 Première ligne

```
p:first-line { ... }
```

agit sur la première ligne de chaque paragraphe.

4.8.2 Première lettre

```
p:first-letter { ... }
```

agit sur la première lettre de chaque paragraphe. C'est un moyen de créer des lettrines.

4.9 Sélection d'après l'état (pseudo-classes)

Certaines balises possèdent des « états » (l'exemple le plus évident est la balise `<a>` qui définit des liens. Ceux-cipeuvent être « *pas encore visité* », « *déjà visité* », « *survolés* », ...).

Il est possible de faire en sorte qu'elles apparaissent différemment en fonction de cet état : `a:link`, `a:visited`, `a:focus`, `a:hover`, `a:active` dans l'ordre, définissent l'état du lien *pas encore visité*, *visité*, *sélectionné* (par tabulation), *survolé* (la souris est dessus), *actif* (bouton de la souris enfoncé).

```
a:link {color: red;} /* lien non-visité */
a:visited {color: blue;} /* lien visité */
a:hover {color: yellow;} /* lien survolé */
a:active {color: lime;} /* lien activé */
```

Les éléments comme `:hover` s'appellent des *pseudo-classes*. Il existe d'autres pseudo-classes comme `:before` ou `:after`, mais tous les navigateurs ne les interprètent pas encore (ou alors de façon un peu aléatoire)[2].

Exercice 1.

Décoder le sélecteur : `div.important p strong:first-letter`.

2. Plus d'informations sur les pseudo-classes : http://www.yoyodesign.org/doc/w3c/css2/selector.html#dynamic-pseudo-classes

Solution. Le premier caractère contenu dans toute balise `<strong>` présente dans un paragraphe `<p>` appartenant à une division `<div>` possédant l'attribut `class="important"` apparaît en gras.

Exercice 2.

Quelle est la différence entre `a.important` et `.important a` ?

Solution. La première règle concerne toutes les balises `<a>` possédant l'attribut `class="important"` alors que la seconde concerne toutes les balises `<a>` contenues dans des conteneurs appartenant à la classe « important ».

4.10 Stratégies de sélection

4.10.1 Mise en forme globale

Pour définir une mise en forme qui s'applique à tous les éléments de la page, il suffit de l'adresser à la balise `<body>` dans le fichier HTML. Les spécifications sont alors « héritées» par toutes les balises puisqu'elles sont contenues dans le conteneur <body>.

Le mécanisme de l'héritage évoqué à la sous-section 2.5 peut s'avérer délicat à gérer[3], quelques règles sont définies ci-dessous.

4.10.2 Définition de classes

L'utilisation des classes est la technique la plus simple à utiliser, *à condition de choisir des noms de classes explicites*.

Par exemple, pour faire des paragraphes d'importance croissante, on peut écrire :

```
p.normal {
         font-size: 1em;
         }
p.important {
         font-size: 1.2em;
         }
p.fondamental {
           font-size: 1.5em;
           color:red;
           }
```

ou, pour un contraste encore plus important,

```
p.important {font-size: 1.5em;}
p.fondamental {
             font-size: 2em;
             font-weight: 700;
             }
```

4.10.3 Identifiant ou classe ?

Un identifiant ne peut s'appliquer qu'à un seul élément alors qu'une classe peut être attribuée à plusieurs éléments. Donc, si une mise en forme doit s'appliquer à un grand nombre de paragraphes (par exemple), il faut définir une classe.

3. http://openweb.eu.org/articles/cascade_css

Au contraire, si une mise en forme doit s'appliquer à *un élément unique*, l'utilisation d'un identifiant est plus judicieuse : elle permet de s'assurer que les règles définies ne sont pas utilisées, par erreur, pour une autre balise.

