CSS 2.0

Cascading Stylesheets

Bruxelles Formation | CEPEGRA | Olivier Céréssia | 2013

webdesigner.cepegra-labs.be

PRÉAMBULE

Nous avons vu ensemble l'HTML4 en premier lieu.

Maintenant, avec les feuilles de styles, nous allons séparer le contenu de sa mise en forme.

C'est très pratique et les feuilles de styles vont vous ouvrir des portes au niveau de la mise en forme que vous n'auriez même pas pu imaginer en restant enfermés dans les règles strictes du vieux HTML4.

Nous commencerons par une approche des CSS dans leur version 2.0 car elle a servi de base pour la version suivante. Nous embrayerons ensuite rapidement vers les CSS dans leur version 3.0 qui est toujours en cours de développement.

À ce stade-ci il est important de se rappeler une chose : toutes les règles que nous allons utiliser ici ne sont pas forcément interprétées de la même façon d'un navigateur à l'autre. Une vérification régulière est donc fortement conseillée.

TABLE DES MATIÈRES

POURQUOI LES CSS ?

• Les avantages des CSS

7 termes clés décrivent parfaitement les feuilles de styles :

Universalité : on conserve la structure logique, hiérarchique et sémantique du document quelle que soit la plateforme.

Productivité : du fait de la séparation du contenu et de la mise en forme, il est plus facile d'effectuer des opérations de création et de maintenance des pages

Légèreté des pages : le faible poids entraine une rapidité accrue (logique ☺)

Accessibilité : pour les personnes souffrant d'un handicap, pour les robots et les nouveaux dispositifs d'accès, la séparation contenu/mise en forme est capitale.

Créativité : Des dizaines et des dizaines de nouvelles règles vont vous offrir des possibilités presque infinies de mise en page et de design.

Compatibilité : la concision et la standardisation des règles offre une plus grande compatibilité cross-browser mais ce n'est toujours pas parfait.

Simplicité : tout semble plus cohérent, plus logique. Les nouvelles règles sont exploitées rapidement et facilement.

• Les faiblesses des CSS

Parce que rien n'est toujours tout rose, CSS a aussi quelques défauts.

L'un des principaux défauts des CSS est qu'ils sont en constante évolution. Nous avons en effet connu 3 version en un temps assez court : CSS1, CSS2 et maintenant CSS3 (toujours en cours de développement).

Chacune de ces versions apporte son lot de commandes et de nouveautés syntaxiques.
Prudence, tous les navigateurs ne sont pas aptes à interpréter les différentes versions. Il y a donc des problèmes de compatibilité.

Comme toujours, de nombreux tests sont donc nécessaires !

Principe général

Le principe est assez simple :

1. On crée des pages HTML valides en se servant exclusivement de balises sémantiques (h1, p, strong, nav, header, footer, …) dont on veille à conserver le sens et la hiérarchie.
2. On crée une série de styles CSS afin de « re-styler » la manière dont le parseur affiche ces balises sémantiques.

Principes secondaires

Les principes secondaires qui suivent sont à comprendre parfaitement :

1. Les déclarations de styles CSS comprennent un sélecteur (qui désigne ce qui doit être re-stylé) et une déclaration de propriétés (qui décrit comment cela doit être re-stylé).
2. Pour que les styles CSS s'appliquent aux pages HTML, il faut choisir une **méthode d'intégration**, plus ou moins centralisée (valable pour une partie de la page, la page entière, plusieurs pages ou toutes les pages).
3. Pour utiliser certains sélecteurs adaptés aux exceptions et cas particuliers, il est nécessaire de compléter les pages HTML avec des balises et/ou attributs spéciaux. Ceci permet l'**application** correcte des styles.
4. L'interprétation des CSS par le parseur se base sur des règles de **gestion des priorités** entre les styles.

ÉTAPES PRÉLIMINAIRES

Dans un monde idéal, voici les quelques étapes préparatoires que vous devriez être amenés à suivre pour créer un site ou une page.

• Préparation de la maquette

1. On crée d'abord un layout le plus homogène et cohérent possible.
2. On structure la page en plusieurs zones distinctes (titre, navigation principale, navigation secondaire, contenu, pied de page, …)
3. On maintient un nombre raisonnable de styles.

• Préparation de la page HTML

Les pages HTML auxquelles s'appliquent les CSS doivent se baser sur un codage exclusivement sémantique.

Vous veillerez donc à n'utiliser que des balises porteuses de sens (h1, p, strong, em, …).

On garde toujours en tête les règles élémentaires de conception d'une page classique et on veille à ce que sa structure soit valide :

- Respect de la hiérarchie des titres (h1 est plus fort que h2 qui est plus fort que h3 etc.)

- Respect de la syntaxe d'écriture et d'imbrication des balises (…)

- Fermeture des balises (…)

Ça peut paraître un peu ringard de rappeler ces 3 règles de bases, mais croyez-moi vous ferez des erreurs de ce style encore pendant un bon moment (on parie ? ^^)

Les techniques héritées du passé (tableaux de mise en forme, pixel transparent, texte sous forme d'image, …) doivent être évitées autant que possible.

On peut encore utiliser les tableaux mais uniquement pour présenter des données tabulaires (ex : un rapport annuel, des statistiques, etc.).

• Application de styles

3 techniques différentes vont vous permettre d'appliquer un style à un élément xhtml. Chacune est utilisée dans un cas précis :

- **CSS en ligne** : cette technique consiste à placer un attribut style directement à l'intérieur d'une balise. On utilise cette technique uniquement pour la réalisation de newsletters.

```
<p style="color:red">Un paragraphe rouge</p>
```

- **CSS dans la balise <head>** : cette technique consiste à placer toutes les propriétés CSS entre les balises <style></style> directement dans le <head> de votre document HTML.

  ```
  <style>p {color:red}</style>
  ```

- **Fichier CSS lié dans la balise <head>** : cette technique consiste à placer un « lien » vers une feuille de style dans le <head> de votre document XHTML.

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

  Vous veillerez à faire attention au chemin relatif indiqué dans l'attribut href afin que le fichier CSS soit bien placé par rapport au fichier .html dans lequel il a été lié.

  Les 2 autres attributs, rel et type sont obligatoires et invariables.

  Vous pouvez lier autant de fichiers CSS que voulu, pour éventuellement séparer certains styles d'autres.

  C'est évidemment cette façon d'ajouter des styles qui sera toujours employée pour la création de site web tandis qu'employer des CSS « en ligne » est encore utilisé pour les newsletters.

