Chapitre 4

Les sélecteurs

Pseudo-classes et pseudo-éléments (CSS3 et 4)

Objectifs du chapitre :

- Comprendre l'arborescence et le principe de sélection hiérarchique (éléments ancêtres, parents, enfants et frères)
- Simplifier avec le regroupement des propriétés, des sélecteurs et la hiérarchie
- Les combinateurs
- Les sélecteurs d'attributs
- Les pseudo-classes
- Les pseudo-éléments
- Calculer la spécificité d'un sélecteur

1. LES SELECTEURS ET L'ARBORESCENCE

Les différents éléments composant un fichier HTML (titres, paragraphes, listes, liens, images…) s'organisent sous la forme d'un « arbre généalogique », que l'on nomme également « arbre du document » (ou DOM pour *Document Object Model*). Au chargement de la page HTML, le navigateur construit un modèle arborescent correspondant à l'ensemble des balises de la page web appelé le DOM. Il charge ensuite la ou les feuilles de styles puis applique les différentes règles aux objets du DOM au fur et à mesure de la lecture des styles.

Les sélecteurs sont la base de la recheche d'éléments HTML dans le DOM.

Selon la structure du document et principalement son arborescence (c'est-à-dire la hiérarchie des blocs, chacun étant placé « sous » l'éventuel bloc qui le contient), il est possible de pointer directement certains éléments en fonction de leur situation dans le document. Cette technique porte le nom de « **sélection hiérarchique** ».

1.1. Éléments ancêtres, parents, enfants et frères

Chaque boîte peut en contenir d'autres. Un paragraphe <p> peut ainsi renfermer des boîtes définies par les éléments ou , et se trouver lui-même inclus dans un élément . **Toutes ces imbrications de boîtes forment une hiérarchie arborescente entièrement comprise dans la boîte de l'élément racine du document (<body>)**. Il est essentiel de comprendre cette arborescence pour bien utiliser les positionnements.

Ceux-ci peuvent être **ancêtre**, **parents**, **enfants** ou **frères**. Ces différents éléments composent une hiérarchie d'imbrications.

1. Un élément **Ancêtre** est un élément qui contient un élément ou une hiérarchie d'éléments
2. Un bloc **Parent** est un élément contenant **directement** un autre bloc.
   Par exemple, un *div* contenant un paragraphe *p*.
   Attention : si ce paragraphe contient lui-même des éléments (ex: *strong*), *div* ne sera pas Parent de l'élément *strong* mais uniquement son **Ancêtre**.
   Le Parent est donc l'Ancêtre immédiat – de 1er niveau.
3. Un bloc contenu directement dans un autre bloc est dit **Enfant** de cet élément. Par exemple, les éléments *li* sont enfants de leur conteneur *ul*.
4. Les éléments ayant le même élément Parent sont appelés **Frères**.

Exemple :

```
<div>
<h1> Titre </h1>
<p> paragraphe de texte à la suite du titre H1 </p>
<h2> sous-titre </h2>
<p> paragraphe de texte à la suite du sous-titre avec certains <strong> termes
importants </strong>  </p>
</div>
```

Le bloc est ***parent*** des titres <h1>, <h2> et des 2 <p>, c'est aussi un ***ancêtre*** de
Le second <p> est ***parent*** de
Les titres <h1>, <h2> et les 2 <p> sont ***frères***

1.2. Regroupement des sélecteurs

Au lieu de répéter la même règle pour plusieurs éléments, le W3C a prévu une notation raccourcie des mêmes propriétés pour plusieurs balises :

Ainsi :

```
.texte {
     margin-left:0;
}
p {
```

```
     margin-left:0;
}
h1 {
     margin-left:0;
}
h2 {
     margin-left:0;
}
```

Sera identique à :

```
.texte, p, h1, h2 {
margin-left:0;
}
```

1.3. Cibler les sélecteurs descendants

Les sélecteurs descendants (une balise contenue dans une autre) permettent de cibler les descendants d'un élément ou d'un groupe d'éléments donnés. Ils sont indiqués par un **espace** entre deux autres sélecteurs.

```
h3 em
     {
          ici les styles ;
     }
```

Sélectionne toutes les balises situées à l'intérieur d'une balise <h3>. Notez qu'il n'y a pas de virgule entre les deux noms de balise.