L'identifiant peut désigner une partie bien définie d'une page avec une balise `<div>`. Par exemple, dans le fichier HTML, on peut écrire :

```
<div id="bandeau_de_navigation"> ... </div>
<div id="centre_de_page"> .... </div>
<div id="pied_de_page"> .... </div>
```

pour structurer la page et, pour avoir une mise en forme particulière pour les liens, dans le bandeau de navigation, un style définit par

```
#bandeau_de_navigation a { ... }
```

4.10.4 `<div>` et `<span>`

Comme évoqué au chapitre 18, les balises `<div>` et `<span>` ont un rôle de structuration de la page. Elles n'apportent aucun « sens » au code et ne déclenchent, par défaut, aucun affichage particulier lorsqu'elles sont interprétées par un navigateur.

- La balise `<div>` est de type « bloc » et définit donc un conteneur capable de regrouper d'autres blocs ou des éléments en-ligne.

- La balise `<span>` est de type « en-ligne ». Elle ne peut délimiter qu'un fragment de texte.

Exemple de structuration d'une page

```
<body>
    <div id="entete_page">
        ...
    </div>
    <div id="menu">
        <ul>
            <li><a href="index.html">Accueil</a></li>
            <li><a href="page01.html">Découverte du langage HTML</a></li>
            <li><a href="page02.html">Découverte du langage CSS</a></li>
            <li><a href="page03.html">Quelques exemples</a></li>
        </ul>
    </div>
    <div id="centre_page">
        ...
    </div>
    <div id="pied_page">
        ...
    </div>
</body>
```

5 Les spécifications

Pour affecter une valeur à une paramètre, on écrit `paramètre:valeur;` Il n'y a pas de guillemets autour de la valeur sauf présence d'espaces ou de caractères spéciaux.

Les valeurs peuvent être nominales (à choisir parmi une liste de possibilités) ou numériques. Les valeurs numériques s'appliquent le plus souvent à des dimensions : la taille de caractères, des dimensions de blocs, des marges ou des fenêtres, ...Elles peuvent être *absolues* ou *relatives* (exprimées en %).

Pour les longueurs, à l'écran, l'unité principale est *le pixel*, `px`.

Pour les dimensions des polices de caractères, l'unité la plus indiquée est le cadratin, `em`. C'est la taille du caractère « m». *Elle est* ***absolue*** *si définie dans la balise* `<body>` *où elle fixe la taille générale des caractères et* ***relative*** *partout ailleurs*. Si l'attribut `font-size`, dans une balise, a pour valeur `1.5em`, cela signifie que la police aura « une fois et demie la taille de référence» qui est soit la taille par défaut, soit la taille héritée depuis un élément contenant l'élément en question.

L'unité `em` est aussi très indiquée pour les marges : les marges et les caractères varient ainsi conjointement lorsqu'on utilise la fonction zoom du navigateur Web.

Les éléments installés par les balises se classent en éléments blocs (division, éléments de listes, paragraphes, ...) et éléments en-ligne (morceaux de texte, ...) comme évoqué à la sous-section ?.

Toutes les règles pour les éléments en-ligne s'appliquent aussi aux blocs mais l'inverse n'est pas vrai.

6 Styles de texte

6.1 Familles de polices de caractères

Le paramètre `font-family` permet de spécifier une police ou une famille de polices de caractères : `{font-family: arial;}` ou `{font-family: sans-serif;}`.

Lorsqu'on indique une famille de polices de caractères, le navigateur Web utilise une police de cette famille. Si une seule police a été déclarée, le navigateur l'utilise si elle est disponible sur l'ordinateur qui « lit » la page Web. Dans le cas contraire, il utilise une police « voisine » de la police demandée ; le résultat peut être tr!s aléatoire ! Il est donc préférable de spécifier une famille de polices de caractères, en principe 3, que le navigateur essaie d'utiliser dans l'ordre : `{font-family: helvetica, arial, sans-serif;}`.

Remarque : il peut aussi être intéressant de déclarer des polices de caractères de plusieurs sortes : *serif* (avec empattement), *sans-serif* (sans empattement), *cursive*, *fantasy* et *monospace* (tous les caractères ont la même largeur).

6.2 Taille des caractères

La taille des caractères est spécifiée grâce au paramètre `font-size`. Il est plus efficace de la spécifier en relatif en utilisant `em` ou %.