LA DÉCLARATION DE STYLE

• Syntaxe générale

La syntaxe de la déclaration de style se présente comme ceci :

```
sélecteur { déclaration de propriété }
```

ou plus précisément :

```
sélecteur {
     propriété1:valeur;
     propriété2:valeur;
     …
}
```

Attention aux détails

- La déclaration de propriétés est entourée par des accolades { } et non par des crochets [ ] ou des parenthèses ( ).

- Le couple « propriété de style : valeur ; » est séparé par un double point :

- La valeur ne doit pas être entourée par des guillemets " "

- Chaque couple « propriété de style : valeur ; » est séparé par un point-virgule ; Seule la dernière déclaration ne nécessite pas de point-virgule

- Il n'y a pas de limite de nombre de règles CSS dans une déclaration.

- Les espaces dans une déclaration de propriétés ne sont pas obligatoires mais aident fortement à la lisibilité du code source.

Exemple :

Déclaration de style redéfinissant la balise <h1> :

```
h1 {
      font-family: Arial;
      font-size: 24pt;
      font-style: italic;
      color: #ff0000;
}
```

À travers cette déclaration, on spécifie la police, la taille, l'inclinaison et la couleur du texte balisé par <h1>.

Dans cet exemple, <h1> est donc re-stylé pour s'afficher en Arial, 24 points, italique et rouge.

Plutôt simple non ?

Quelques commentaires s'imposent néanmoins…

Insensibilité à la casse

Les CSS ne sont pas sensibles à la casse que ce soit pour l'écriture des sélecteurs, propriétés ou valeurs.
Cependant les éléments qui ne sont pas propres aux CSS, comme les noms de police ou les URLs, peuvent être sensibles à la casse.
Par exemple, pour certains systèmes d'exploitation (tel que Unix), 'Arial' n'est pas égal à 'arial', de même 'IMAGE.gif' n'est pas égal à 'image.gif' !

Dans le même esprit que celui du XHTML, certains codeurs ont pris l'habitude de taper toutes leurs commandes CSS en minuscule.

Insensibilité aux sauts de ligne

La déclaration de style peut s'écrire aussi bien sur une ligne :

```
h3 { font-family: Arial;font-style: italic; color: #00ff00}
```

… que sur plusieurs lignes (plus lisible):

```
h3 {
font-family: Arial;
font-style: italic;
color: #00ff00
}
```

• Les commentaires

Il est également possible de commenter ses déclarations de style avec la syntaxe :

```
/* Ceci est un commentaire */
```

Pour rappel, tout ce qui se trouve entre commentaires ne sera jamais visible par les utilisateurs. En revanche, il est important de commenter vos pages pour vous-même :

- Si vous travaillez sur vos pages à des périodes assez espacées
- Si vous travaillez à plusieurs personnes sur la même page
- Par soucis de clarté et de structure

LA DÉCLARATION DE STYLE

• Mission du sélecteur

Dans une déclaration de style, le sélecteur désigne ce qui doit être re-stylé dans la page HTML. Il doit donc pouvoir apporter le degré de précision adapté à chaque situation.

C'est la raison pour laquelle il existe plusieurs types de sélecteurs répondants chacun à des besoins particuliers.

Chaque type de sélecteur est reconnaissable syntaxiquement et c'est justement par cette syntaxe qu'il gagne en précision et en concision.

Table de compatibilité

À partir d'ici, vous verrez souvent apparaître sous certains éléments un tableau de compatibilité qui reprendra le support de cet élément dans les différents navigateurs.

Ce tableau est là pour vous aider à savoir ce que vous pouvez utiliser et dans quel navigateur et ce qu'il vaut mieux éviter (ou contourner) dans certains cas.

On y détaille 3 versions d'Internet Explorer car il est le navigateur le plus difficile à contenter.

signifie que tout est ok, signifie que tout n'est pas parfait, signifie que cela ne fonctionne pas

• Les différents types de sélecteurs

Voici les différents sélecteurs disponibles jusqu'à la version 2.0 de CSS (de nouveaux sélecteurs arrivent avec CSS 3.0) :

1. Le sélecteur **universel** : *****
2. Le sélecteur de **type** : **X**
3. Le sélecteur **d'attribut** : balise**[attribut]**
4. Le sélecteur de **classe** :**.**.maclasse
5. Le sélecteur d'**ID** : **#**monid
6. Le sélecteur **descendant** : X Y
7. Le sélecteur **adjacent** : X **+** Y
8. Le sélecteur **d'enfant** : X **>** Y
9. Les **pseudo-éléments** : **:before** et **:after**
10. Les **pseudo-éléments** : **:first-line** et **:first-letter**
11. Les **pseudo-classes** : **:link**, **:visited**, **:hover** et **:active**
12. Le **pseudo-classes** : **:first-child** et **:focus**

1. Le sélecteur universel

Le sélecteur universel sélectionne indifféremment toutes les balises d'un document. C'est le sélecteur le plus puissant mais aussi le moins précis.

Il se présente comme suit :

```
* { déclaration de propriétés }
```

Exemple :

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

Les marges internes et externes par défaut de toutes(*) les balises du document sont supprimées.

2. Le sélecteur de type (de balise)

Toutes les balises sémantiques peuvent servir de sélecteur, absolument TOUTES !

<html>, <body>, <p>, <a>, <input>, <table>, <tr>, <td>, , , <ul>, <ol>, <li>, <img>, etc.

Exemple :

```
p {
      line-height: 2em;
      border-left-width: thick;
      padding-left: 25px;
}
```

Dans cet exemple, les balises p du document sont re-stylées avec un interlignage correspondant à 2 fois la taille du texte par défaut. De plus, un filet épais apparaît sur le côté gauche du paragraphe, à une distance de 25px.

3. Le sélecteur d'attribut

Les sélecteurs d'attributs permettent d'appliquer un style en fonction de la valeur de l'attribut de la balise HTML. Ils prennent la forme suivante :

```
balise [attribut="valeur"] { … }
```

Exemple :

```
hr [size="3"] { width: 50%; }
```

Le style (largeur fixée à 50%) ne sera appliqué qu'aux filets horizontaux dont l'épaisseur est de 3.

4. Le sélecteur de classe

Les classes permettent de cibler l'application d'un style en lui attribuant un nom. Elles permettent de gérer les cas particuliers et les exceptions.

```
.nom_de_la_classe { … }
```

Elles sont ensuite appelées dans le document avec l'attribut class :

```
<balise class="nom_de_la_classe">…</balise>
```

Le choix du nom de la classe

L'application du style n'est plus automatique. Seules les balises appelant la classe appliqueront son style.