Ainsi

```
a img {border-width:0;}
```

ne supprimera les bordures qu'aux images contenues dans un lien

1.4. Le sélecteur universel

Identifié par un astérisque, il est souvent utilisé pour appliquer des mises en forme à tous les éléments d'une page.

```
/* Pour supprimer le remplissage et les marges par défaut du navigateur sur tous
les éléments : */
* {
  padding: 0;
  margin: 0;
}
```

Combiné avec d'autres sélecteurs, le sélecteur universel peut servir à **mettre en forme tous les descendants d'un élément particulier** ou à ignorer un niveau de descendants.

```
#header * {
  font-weight: bolder;
  color: red;
}
```

1.5. Simplifier grâce à la hiérarchie

Dans un site web bien architecturé, cette méthode permet d'éviter le recours aux classes dans la plupart des cas.

Exemple :

```
<ul id="menu">
<li><a class="lienmenu" href="…">premier lien du menu</a></li>
<li><a class="lienmenu" href="…">deuxième lien du menu</a></li>
…
</ul>
```

Pourra devenir :

```
<ul id="menu">
<li><a href="…">premier lien du menu</a></li>
<li><a href="…">deuxième lien du menu</a></li>
…
</ul>
```

Vous désignez vos liens de cette manière :

```
#menu a {propriétés}
```

Ou plus précis :

```
ul#menu li a {propriétés}
```

Au lieu de leur attribuer différentes classes, pensez à appliquer une classe ou un ID à l'un de leurs parents, puis ciblez-les à l'aide d'un sélecteur descendant.

1.6. Limiter les styles aux balises dotées de certaines classes ou identifiants

Ainsi,

```
a.nav {background-color:yellow;}
```

ne désignera que les liens <a> appartenant à la classe nav, et ...

```
<a class="nav" href="…"> lien hypertexte </a> /*sera affecté par la propriété*/

<p class="nav"> paragraphe </p> /* ne sera pas affecté
<p class="nav"> <a href="..."> lien </a> </p>  /* ne sera pas affecté
```

1.7. Un nommage polyvalent

Il est autorisé de **cumuler un attribut d'identifiant et un attribut de classe** pour le même élément, par exemple :

```
<p class="exotique" id="fruits">
```

Un élément peut disposer de **noms de classes multiples** en les séparant par un espace, par exemple :

```
<p class="fruit exotic">.
```

1.8. Héritage

Certaines propriétés, comme la couleur ou la taille des polices, sont héritées par les descendants des éléments auxquels elles sont appliquées. Notons que Internet Explorer et Netscape ont quelques problèmes avec l'héritage des tailles de police dans les tableaux, il est donc préférable dans ce cas précis de définir leur taille de police séparément car *tous les styles qui sont appliqués directement à un élément redéfinissent un style hérité*.

L'héritage est très utile car il évite d'ajouter le même style à chaque descendant d'un élément :

```
p, div, h1, h2, h3, ul, ol, dl, li {color: black;}
```

=

```
body {color: black;}
```

L'héritage bien utilisé peut aider à réduire le nombre et la complexité des sélecteurs dans le code.

Lecture Web :
http://www.alsacreations.com/tuto/lire/545-Comprendre-l-heritage-et-la-parente-des-styles-CSS.html
http://www.css-faciles.com/heritage-cascade.php

2. LES COMBINATEURS

Les combinateurs permettent d'agir sur un élément en fonction de sa relation avec un autre.

Nous connaissons déjà :

```
h1 em {
      font-size:2em;
}
```

Qui ciblera tous les éléments contenus dans l'élément <h1>

D'autres combinateurs plus avancés permettent de ne désigner que le premier enfant d'un élément, ne pointer que des éléments directement adjacents à un autre, …

2.1. Le sélecteur d'enfants[1]

Le signe « **>** » indique un sélecteur d'enfant, il désigne les éléments placés directement au dessous, **il ne cible que les descendants immédiats de l'élément** (que l'on appelle ses enfants) :

```
#nav > li {
  background: url(folder.png) no-repeat left top;
  padding-left: 20px;
}

<ul id="nav">
  <li><a href="/home/">Accueil</a></li>/*image présente*/
  <li><a href="/services/">Services</a>/*image présente*/
    <ul>
      <li><a href="/services/design/">Conception</a></li>/*pas d'image*/
      <li><a href="/services/development/”>Développement</a></li>/*pas d'image*/
      <li><a href="/services/consultancy/">Consulting</a></li>/*pas d'image*/
    </ul>
  </li>
  <li><a href=”/contact/”>Nous contacter</a></li>/*image présente*/
</ul>
```

2.2. Le sélecteur de frère adjacent[2]

Le signe « **+** » indique un sélecteur adjacent, il permet de **cibler un élément précédé par un autre élément qui possède le même parent :**

```
h3 + p
{  /* mes styles */ }
```

Sélectionne toutes les **premières** balises <p> situées après une balise <h3>.