6.3 Choix des caractères

`font-style`	normal	oblique	italic		
`font-weight`	normal	bold	bolder	lighter	400
`font-weight`	500	600	700	800	900
`font-transform`	capitalize	uppercase	lowercase	none	
`font-variant`	none	small-caps			
`text-decoration`	underline	overline	line-through	none	

Tableau 1.

Parmi les autres propriétés des caractères qu'il peut être utile de choisir, on trouve aussi :

- `line-height` : *définit l'interligne*. Il est préférable d'utiliser un multiple (> 1, pas forcément entier) de `em`, ou un pourcentage ($> 100\%$).
- `word-spacing` : *fixe l'espacement des mots*.
- `letter-spacing` : *définit l'espacement des caractères (ou crénage).*
- `text-indent` : *crée une indentation de la 1ère ligne d'un paragraphe.*
- `text-align` : *détermine l'alignement du texte*. Les valeurs possibles sont : `left` (gauche), `right` (droite), `center` (centré) et `justify` (justifié). En fait, c'est un paramètre qui s'applique aux blocs.

6.4 Couleurs des caractères ou du fond du texte

On peut définir la couleur des caractères, la couleur du fond de toute la page, d'un bloc ou d'une portion de texte et la couleur des bordures.

La couleur du texte d'un élément est définie par le paramètre `color`, et celle du fond par `background-color` (ce paramètre s'applique donc aussi bien aux éléments de type « en-ligne» qu'aux éléments de type « bloc »).

On peut spécifier une couleur par son nom anglais (`blue`, `red`, ...) ou par la fonction RGB. On écrit `rgb(r,v,b)` ou `rgb(r%,v%,b%)` où `r`, `v` et `b` (entiers compris entre 0 et 255) représentent respectivement l'intensité lumineuse du rouge, du vert et du bleu. `r%`, `v%` et `b%` représentent la même chose mais en pourcentages.

7 Blocs, paragraphes, images, ...

Toutes les balises de type « bloc » sont vues et interprétées par les navigateurs comme des boites. Leurs principales caractéristiques sont résumées sur le schéma ci-dessous :

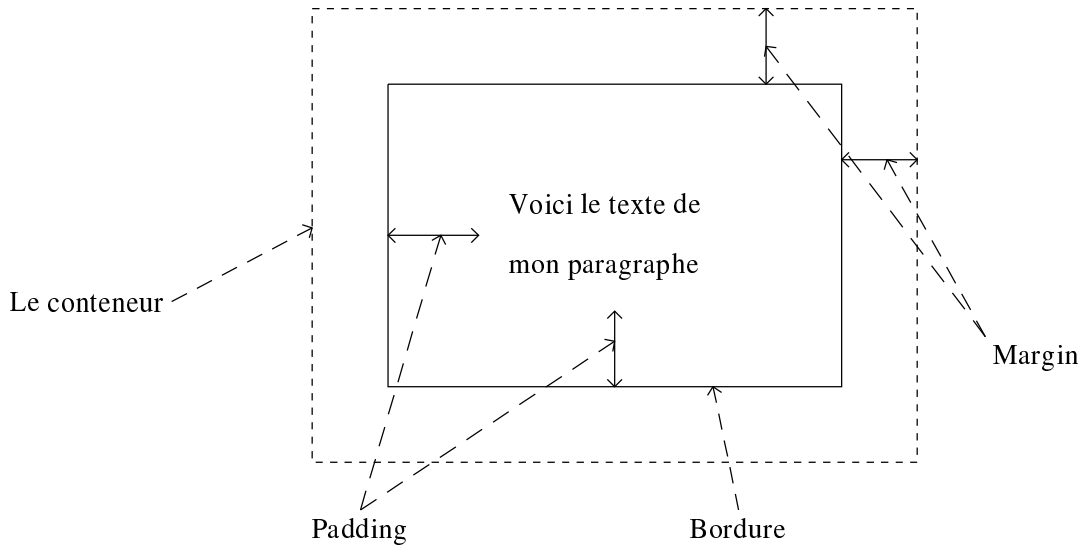

Figure 1.