Le choix du nom pour les classes est libre. Toutefois, les noms des classes ne peuvent contenir que les lettres de a-z ou de A-Z, les chiffres de 0-9, le trait d'union (-) et le caractère souligné (_). Les noms ne peuvent pas commencer par un chiffre ou un tiret.

Exemple :

Je souhaite faire ressortir ce qui est très important dans un document. Je crée la classe essentiel à laquelle j'associe un style spécifique (gras et bleu)…

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

Dans le document Html, il suffit d'appeler la classe essentiel quand cela s'avère nécessaire :

```
<h1 class="essentiel"> ... blabla ... </h1>
<p> ... blabla ... </p>
<p class="essentiel"> ... blabla ... <p>
<table>
  <tr>
   <td class="essentiel"> ... </td>
  </tr>
</table>
```

Pour limiter la portée d'une classe, on peut lors de sa définition la restreindre à n'être appelée que par une balise particulière :

```
baliseX.nom_de_la_classe { … }
```

Dans le document, la classe ne peut être appelée que dans la balise spécifiée.

```
<baliseX class="nom_de_la_classe">…</baliseX>
```

Si la classe est appelée dans une autre balise, le parseur n'applique pas la classe.

```
<baliseY class="nom_de_la_classe">…</baliseY>
```

Exemple :

On crée la classe super associée à la balise h1…

```
h1.super { font-size: 12pt; font-weight: bold; }
```

On peut ensuite appeler la classe dans la page Web…

```
<h1 class="super"> …blabla… </h1>
```

Mais, attention :

```
<h2 class="super"> …blabla… </h2>
```

Cette déclaration sera sans effet car la classe super n'a pas été associée à h2 dans la déclaration de style.

Avec cette écriture plus restrictive des classes, car liée à une balise spécifique, rien n'empêche d'utiliser plusieurs fois le même nom de classe avec des balises différentes.

```
h1.super { font-size: 12pt; font-weight: bold; }
h2.super { font-size: 10pt; font-weight: normal; }
h3.super { font-size: 8pt; font-style: italic; }
```

Il s'agit bien ici de trois sélecteurs différents.

5. Le sélecteur d'ID

Comparables aux classes, les ID permettent de cibler l'application d'un style en lui attribuant un nom. Elles permettent donc aussi de gérer les cas particuliers et les exceptions. Les ID se présentent comme suit…

```
#nom_de_l'ID { … }
```

Le choix du nom de l'ID doit respecter les mêmes restrictions que pour les classes.

Les ID sont ensuite appelées dans le document avec l'attribut id :

```
<balise id="nom_de_l'ID">…</balise>
```

Tout comme pour les classes, pour limiter sa portée, une ID peut voir son appel restreint à une balise particulière…

```
baliseX#nom_de_l'ID { … }
```

Dans le document, la classe ne peut être appelée que dans la balise spécifiée.

```
<baliseX id="nom_de_l'ID">…</baliseX>
```

A première vue, Classes et ID remplissent la même fonction.

Mais attention, une ID ne peut être appelée qu'une seule fois par document !

Prenons la déclaration suivante : `#essentiel { ... }`

Dans le document, l'appel `<p id="essentiel">` est correct. Mais si on rencontre un peu plus loin, dans le même document, `<h1 id="essentiel">`… le document n'est plus valide !

Du fait de cette contrainte, on voit mal pourquoi ne pas abandonner les ID au profit des Classes qui sont beaucoup plus souples. Nous verrons que cette contrainte peut être un avantage lorsque l'on combine les ID avec certaines formes syntaxiques de sélecteur. Affaire à suivre, donc…

6. Le sélecteur descendant

Le sélecteur de descendance permet d'appliquer un style à la balise fille d'un parent (imbrication). On sépare chaque élément par un espace blanc. Cela se présente comme ceci :

```
sélecteurA sélecteurB { … }
```

Exemple :

```
p strong { border: solid 1px #999; }
```

Appliquera une bordure dans le document à...

```
<p>…<strong> Cible </strong>…</p>
```

mais pas à...

```
<h1>...<strong>blabla</strong>...</h1>
```

Le sélecteur de descendance associé avec des ID offre beaucoup de précision et de puissance. En effet, lorsque la page est découpée en zones associées à des ID uniques (#en-tete, #menu, #contenu, #footer…), on peut utiliser dans chacune des zones les mêmes balises sémantiques ou les mêmes classes sans qu'elles n'aient le même style.

Avec quelques zones et un nombre réduit de balises sémantiques, on peut varier facilement les styles!

Exemples :

```
#en-tete p { font-size: 12pt; font-weight: bold; }
#en-tete a { color: #00f; text-decoration: none; }
#contenu p { font-size: 10pt; line-height: 1.4em; }
#contenu a { color: #000; font-style: italic; }
```

7. Le sélecteur adjacent

Le sélecteur adjacent permet d'appliquer un style à la balise suivant immédiatement une autre balise. Il se présente comme suit :

```
sélecteur A + sélecteurB { … }
```

Exemple :

```
h1 + p { font-weight: bold; }
```

Dans ce document HTML :

```
<h1>… blabla …</h1>
<p>Cible</p>
<p>… blabla …</p>
```

Voilà le résultat obtenu :

… blabla …

Cible

… blabla …

Il faut donc lire la déclaration comme suit : « à partir d'un <h1>, la première balise <p> aura le style suivant ». Et si vous voulez cibler le 2e paragraphe, rien de plus simple :

```
h1 + p + p {
      color:blue;
}
```

8. Le sélecteur d'enfant

Le sélecteur d'enfant permet de sélectionner l'enfant direct d'une balise. Il se présente comme suit :

```
sélecteur A > sélecteurB { … }
```

Exemple :

```
p > strong { font-weight:normal; }
```

Dans ce document HTML :

```
<p><strong>sélecteur <div><strong>css</strong></div> d'enfant</strong></p>
```

Voilà le résultat obtenu :

sélecteur

css
d'enfant

Le sélecteur n'agit donc que sur le tout premier élément et pas sur tous ceux qui sont inclus en lui comme c'était le cas avec le sélecteur descendant.

9. Les pseudo-éléments :before et :after

Les pseudo-éléments :before et :after permettent de placer du contenu avant ou après un élément HTML. Ils se présentent comme suit :

```
Sélecteur:before {content : " and some text after."}
Sélecteur:after{content: url(../pix/logo_ppk.gif)}
```

Comme vous pouvez le voir ci-dessus, vous avez la possibilité de placer du contenu textuel ou image avant ou après un élément HTML grâce à la propriété content.