2.3. Le sélecteur d'adjacence indirecte

Sélecteur utilisant le caractère « **~** **»**, il permet d'appliquer des styles d'un seul coup **à tous les frères d'un élément ciblé** répondant à un sélecteur précis. Il est important de noter que ce sélecteur **ne cible que les frères suivants d'un élément ciblé, en aucun cas les frères précédents ne seront concernés**.

Exemple01

```
h2 ~ p {    /* mes styles */  }
```

Ici les styles seront appliqués à tous les p frères de h2, même si un autre frère vient se mêler entre deux p.

Exemple02

[1] Celui-ci est pris en charge par Internet Explorer 7 et ses versions ultérieures s'il n'y a pas de commentaires entre les éléments ciblés.
[2] Celui-ci est pris en charge par Internet Explorer 7 et ses versions ultérieures s'il n'y a pas de commentaires entre les éléments ciblés.

```
li:hover ~ li {
   opacity: 0.4;
}
```

Au survol d'un des li du menu, les autres li (qui suivent) apparaîtront plus clairs (**`opacity: 0.4`**;)

2.4. Le sélecteur de parents (CSS4)

Sélecteur utilisant le caractère « **!** », il permet de sélectionner le parent d'un élément. Il était impossible avant le CSS4 de sélectionner un élément en fonction de l'état de son enfant.

```
ul li! a.active {
   color: red;
}
```

Nous allons styler les éléments li dont le fils « a » possède la classe active.

3. LES SELECTEURS D'ATTRIBUT

Le **sélecteur d'attribut** permet de cibler un élément en fonction de l'existence ou de la valeur d'un attribut.

3.1. Signaler un title à un acronyme

```
<p>L'acronym <acronym title="Institut de Formation Supérieure de la ville de Wavre"> IFOSUP
</acronym> désigne le nom de l'école.</p>
<!--L'attribut title permet de faire apparaître une info-bulle au survol de l'élément-->
```

```
acronym[title] {
  border-bottom: 1px dotted #999;
} /* lorsque le mot possède un attribut "title", il est souligné pointillé*/
acronym[title]:hover, acronym[title]:focus {
  cursor: help;
}  /* lorsque le mot est survolé, le curseur change d'aspect*/
```

3.2. Différencier des types de liens

Nous souhaitons *signaler les liens externes* en y ajoutant une petite icône à côté, et ainsi laisser le choix à l'utilisateur d'ouvrir le lien dans une nouvelle fenêtre.

La spécification CSS3 permet de cibler un élément en établissant la correspondance entre un fragment de texte choisi et une partie de la valeur d'un attribut (mise en correspondance de sous-chaînes) :

`[attribut^="texte"]` caractérise un attribut dont la valeur commence exactement par la chaîne "`texte`".
`[attribut$="texte"]` représente un attribut dont la valeur finit exactement par le suffixe "`texte`".
`[attribut*="texte"]` désigne un attribut dont la valeur contient au moins une fois la souschaîne "`texte`".

Cette technique fonctionne en ciblant d'abord tous les liens qui commencent par le texte `http:` avec le sélecteur d'attribut `[att^="http :"]` :

```
a[href^="http:"] {
  background: url(/img/lienexterne.gif) no-repeat right top;
  padding-right: 10px;
}
```

Ou encore pour distinguer les liens de messagerie :

```
a[href^="mailto:"] {
  background: url(img/email.png) no-repeat right top;
  padding-right: 10px;
}
```

3.3. Signaler un document téléchargeable

Le sélecteur d'attribut `[att$=val]` permet de cibler les attributs qui se terminent par une valeur particulière, comme `.pdf` ou `.doc` :

```
a[href$=".pdf"] {
  background: url(img/pdflink.gif) no-repeat right top;
  padding-right: 10px;
}
a[href$=".doc"] {
  background: url(img/wordlink.gif) no-repeat right top;
  padding-right: 10px;
}
```

Comme pour les précédents exemples, vous pouvez donc signaler les liens vers les documents Word ou PDF avec leur propre icône, afin de prévenir les utilisateurs qu'ils cliquent sur un document téléchargeable au lieu d'un lien vers une autre page.