`padding` est l'espace à l'intérieur de la bordure (que cette dernière soit visible ou pas). Il permet d'espacer le contenu de la boîte de la bordure. `padding` *possède la couleur du fond de l'élément concerné.*

`margin` est l'espace situé à l'extérieur de la bordure. Il permet d'espacer la bordure des éléments à l'extérieur de la boîte. `margin` possède la couleur du conteneur. `margin` est important pour le positionnement des blocs les uns par rapport aux autres[4], `padding` n'intervient pas à ce niveau.

L'encombrement total de la « boite » est donc très différent de la seule largeur du paragraphe (sur l'exemple).

Les marges extérieures sont spécifiées par 4 paramètres, ce qui permet d'avoir 4 valeurs différentes `margin-top` (en haut), `margin-right` (droite), `margin-bottom` (en bas) et `margin-left` (à gauche). Mais on peut grouper ces valeurs grâce à `margin`. Si on fournit une seule valeur, elle s'applique aux 4 ; si on fournit 4 valeurs, elles s'appliquent dans l'ordre à `top`, `right`, `bottom` et `left`.

Les valeurs sont fournies soit en pixels (`px`), soit en pourcentage qui s'entend par rapport à la largeur de l'élément parent. Tout se passe de la même façon pour les marges intérieures (`padding`, `padding-top`, `padding-right`, `padding-bottom` et `padding-left`).

```
{
    margin-top: 100px;
    margin-bottom: 100px;
    margin-right: 50px;
    margin-left: 50px;
}

{
    margin: 100px 75px 100px 75px;


{
    margin: 25px;
 }
```

7.1 Bordure d'un bloc

Une bordure met en jeu trois attributs :

- `border-style` : *le style de la bordure*. Valeurs possibles : `none` (pas de bordure), `dotted` (pointillé), `dashed` (tirets), `solid` (trait plein), `double` (double trait), `groove` (baguette en creux), `ridge` (baguette en relief), `inset` (enfoncement), `outset` (relief) ;

- `border-width` : *l'épaisseur de la bordure*. On la spécifie en principe en pixels (`px`). 1 px est très fin, 5 px est déjà assez épais ; on peut aussi utiliser `thin`, `medium` et `thick`.

- `border-color` : *la couleur de la bordure*. On spécifie les valeurs comme vu précédemment pour le texte et le fond. La valeur par défaut est la valeur de `color` pour l'élément, donc, généralement, la couleur du texte contenu.

On peut rassembler les trois spécifications : `{border: 5px solid red;}`.

Il faut toujours spécifier au moins le type, sinon la bordure n'apparaît pas car la valeur par défaut est `none`. On peut définir chaque segment `top`, `right`, `bottom` et `left` séparément.

4. Cf. la section 8 sur la « fusion des marges ».

7.2 Largeur d'un bloc

Un attribut important d'un élément bloc est sa largeur `width`. Il y a aussi la hauteur `height`, mais on la laisse presque toujours se définir automatiquement.

`width` se spécifie soit par nombre et unité (le plus souvent en `px`), soit en pourcentage (par rapport à la largeur de l'élément parent).

Si le contenu est trop grand par rapport aux dimensions de son conteneur, le comportement est régi par la propriété `overflow` du conteneur . Les valeurs possibles sont : `visible` (le contenu dépasse le cadre), `hidden` (ce qui est en trop est coupé), `scroll` (il apparaît des barres de défilement) et `auto` : seule(s) apparaît la (ou les) barre(s) de défilement nécessaire(s).

Avertissement. Contrairement à ce que cette présentation peut laisser penser, la largeur totale du conteneur n'est pas égale à `width` car si `padding` est inclus dans cette dimension, `margin` ne l'est pas. La largeur totale est donc égale[5] à : 2 `margin` + `width`.