Exemple :

```
p.test:after {
      font-style: italic;
      content: " gosh!";
}
```

Dans ce document HTML :

```
<p class="test">Oh my</p>
```

Voilà le résultat obtenu :

Oh my *gosh!*

10. Les pseudo-éléments :first-line et :first-letter

Les pseudo-éléments :first-line et :first-letter permettent de styler la première ligne ou la première lettre d'un élément HTML. Ils se présentent comme suit :

```
Sélecteur :first-line {…}
Sélecteur :first-letter {…}
```

:first-line est déterminé par le navigateur, càd qu'il s'agira toujours de la première ligne visible à l'écran et non de la première ligne se terminant par un point.

Exemple :

```
p:first-line {
      color:salmon;
}
p:first-letter {
      font-size:30px;
}
```

Dans ce document HTML :

```
<p>Il était sur le dos, un dos aussi dur qu'une carapace, et, en relevant
un peu la tête, il vit, bombé, brun, cloisonné par des arceaux plus
rigides, son abdomen sur le haut duquel la couverture, prête à glisser tout
à fait, ne tenait plus qu'à peine.</p>
```

Voilà le résultat obtenu :

Il était sur le dos, un dos aussi dur qu'une carapace,
et, en relevant un peu la tête, il vit, bombé, brun,
cloisonné par des arceaux plus rigides, son abdomen
sur le haut duquel la couverture, prête à glisser tout à
fait, ne tenait plus qu'à peine.

11. Les pseudo-classes :link, :visited, :hover et :active

Les pseudo-classes précisent la portée (l'état) de la balise a :

```
a:link /*Liens non-visités (par défaut)*/
a:visited /*Liens dont la cible a été visitée*/
a:hover /*Liens survolés avec la souris*/
a:active /*Liens cliqués avec la souris (actif) */
```

Attention, si plusieurs de ces pseudo-classes sont utilisées dans une déclaration de style, l'ordre repris ci-dessus doit être respecté.

Exemple :

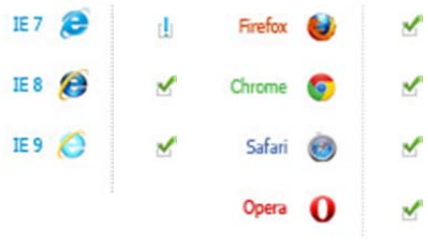
```
a { text-decoration:none; }
a:link { font-variant: small-caps; }
a:hover { background-color: #ff3300;}
```

Les liens (quel que soit leur état) ne seront pas soulignés. Les liens non-visités seront en petite majuscule. Les liens survolés par la souris auront un arrière-plan uni orangé (effet de « surlignage »).

Attention, certains navigateurs acceptent d'utiliser la pseudo-classe :hover en combinaison avec d'autres balises :

```
h1:hover {…} /*h1 survolé avec la souris*/
td:hover {…} /*Cellule survolée avec la souris*/
```

C'est typiquement le genre d'effets qu'on voit de plus en plus depuis l'apparition de CSS3, n'hésitez donc pas à en abuser !



12. Les pseudo-classes :focus et :first-child

Il existe encore deux autres pseudo-classes. La première, :focus permet d'appliquer un style à, par exemple, un champ de formulaire quant l'internaute y saisit des données.

Tandis que :first-child a une toute autre utilité : elle permet d'appliquer un style à la première balise fille d'une balise parente.

Exemple pour :focus :

```
input:focus { font-weight: bold; }
```

Appliqué à ce code HTML suivant :

```
<input type="text" />
```

Voici ce que ça donne quand on n'a pas encore cliqué sur le champ :

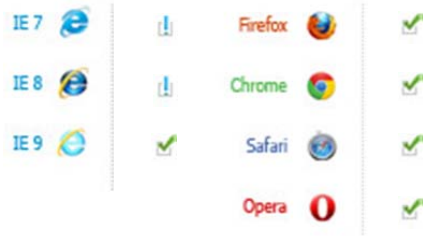
Et voici ce que ça donne quand on clique sur le champ :

Le champ se met en surbrillance (comportement par défaut au click) et le texte prend bien la graisse demandée dans notre déclaration CSS.

Exemple pour :first-child :

```
li:first-child { text-transform: capitalize; }
```

Le texte du premier item d'une liste sera en majuscule.



Attention !

Ces sélecteurs se ressemblent mais sont différents :

`ul.super` Classe super appelée dans la balise ul
`ul .super` Classe super fille de ul
`ul , .super` Classe super et ul définit de la même manière

• Le poids des sélecteurs

La règle de la cascade s'applique en CSS, càd que la dernière propriété qui a été énoncée a la priorité sur les autres qui précèdent.

Il y a néanmoins une exception qui tient en une règle de poids selon le sélecteur que vous aurez choisi. Voici, dans l'ordre les déclarations qui ont d'abord plus de poids et finalement celles qui en ont moins :

1) Poids « a » : règle CSS déclarée à l'aide de l'attribut HTML style= au sein de l'élément
2) Poids « b » : sélecteur d'identifiant (id)
3) Poids « c » : sélecteur de classe (.), d'attribut ([]) ou de pseudo-classe (:hover, :focus, :first-child)
4) Poids « d » : sélecteur d'élément (p, div, a, li, ul, …) ou de pseudo-élément (:first-letter, :before, :after, :first-line)
5) Poids nul : sélecteur universel (*), de parenté (>) ou d'adjacence (+).

!important

La déclaration !important a été introduite par CSS dans le but d'outrepasser volontairement la priorité conférée par défaut aux modes de déclaration que sont les feuilles de styles. Dans la pratique, une déclaration suivie du mot-clé !important devient préférentielle, quel que soit le poids du sélecteur qui l'accompagne : les styles marqués ainsi écrasent d'office tous les styles similaires antérieurs.

Par exemple :
```
a.error {color :red !important ;}
```

LA DÉCLARATION DE STYLE

• Déclaration de propriétés

Dans une déclaration de style, la déclaration de propriété détermine comment doit être re-stylée la partie du document circonscrite par le sélecteur.

Pour rappel, la déclaration de propriété { … } se compose d'un ou plusieurs couples « propriété: valeurs; ».

Il existe à l'heure actuelle une cinquantaine de propriétés typographiques et de composition ainsi qu'une gamme étendue de valeurs répondants chacune à des besoins particuliers.

Attention !