3.4. Cibler les éléments de formulaire

Eléments <input> dont l'attribut type a exactement pour valeur "text"

```
input[type="text"] {
      border:1px solid green;
}
```

Eléments <input> dont les attributs text et name ont respectivement les valeurs "text" et "prenom"

```
input[type="text"][name="prenom"] {
      text-transform:uppercase;
}
```

4. LES PSEUDO-CLASSES

4.1. Qu'est-ce qu'une pseudo-classe ?

Une pseudo-classe CSS est un **mot-clé ajouté au sélecteur** pour indiquer un état particulier de l'élément qui doit être sélectionné. Les pseudo-classes **portent sur des éléments existants dans le code source du document** et auxquels on peut accéder par des caractéristiques autres que leur nom, attribut ou contenu. C'est une manière de cibler un élément sans ajouter une classe (manuellement ou par l'intermédiaire de JavaScript).

Exemples de pseudo-classes :

- ✓ **structurelles** : `:first-child, :last-child, nth-child(n)`
- ✓ **d'ancres :** `:link` et `:visited`
- ✓ **dynamiques** `:hover, :active et :focus`
- ✓ **de langue** `:lang()`

4.2. Syntaxe

```
selecteur:pseudo-classe {
  propriété: valeur;
}
```

4.3. Les pseudo-classes pour les liens

Grâce aux CSS, vous pouvez appliquer aux liens des changements de couleur, de police, des soulignés, … Le CSS permet également de définir ces propriétés différemment, selon que le lien est visité, non visité, activé, ou si le curseur le survole. **Pour contrôler ces effets, on utilise ce qu'on appelle des pseudo-classes.**
Les liens sont définis en HTML avec des balises <a>, on l'utilise telle quelle comme sélecteur CSS :

```
a {
```

```
     color: red;
}
```

Permettra d'avoir tous les liens de couleur rouge.

4.3.1. Ciblage dynamique

Un lien est susceptible d'avoir plusieurs états. Par exemple, il peut être visité ou non. Vous pouvez utiliser des pseudo-classes pour assigner des styles différents aux liens visités et non visités.

```
a:link {
     color: blue;
}
a:visited {
     color: red;
}
```

Donnera des liens de couleur bleu et des liens visités de couleur rouge…

Utilisez respectivement **a:link** pour les liens **non visités** et **a:visited** pour ceux **visités**. Les liens **actifs** (quand on clique dessus) ont pour pseudo-classe **a:active**, et la pseudo-classe **a:hover** intervient lorsque le **curseur survole** le lien.
a:focus : l'état focus est ajouté pour la prise en charge du clavier.
Lorsque deux règles possèdent la même spécificité, la dernière à être définie prend le dessus. Pour éviter des dysfonctionnements liés à l'ordre de déclaration des pseudo-classes, appliquez les styles de liens dans l'ordre suivant : `a:link, a:visited, a:hover, a:focus, a:active.`

`:hover`, `:active` et `:focus` sont des pseudo-classes dynamiques, qui peuvent théoriquement s'appliquer à n'importe quel élément[1].

4.3.2. Ciblage dynamique complexe

Les pseudo-classes peuvent être adjointes les unes aux autres afin de créer des comportements plus complexes :

```
/* affiche tous les liens visités survolés en vert olive : */
a:visited:hover {color:olive;}
```

4.4. Mise en forme des cibles des liens grâce au selecteur target

:target est une pseudo-classe CSS3 qui permet de **cibler un élément qui est la cible d'un lien** (une ancre). *La pseudo-classe :target permet donc d'appliquer un style à l'élément qui reçoit le lien.*

```
.comment:target {
 background-color: yellow;
}
```

4.5. first-child

La pseudo-classe **:first-child**, désigne le **premier élément enfant** au sein d'un élément**.** Par exemple, `li:first-child` qualifie le premier élément `<li>` d'une liste `<ul>` ou`<ol>`, pratique pour appliquer une bordure sous les éléments d'une liste, sauf le premier ou mettre en évidence les 1er paragraphes de plusieurs articles.

4.6. Les selecteur nth-child, nth-last-child, nth-of-type, only-child et only-of-type

:nth-child est une pseudo-classe CSS3 qui permet de **cibler le n-ième enfant d'un élément**.
nth signifie n-ième, ce sélecteur permet donc de sélectionner le/les n-ièmes enfants d'un élément- Il y a plusieurs façon de procéder:

Valeur numérique entière (cible un seul enfant)

```
ul li:nth-child(2){
     background-color:orange;
}
```

Nous ciblons ici un seul enfant, le 2ème.