7.3 Bloc image

Pour une image, les paramètres `width` et `height` sont généralement spécifiés comme attributs dans la balise pour que le navigateur dispose le plus vite possible des dimensions de la zone à réserver pour l'image. On place plutôt dans la feuille de style les paramètres d'alignement par rapport au texte qui se trouve dans le même conteneur.

On utilise le paramètre `vertical-align` dont les valeurs possibles sont :

- `bseline` : le bas de l'image est aligné sur le bas du conteneur.
- `middle` : le milieu de l'image est aligné sur le milieu du conteneur.
- `sub` ou `super` : l'image est placée respectivement en indice ou en exposant par rapport au conteneur.
- `text-top` : le haut de l'image est aligné avec le haut du conteneur.
- `text-bottom` : le bas de l'image est aligné avec le bas du conteneur.
- `top` : le haut de l'image est aligné avec le le haut de l'élément le plus haut de la ligne.
- `bottom` : le bas de l'image est aligné avec le bas de l'élément le plus bas de la ligne.

Remarque : *tous les paramètres abordés dans cette section ne se limitent pas aux images mais à toutes les balises de type « bloc ».*

7.4 Arrière-plan d'un bloc

La propriété `background` regroupe dans l'ordre les propriétés suivantes :

- `background-color` : *la couleur de fond*. Les valeurs possibles sont : `transparent` ou une couleur spécifiée.
- `background-image` : *l'image de fond*. Les valeurs possibles sont : `inherit`, `none` ou `url` (pour faire référence à un fichier).

5. Attention aussi au phénomène de « fusion des marges ».

– `background-position` : *la position de l'image de fond*. Les deux valeurs (x et y) sont comptées à partir du coin supérieur gauche de l'élément. On peut exprimer les différentes valeurs en pourcentages ou valeurs absolues (en pixels par exemples).

– `background-repeat` : *comment l'image est traitée si elle est plus petite que l'élément*. Les valeurs possibles sont : `repeat` (l'image est dupliquée dans les deux directions, haut et bas), `repeat-x` (l'image est dupliquée horizontalement), `repeat-y` (l'image est dupliquée verticalement), `no-repeat` (l'image n'est pas dupliquée et apparait en un seul exemplaire).

– `background-attachment` : *le fond défile-t-il ou reste-t-il fixe lorsque le contenu défile*. Les valeurs possibles sont : `scroll` (le contenu défile ; c'est le réglage par défaut) et `fixed` (le contenu ne défile pas). Cette dernière valeur n'est possible que si l'élément a un positionnement absolu (voir la section 8).

7.5 Les blocs listes

Un grand nombre de propriétés CSS sont consacrées à la mise en forme des listes. Pour l'essentiel :

– `list-style-type` : *choix du type de liste et du caractère utilisé*. Les valeurs possibles sont : `disc` (•), `circle` (◦) et `square` (■) pour `<ul>`, `decimal` (1, 2, 3, ...), `lower-alpha` (a, b, c, ...), ...pour `<ol>`.

  La valeur `none` s'applique aux deux sortes de listes, elle supprime puces et numérotation, *ce qui est très utile lorsqu'on crée un menu de navigation sous forme de liste*.

– list-style-image permet de choisir une image comme puce.

```
ul {list-style-image: url("./pucespeciale.png");}
```

Exemple.

```
ul {
   list-style-type: none;
   padding: 0px;
   margin: 0px;
   }
ul li {
    background-image: url("sqpurple.gif");
    background-repeat: no-repeat;
    background-position: 0px 5px;
    padding-left: 14px;
 }
```

8 Positionnement des éléments sur une page

Les informations qui constituent une page forment un « flot » constitué des conteneurs de type « bloc » (au fur et à mesure qu'ils sont lus par le navigateur). *Lorsqu'on ne spécifie aucun positionnement, les conteneurs successifs de type « bloc » se superposent, les uns au dessus des autres*.

Au sein d'un conteneur, *les éléments « en-ligne » successifs sont positionnés de la gauche vers la droite sur une ligne*. Une nouvelle ligne est commencée lorsque la largeur du bloc qui les contient est dépassée.