Attention aux similitudes/différences entre, d'une part les balises et attributs HTML et, d'autre part, les propriétés CSS.

```
<font size="4"> n'est pas … { font-size: 4px; }
<font color="#cccccc"> n'est pas … { color: #cccccc; }
```

Un temps d'adaptation est nécessaire pour ne pas les confondre et éviter les pièges.

• Les familles de propriétés

Les propriétés CSS peuvent être regroupées selon leur usage. On peut, par exemple, distinguer les catégories suivantes :

Fontes	Font-family, font-size, font-weight, …
Texte	Letter-spacing, word-spacing, text-decoration, line-height, …
Couleurs et images de fond	Color, background-color, background-image, background-repeat, …
Boîtes	Margin-top, padding-left, border, width, height, display, …
Listes	List-style-type, list-style-image, …
Positionnement	Position, left, top, z-index, …

Du fait de l'existence de ces familles, certains regroupement peuvent se faire au niveau des propriétés afin d'obtenir une syntaxe plus courte. Ainsi :

```
h1 {
font-weight: bold;
font-size: 12pt;
line-height: 14pt;
font-family: helvetica;
}
```

peut aussi s'écrire…

```
h1 { font: bold 12pt/14pt helvetica; }
```

Les types de valeurs

Selon les cas, les valeurs des propriétés peuvent être de différents types :

Appellations spécifiques

italic, underline, none, small, larger, thin, dotted, normal, left, … adaptées à chaque propriété.

Valeurs numériques

Valeurs exprimées dans l'une des unités de mesure disponible.

Couleurs

Les valeurs numériques

Les unités de mesure se classent en deux catégories :

Les unités « absolues » :

cm (centimètre)
mm (millimètre)
in (inch (pouce, soit 2,54 cm))
pt (point (1/72 de pouce))
pc (pica (12 points))

Les unités « relatives » :

em (hauteur par défaut de la fonte)
ex (hauteur par défaut du caractère x minuscule (utilisé pour les petites majuscules))
px (pixel)
% (pourcentage)

Remarques

Attention, il n'y a pas d'espace entre la valeur numérique et l'unité.

```
… { font-size: 12pt; }
```

Pour toutes les unités de mesures (sauf le pixel et le pourcentage), une valeur décimale est autorisée. Le séparateur est le point.

```
… { line-height: 1.6em; }
```

Pour la mesure des distances, les valeurs peuvent être négatives !

```
… { margin-left: -50px; }
```

Si la valeur numérique est nulle, il n'est pas nécessaire de préciser l'unité de mesure

```
… { padding-right: 0; }
```

Quelle unité choisir em ou px ?

Sur ce point, beaucoup de monde s'oppose mais pourtant une unité est peut-être un peu plus correcte que l'autre.

Par défaut, quand on commence à coder en CSS, on utilise plus facilement les Px car il s'agit d'une unité qu'on connait, avec laquelle on a l'habitude de travailler dans photoshop par exemple.

Une des grandes différences qui opposait les em ou px c'est que jusqu'à la version 8 d'Internet Explorer, sous Windows, il était impossible d'agrandir la taille de la font selon nos envies avec l'utilisation des px. Ce qui, au niveau accessibilité, est quand même fortement peu recommandé.

C'est pourquoi les em sont recommandés par le W3C.

Le principe de l'unité em est assez simple :

- Ne pas fixer de corps de base. Le corps de base étant déterminé par le visiteur (dans les préférences de son logiciel de navigation).
- Fixer tous les autres corps en valeurs proportionnelles au corps de base (de préférence en em). Ainsi, en assignant 1.5em à H2, les intertitres de niveau 2 auront une fois et demi le corps de base fixé par les préférences du visiteur. Si les préférences du visiteur sont réglées à 16 pixels dans son logiciel de navigation, on obtiendra la valeur de 24 pixels pour H2.

Le calcul des em à choisir est assez simple.

1em = la hauteur de font par défaut choisie par l'utilisateur

De cette manière, par exemple le code ci-dessous :

```
body{font-size :0.8em;}
```

… signifie qu'on prend 80% de la hauteur de la font par défaut de l'utilisateur et qu'on l'applique à tout le document.

Tandis que :

```
H2{font-size :1.5em;}
```

… signifie qu'on agrandit d'une demi-fois la taille du h2.

L'inconvénient des EM est peut-être aussi un peu dérangeant pour des graphistes travaillant au pixel près. Étant donné qu'on travaille avec une unité relative sur laquelle on n'a pas d'influence au niveau de son unité de base, on ne peut deviner à 100% à quoi ressemblera notre rendu sur chaque poste. Dès lors, il faut penser ses designs en leur offrant la possibilité de s'adapter selon les paramètres réglés par les utilisateurs.

Après, libre à vous d'utiliser les EM ou les PX, il n'y a pas de réelle erreur proprement-dite concernant les PX, ils sont juste un peu moins « accessibles » et « adaptatifs » que les EM.

Un convertisseur PX vers EM est disponible ici

Les couleurs

Les couleurs sont formulées comme en HTML :

#valeur hexadécimale RVB.

… { color: #3366ff; }

Ou représentées par leur nom en anglais.

Écriture courte

Si la couleur est exprimée par trois paires (c'est le cas de toutes les couleurs de la palette Web), on peut se contenter d'écrire un seul caractère de chaque paire.

#ffffff devient #fff
#000000 devient #000
#3366ff devient #36f

Les écritures suivantes sont également autorisées, et on verra en CSS3 qu'elles vont revenir au goût du jour car aux 3 paramètres de couleur que sont RGB viendra s'ajouter une couche supplémentaire qui pourra régler l'alpha (la transparence).