[1] Internet Explorer > 7 prend en charge :hover sur n'importe quel élément, mais ignore :active et :focus.

Valeur de n, éventuellement suivie d'une valeur entière ajoutée ou soustraite.

```
ul#ciblage02 li:nth-child(2n){
     background-color:green;
     }
```

*Nous ciblons ici tous les éléments paires car 2n nous donne 2*0=0, 2*1=2, 2*2=4, 2*3=6 et ainsi de suite…*

:nth-last-child permet de **cibler des éléments suivant la position qu'ils occupent dans la liste des enfants de l'élément parent**. C'est comme **:nth-child** sauf que l'on part de la fin. Les arguments acceptés sont les mêmes que `:nth-child` ;

:only-child cible un élément s'il est l'unique enfant ;

Le sélecteur **:nth-of-type** ne prend en compte que **les éléments de même type pour effectuer les groupes** ;

:only-of-type cible un élément s'il est l'unique enfant ce de type ;

4.7. Le selecteur :not()

:not est une pseudo-classe qui permet de retirer un élément, ou l'état d'un élément, du sélecteur.

```
.item:not(p){ ... }
```

Cet exemple cible tous les éléments qui ont la classe .item, mais qui ne sont pas des <p>

```
ul li:not(:hover){ ... }
```

Cet exemple cible les éléments <li> non survolés :

```
div:not(.item){ ... }
```

Cet exemple cible les éléments qui n'ont pas la classe .item :

Limites du sélecteur :not

En CSS3, seuls les sélecteurs et pseudo-classes simples peuvent êtres utilisées et les :not() ne peuvent pas être imbriqués. Depuis le CSS4, elle permet de désélectionner les éléments contenus entre les parenthèses.

```
p:not(.S1, #S2) {
     color:blue;
}
```

Le texte contenu dans tous les paragraphes sera de couleur bleu sauf si les paragraphes ont la classe S1 et/ou l'id S2.

4.8. Les selecteurs utiles aux formulaires[1]

:checked cible les éléments cochés (bouton radio ou case à cocher) ;

:required cible les éléments de formulaire définis comme requis (attribut required) ;

:valid cible les éléments de formulaire respectant la valeur attendue ;

:invalid cible les éléments ne respectant pas la valeur attendue ;

4.9. Le selecteurs :contains(value)[2]

Correspond aux éléments dont le contenu textuel contient la sous-chaîne donnée en argument.

```
p:contains('essai'){
          background:#900;
     }
```

[1] Des exemples concrets de l'utilisation de ces sélecteurs sont disponibles dans le chapitre consacré aux formulaires

[2] L'usage de la pseudo-classe de contenu (:contain) est restreint aux médias statiques.

Signifie que tous les éléments "p" contenant la sous chaine "essai" auront pour couleur d'arrière plan, la valeur '#900'.

4.10. Le selecteurs :empty

Correspond aux éléments n'ayant pas d'enfant.

4.11. Les pseudo-classes de sélection :matches(), :nth-match(), nth-last-match() - CSS4

La pseudo-classe de sélection **:matches** permet de sélectionner plusieurs arguments pour un même élément. Grâce à ce sélecteur nous allons pouvoir grouper et identifier des éléments dans nos feuilles de styles.

```
p:matches(:hover, .S1, #S2) {
     color:blue;
}
```

Le texte contenu dans les paragraphes qui seront survolés (:hover) et/ou avec la classe S1 et/ou avec l'id S2 seront bleu.

La pseudo-classe de sélection **:nth-match()** rassemble **:nth-child()** , qui permet de cibler facilement et rapidement un élément pour une valeur positive ou égale à 0 de n d'un élément parent, et **:nth-match()** expliqué au-dessus..

```
p:nth-matches(2n+1 of .S1, #S2) {
     color:blue;
}
```

Le texte contenu dans les paragraphes pairs avec la classe S1 et/ou le paragraphe avec l'id S2 sera bleu.

La pseudo-classe de sélection **:nth-last-child()** fonctionne de la même façon que **:nth-match()** mais en partant de la fin.