Deux conteneurs sont séparés non pas de la valeur de la somme de leurs marges mais par la plus grande des deux ; cette propriété s'appelle la *fusion des marges*.

Ce sujet est très important et pas toujours facile à comprendre. La lecture des ces trois articles est recommandée :

- *Initiation au positionnement CSS : 1. flux et position relative* – https://openweb.eu.org/articles/initiation_flux/ ;
- *Initiation au positionnement CSS : 2. position float* – https://openweb.eu.org/articles/initiation_float/ ;
- *Initiation au positionnement CSS : 3. position absolue et fixe* – https://openweb.eu.org/articles/initiation_absolue

8.1 Propriété `display`

Il est parfois utile de transformer un élément « en-ligne »en élément « bloc »(ou de faire le contraire). Cette transformation est facile à réalise grâce à au paramètre `display` :

```
display: block; /* transforme un élément de type en-ligne en un élément de type bloc */
display: inline; /* transforme un élément de type bloc en un élément de type en-ligne */
```

Deux éléments d'une liste de style `li {display: inline;}` seront alors placés l'un à la suite de l'autre (au lieu d'être l'un au dessus de l'autre) tandis que deux parties de texte contenues dans deux balises `<span>` de style

```
span {display: block;}
```

seront superposées.

Remarque : `display` permet juste de modifier la façon dont les éléments (blocs ou en-ligne) sont positionnés sur la page. Leurs caractéristiques restent inchangées. Par exemple, il reste impossible de placer une balise de type « bloc » dans une balise de type « en-ligne » même si cette dernière possède la propriété `display: block;`.

8.2 Positionnement des éléments sur la page

Le paramètre `position` permet de modifier le placement par défaut des éléments sur la page. Ce paramètre nécessite l'utilisation conjointe des paramètres `top`, `left`, `bottom` et `right` pour préciser la nouvelle position de l'élément.

Remarque : Les valeurs de `top`, `left`, `bottom` et `right` peuvent être indiquées en pixels ou pourcentages. Elles peuvent être positives ou négatives (on peut alors faire sortir l'élément de l'écran).

- Si le paramètre `position` prend la valeur `relative`, `top`, `left`, `bottom` et `right` *indiquent le décalage entre la position voulue de l'élément et la position qu'il aurait normalement occupée à l'écran.* La place normale reste vide, on ne la récupère pas.
- Si deux éléments se recouvrent en partie, la propriété `z-index` permet de déterminer lequel des deux sera situé au premier plan : c'est celui ayant la plus grande valeur de `z-index`.
- Si le paramètre `position` prend la valeur `absolute`, `top`, `left`, `bottom` et `right` indiquent la position de l'élément depuis le premier élément non statique.
- Si le paramètre `position` prend la valeur `fixed`, l'élément est positionné à l'endroit indiqué de l'écran et ne bouge pas. Si nécessaire, des barres de défilement apparaissent à l'écran afin que tous les éléments qui suivent le « flot » naturel de la page puissent être visualisés (pour ces éléments, celui qui est fixé à l'écran n'existe pas).

Cette propriété peut être très utile pour conserver dans un coin de l'écran un texte (ou un menu de navigation).

```
p.pos_fixe { /* concerne paragraphe de classe pos_fixe */
    position : fixed; /* diot être fixé à l'écran */
    top: 30px; /* à 30 pixels du haut */
    right: 5px; /* et à 5 pixels de la droite */
    }
```

8.3 Positionnement flottant

Il arrive très souvent que l'on souhaite placer à l'écran une image et du texte de telle sorte que ce dernier suive les contours de l'image. Le paramètre `float` permet de retirer du « flot » normal d'affichage un élément et de le faire « flotter » à l'écran. Ce paramètre peut prendre les valeurs `right` ou `left`.

```
img {float: right; /* on fait flotter les images à droite*/}

.vignettes {
          float: left; /* les blocs appartenant à la classe vignettes flottent à gauche */
          width: 110px;
          height: 90px;
          margin: 5px;
      }
```