Mais en CSS2 on ne peut pas encore jouer sur l'alpha donc on se contente d'une écriture comme celle-ci :

```
… { color: rgb(255, 204, 0) ; }
… { color: rgb(100%, 80%, 0%) ; }
```

Valeurs multiples

Dans certains cas, 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; }
```

Nous reviendrons sur les différentes propriétés CSS2 qui permettent ce genre d'écriture quand nous les rencontrerons plus tard.

LA DÉCLARATION DE STYLE

• Index des propriétés et valeurs

Les propriétés de style CSS sont nombreuses et plus riches que les attributs HTML. Les spécifications officielles reprenant la liste de toutes les propriétés et leurs valeurs pour les versions 1 et 2.1 des CSS est disponible sur le site du W3C :

http://www.w3.org

Le présent document contient la quasi-totalité des propriétés existantes et pour chaque propriété, vous aurez un tableau présentant sa fonctionnalité et ses valeurs possible en plus d'un petit tableau de compatibilité inter-navigateurs.

Pour toutes les propriétés à venir, la valeur « inherit » est toujours présente. Elle prend la même valeur que son élément parent.

Nom-de-la-propriété	Valeur par défaut
Texte de description de la propriété, de ses différentes valeurs et de l'effet de ces différentes valeurs.	Autre valeur 1 Autre valeur 2 Autre valeur 3 etc.

• Les fontes

Font-family	Times New Roman
Définit la famille de la fonte : **police** (Arial), **suite de polices** ("Times New Roman", Times, serif) ou **famille de police** (serif | sans-serif | cursive | fantasy | monospace).	Verdana Georgia Times New Roman Courier New Courier Arial Tahoma Trebuchet MS Arial Black Palatino Linotype Book Antiqua Lucida Sans Unicode Lucida Console Comic Sans MS

Font-style	normal
Définit le style de la fonte.	Italic Oblique

Font-variant	normal
Définit la variante de la fonte.	small-caps

Font-weight	normal
Définit la graisse de la fonte.	bold bolder lighter 100>900

Font-size	medium
Définit la taille de la fonte : valeur spécifique en : **Les unités « absolues » :** **cm** (centimètre) **mm** (millimètre) **in** (inch (pouce, soit 2,54 cm)) **pt** (point (1/72 de pouce)) **pc** (pica (12 points)) **Les unités « relatives » :** **em** (hauteur par défaut de la fonte) **ex** (hauteur par défaut du caractère x minuscule (utilisé pour les petites majuscules)) **px** (pixel) **%** (pourcentage)	xx-small x-small small large x-large xx-large valeur numérique + unité de mesure

Line-height	normal
Définit la distance d'espacement entre deux lignes de texte.	Valeur Numérique + unité de mesure

Forme raccourcie

Vous pouvez utiliser la propriété font qui est la forme raccourcie pour les 6 propriétés vues ci-dessus. Mais attention, la propriété font est une des seules qui nécessite de faire attention à l'ordre des éléments quand vous les écrivez :

```
font:style variant weight size/line-height family;
```

• Le texte

Letter-spacing	normal
Définit la distance d'espacement entre deux caractères.	Valeur Numérique + unité de mesure

Word-spacing	normal
Définit la distance d'espacement entre deux mots.	Valeur Numérique + unité de mesure

Text-indent	normal
Définit l'indentation (retrait) de la première ligne.	Valeur Numérique + unité de mesure

Remarque

La plupart des valeurs pour les propriétés numériques supportent des valeurs négatives, ne l'oubliez jamais !

Text-transform	none
Définit la casse du texte : aucun effet (none), tout en majuscule (uppercase), tout en minuscule (lowercase), premières lettres de chaque mot en majuscule (capitalize)	Uppercase lowercase capitalize

Text-decoration	none
Définit un effet sur le texte : aucun effet (none), souligné (underline), barré (line-through), surligné (overline) ou clignotant (blink – cette dernière valeur ne fonctionne que sous Firefox).	Underline line-through overline blink

Text-align	left
Définit l'alignement du texte à l'intérieur de l'endroit où il se trouve.	Center right justify

Vertical-align	top
Définit l'alignement vertical d'un élément par rapport à l'environnement en ligne dans lequel il se trouve. De plus cet attribut s'applique presque exclusivement à l'élément display :table-cell.	Middle Bottom ..

Attention

Contrairement à ce qu'on avait rencontré avec les tableaux, centrer un élément verticalement en CSS est beaucoup plus compliqué qu'il n'y parait, même avec l'aide de cette propriété vertical-align. Nous y reviendrons plus tard quand nous aurons vu d'autres propriétés CSS pour étoffer les exemples.

White-space	normal
Définit le type d'espace blanc : normal, interprète les espaces multiples et les sauts de lignes (pre), interdit les sauts de ligne (nowrap).	Pre nowrap pre-wrap pre-line

• Les couleurs et images de fond

Color	Black ou #000

Définit la couleur d'un texte.	#RRVVBB
	rgb(R%,V%,B%)
	nom de la couleur en anglais

Background-color	transparent

Définit la couleur de l'arrière-plan.	#RRVVBB
	rgb(R%,V%,B%)
	nom de la couleur en anglais

Background-image	none
Définit l'image d'arrière-plan.	URL(adresse et chemin de l'image – en absolu ou relatif)

Background-repeat	repeat
Définit comment l'image d'arrière-plan se répète : horizontalement et verticalement (repeat), horizontalement (repeat-x), verticalement (repeat-y), pas du tout (no-repeat)	Repeat-x repeat-y no-repeat

Background-attachment	scroll
Définit si l'image d'arrière-plan reste fixe avec le défilement de l'ascenseur (fixed) ou pas (scroll).	fixed

Background-position	Top left
Définit la position de l'image d'arrière-plan par rapport au conteneur dans lequel elle se trouve. On peut l'aligner horizontalement : à gauche (left), au centre (center) et à droite (right). On peut aussi l'aligner verticalement : en haut (top), au milieu (center), en bas (bottom).	Horizontalement : left center right Verticalement : top center bottom

Forme raccourcie

Vous pouvez utiliser la propriété background qui est la forme raccourcie pour les 5 propriétés vues ci-dessus.

Il n'y a pas d'ordre particulier pour déclarer chacune des propriétés mais je vous recommande d'écrire vos propriétés comme ci-dessous.

De plus notez que quand vous précisez une position horizontale et une position verticale, il faut toujours commencer par la position horizontale.

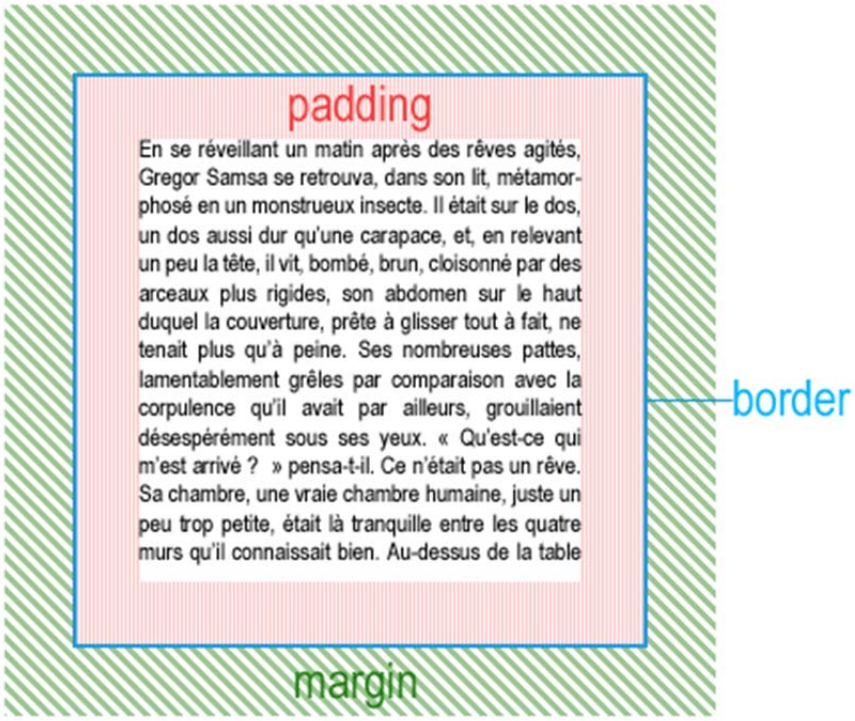
Dernière remarque, on peut associer une image et une couleur de fond, dans ce cas, la couleur de fond viendra se placer derrière l'image pour combler tous les endroits où l'image ne peut pas forcément aller.

```
background:url(dadjack.jpg) #0000FF no-repeat center 100px fixed;
```

• Les boîtes ou conteneurs

Une boite (ou un conteneur) se définit par différents paramètres : ses dimensions en hauteur (height) et en largeur (width), son espace entre sa bordure et le contenu intérieur – ou marge interne - (padding), ses bordures (border) et son espace entre sa bordure et le contenu extérieur – ou marge externe - (margin).

Sur cette illustration, une visualisation peut-être plus simple de chaque partie qui compose une boîte :



Width	
Définit la largeur d'un élément (texte, filet, image, calque, …).	Valeur numérique positive + unité de mesure

Height	
Définit la hauteur d'un élément (texte, filet, image, calque, …).	Valeur numérique positive + unité de mesure

Remarque

Sur le web, souvent, on n'attribue pas de hauteur fixe à un conteneur pour une simple raison : on préfère que sa hauteur soit déterminée automatiquement par le contenu qui se trouve en lui.

On utilise par contre la propriété height pour des éléments dont on est certains à 100% qu'ils ne varieront par au niveau de la hauteur (mais ça reste non obligatoire).

Padding-top, padding-right, padding-bottom, padding-left	
Définit la valeur de remplissage : En haut à droite en bas à gauche d'un conteneur.	Valeur numérique + unité de mesure

Forme raccourcie

Vous pouvez utiliser la propriété padding qui est la forme raccourcie pour les 4 propriétés vues ci-dessus.

Si vous l'utilisez, vous devez respecter cet ordre :

```
… { padding: top right bottom left; } /* Si vos 4 valeurs sont différentes */
```

Par exemple :

```
… { padding: 5px 10px 15px 20px; } /* Si vos 4 valeurs sont différentes */
```

Si vous avez 4 valeurs identiques pour vos remplissages, vous opterez pour :

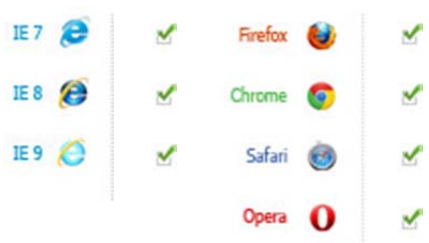
```
… { padding: 10px; } /* Si vos 4 valeurs sont identiques */
```

Si vous avez des valeurs identiques pour les remplissages horizontaux et verticaux :

```
… { padding: 10px 50px; }
/* Si vous avez 2 paires de valeurs identiques en top et en bottom et en left et en right */
```

Si vous avez des valeurs identiques pour les remplissages horizontaux mais pas les verticaux :

```
… { padding: 10px 20px 50px; }
/* Si vous avez 2 paires de valeurs identiques en left et right et des valeurs différentes en haut et en bas – H – DG - B */
```



Margin-top, Margin-right, Margin-bottom, Margin-left	
Définit la valeur de la marge : En haut à droite en bas à gauche d'un conteneur.	Valeur numérique + unité de mesure

Forme raccourcie

Vous pouvez utiliser la propriété margin qui est la forme raccourcie pour les 4 propriétés vues ci-dessus. Elle fonctionne exactement de la même façon que padding vu précédemment.

Centrer un élément horizontalement

La propriété margin est très utile en CSS pour pouvoir centrer un élément de type block là où il se trouve. Syntaxiquement, il suffit d'indiquer que ce conteneur a ses marges de gauche et droite réglées sur la valeur auto comme ceci :

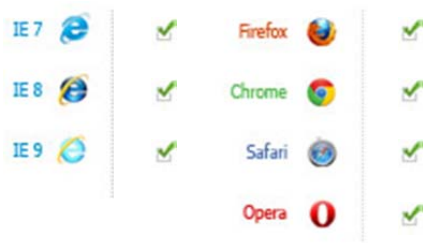
```
… { margin: 0 auto;} /* Centre horizontalement un conteneur là où il se trouve */
```

Attention, pour qu'un élément de type block soit centré, il doit nécessairement avoir une largeur plus petite que son parent sinon le centrage ne sert pas à grand-chose.
Malheureusement, **faire pareil sur les marges supérieures ne centre pas un élément sur la hauteur** de là où il se trouve (ce serait trop simple ☺).

Fusion des marges

La fusion des marges est une particularité du modèle de boîte qui concerne les marges externes verticales des éléments de type block positionnés dans le flux. Lorsque deux ou plusieurs éléments sont adjacents, qu'ils soient frères ou imbriqués (parent-enfant), leurs marges verticales se combinent pour n'en former qu'une seule : la plus grande des deux.

Pour contrer ce problème, entre un conteneur et son parent, voici quelques propositions :
- Appliquer une bordure (même transparente) sur le conteneur parent.
- Appliquer un padding sur l'élément parent (même de 1px).
- Si vous placez du contenu entre l'ouverture du conteneur parent et son enfant.
- Quand la propriété overflow :hidden est appliquée sur le parent.



Border-top, Border-right, Border-bottom, Border-left	

Écriture raccourcie pour affecter au bord d'en haut, droite, bas ou gauche d'un élément :
- Une épaisseur de trait
- Une couleur de trait
- Un style de trait
 	
Chaque valeur doit être espacée l'une de l'autre par un espace simple.
	Valeur numérique + unité de mesure
	
	#RRVVBB
	rgb(R%,V%,B%)
	nom de la couleur en anglais
	
	solid
	dashed
	dotted
	none

Exemple concret

```
p { border-top : 2px dashed red;} /* Place une border de 2 pixels de large, rouge et de style pointillées au paragraphe */
```

• Les listes à puce

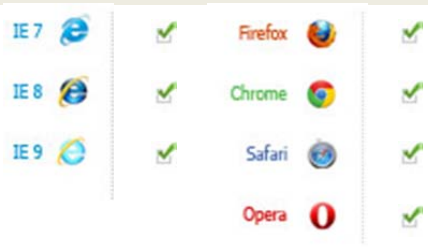
Les listes à puce ont aussi leur lot de propriétés CSS dédiées.

List-style-type	disc
Définit le type de liste.	Circle square decimal lower-roman upper-roman lower-alpha upper-alpha

List-style-image	none
Définit une image comme puce	url(adresse et chemin de l'image)

Utilisez background pour mettre une puce plutôt que list-style-image

Pourquoi ? Parce que la propriété background a plus de possibilité en terme de positionnement que la propriété list-style-image, elle est plus souple. Associée à un petit padding, un background-position et un no-repeat, vous obtiendrez un résultat similaire et plus souple.



List-style-position	outside
Définit la position des puces.	inside

Forme raccourcie

Vous pouvez utiliser la propriété list-style qui est la forme raccourcie pour les 3 propriétés vues ci-dessus.

```
… { list-style : upper-roman inside; }
```

• Changer l'état original d'une balise, ça sert à quoi ?

La propriété display et ses valeurs associées permet de changer l'état d'une balise.

On peut par exemple changer comme ça l'état d'une balise <a> qui, par défaut, est une balise inline pour la transformer en balise block.

L'avantage ? Avec une balise block, vous pouvez jouer avec les margin et padding sur la hauteur également, chose qui est impossible avec des éléments inline. Ceci n'était qu'un exemple parmi tant d'autres.

La valeur inline-block, par exemple, combine l'avantage d'une balise inline (pas de retour à la ligne) et d'une balise block (possibilité de pousser en haut et en bas les margin et padding.

La valeur none, en revanche, peut être utile pour jouer avec des effets de :hover, pour faire apparaître ou disparaître des éléments au survol.

display	
Définit comment un élément est affiché	None block inline inline-block table table-cell

visibility	visible
Définit la visibilité d'un élément.	hidden

Quelle différence entre display :none et visibility :hidden ?

C'est simple, dans les deux cas, vous ne verrez plus votre bloc sur lequel vous avez placé l'une des propriétés.

La seule différence, c'est que le visibility :hidden réserve la place que prenait la block et empêche n'importe quel élément de prendre sa place tandis que display :none sort complètement du flux le bloc et ne l'affiche plus, laissant ainsi tous les autres éléments du flux coulisser et prendre sa place.

• Le positionnement des blocs avec la propriété float

La propriété float spécifie le flottement horizontal d'une boîte (bloc), càd qu'elle spécifie de quel côté se fera son habillage (un peu comme avec l'alignement des balises img en HTML4).

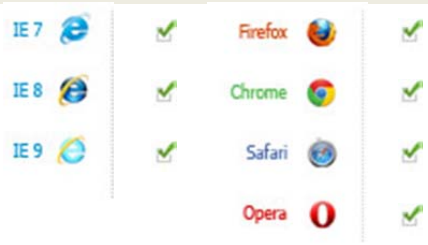
C'est cette propriété float qui, placée sur des balises , va servir à créer vos structures qui vount remplacer vos vieux tableaux (tout moisis).

float	none
Définit le flottement horizontal d'un bloc.	Left right none

Attention au Float : right

Un des effets un peu pervers de float quand il est utilisé avec la valeur right, c'est que 2 éléments floatés à droite alors qu'ils sont au même niveau vont bien se mettre l'un à côté de l'autre par rapport au bord droit du conteneur dans lequel ils sont placés MAIS dans l'ordre inverse de l'apparition dans l'XHTML.

C'est logique, puisque le premier bloc à être floaté à droite occupe en premier la place près du bord et le suivant viens occuper la place à côté de ce premier bloc.



clear	none
Définit les côtés d'une boite où l'habillage n'est pas autorisé.	Left right both all

Qui dit float dit clear :both ou overflow :hidden !

Le float a un autre effet pervers : les blocs floatés sont sortis du flux et donc, tout le contenu qui suit a tendance à venir se placer derrière les blocs floatés.

Pour contrer ce problème, 2 façons de faire différentes :
- Placer un élément vide en block juste en-dessous du dernier élément floaté qui aura pour propriété css unique clear :both
- Placer une propriété overflow :hidden sur l'élément qui contient les éléments floatés

• Le positionnement des blocs avec la propriété position

La propriété position a trois valeurs intéressantes pour des placements de bloc de façon précise. C'est avec cette position entre autre que vous allez pouvoir positionner un bloc de façon fixe sur une page tandis que le reste de la page continuera à scroller.

position	static
Définit la méthode de repérage pour positionner un élément. Absolute place le bloc par rapport aux quatre coins de son premier référent qui est en position relative, s'il n'en trouve pas dans ses ancêtres, il se réfère aux 4 coins de la fenêtre de navigation. Il sort également du flux, permettant aux éléments qui le suivent de se placer sous lui. Il y a donc superposition d'éléments. Relative place un bloc comme référent pour un bloc en absolu qui est placé comme enfant. Fixed place un bloc comme élément ne bougeant pas par rapport aux 4 coins de la fenêtre.	Absolute relative fixed

Avec ces positions absolute et fixed voient le jour 4 nouvelles propriétés CSS qui permettent d'aligner ces blocs en absolute et fixed par rapport aux bords de leur référent.

Top, right, bottom, left	
Définit la distance en abscisse (left ou right) ou en ordonnée (top ou bottom) d'un élément.	Valeur numérique + unité

Avec l'apparition de la position absolute, il devenait nécessaire de pouvoir jouer avec la superposition des calques et de donc pouvoir leur donner un ordre d'empilage les uns sur les autres. C'est là que la propriété CSS z-index intervient. Elle permet de jouer sur la profondeur d'affichage de calques qui sont en position absolute, fixed ou relative.

z-index	
Définit l'ordre de superposition de manière arbitraire avec une valeur numérique positive. Plus le z-index est élevé, plus l'élément se trouve haut dans l'affichage.	Ordre d'empilage du calque

overflow	visible
Définit le comportement d'un calque lorsque son contenu est supérieur à sa taille	Auto scroll hidden