5. LE PSEUDO-ELEMENT

Un pseudo-élément est une chose que l'on n'aurait pas pu cibler sans ajouter une balise (universelle ou non) au code source. Exemple: à la place de **::first-letter**, on aurait dû ajouter un élément span entourant la première lettre de l'élément ciblé.
CSS3 propose une nouvelle syntaxe pour les pseudo-éléments : au lieu du « : » il faudra écrire « **::** » (deux fois deux-points). La syntaxe d'une pseudo-classe reste inchangée. Notez que tous les navigateurs supportent les deux orthographes pour les pseudo-éléments.

5.1. pseudo-éléments ::first-letter et ::first-line

Les pseudo-éléments **::first-letter** et **::first-line** correspondent respectivement au premier caractère (pratique pour créer une lettrine) et à la première ligne de texte au sein d'un élément de type bloc ou équivalent.

```
p::first-letter[1]{
  float: left; /* positionnement de la lettrine dans le conteneur */
  font: bold 3em Georgia, "Times New Roman", Times, serif;
  color: #900;
  padding: 3px 8px;
  margin: 5px 5px 0 0;
  border: 1px solid #900;
}
```

[1] Attention : pas d'espace avant l'ouverture de l'accolade

5.2. pseudo-éléments ::selection

Applique la règle de style à la sélection du texte de l'élément faite par l'utilisateur.

```
p::selection { background:#006644 }
```

A la sélection, le texte sélectionné aura une couleur d'arrière plan de valeur '#006644'.

5.3. Générer du contenu avec ::before et ::after

Avec les pseudo-éléments **::before** et **::after**, on peut automatiser l'inclusion de caractères ou d'images grâce à l'attribut content.

::before **Cible le premier enfant de l'élément sélectionné.** Généralement utilisé pour ajouter du contenu esthétique à un élément, en utilisant la propriété CSS **content**. C'est un élément de type en-ligne par défaut.

```
element::before { propriétés de style }
```

:after **Cible le dernier enfant de l'élément sélectionné.** Généralement utilisé pour ajouter du contenu esthétique à un élément, en utilisant la propriété CSS **content**. C'est un élément de type en-ligne par défaut.

```
element::after { propriétés de style }
```

content est utilisée avec les pseudo-éléments **::before** et **::after** pour générer du contenu dans un élément.

```
h1::before  { content: "Chapitre: "; }
```

HTML regorge de caractères spéciaux qui peuvent être exploités et ainsi remplacer certaines images.

```
.fini::after{
content: "\263A";
}
```

Attention ! Rien de ce qui est généré de cette façon n'est accessible et IExplorer < 8 ne l'implémente pas. Ainsi, il ne faut en aucun cas utiliser pour diffuser des informations importantes.

Lecture Web :
Utiliser les sélecteurs CSS : http://css.developpez.com/tutoriels/utiliser-nouveaux-selecteur-css-3/ et http://debray-jerome.developpez.com/articles/les-selecteurs-en-css3/

6. CASCADES ET SPECIFICITES

Nous savons que lorsque deux règles ciblent le même élément, les CSS gèrent ces conflits par le biais d'un processus appelé « cascade ».
Les règles sont ensuite ordonnées en fonction de la spécificité du sélecteur : les règles aux sélecteurs plus spécifiques redéfinissent celles dont les sélecteurs sont moins spécifiques. Si deux règles possèdent la même spécificité, c'est celle qui est définie en dernier qui prend le dessus.

6.1. Calcul de la spécificité d'un sélecteur

La nature du CSS est d'attribuer un poids aux valeurs en fonction de l'endroit où elles apparaissent dans la "cascade".
Un style CSS sera appliqué en fonction de quatre facteurs selon l'ordre suivant :

1. spécificité directe (par exemple les éléments <ul> se voient attribuer une largeur de 300px)
2. importance du style (par exemple le style est t'il déclaré dans la feuille de style de l'utilisateur ?)
3. spécificité du sélecteur (par exemple les éléments <ul> à l'intérieur d'éléments se voient attribuer une largeur de 400px)
4. ordre d'apparition (par exemple le dernier déclaré s'applique)

6.2. Utiliser la déclaration !important

Quand la déclaration !important est utilisée sur une paire spécifique de propriété/valeur, cette valeur là échappera à la cascade et deviendra, comme son nom le suggère, la valeur la plus importante pour cette propriété, surclassant toutes les autres valeurs.

```
#leftSide {
    background-color: green !important;
}

#header #leftSide {
    background-color: red;
}
```

Dans l'exemple ci-dessus, même si le style du background-color spécifié dans la seconde règle est plus spécifique du fait du sélecteur et est déclaré en dernier, la première règle sera prioritaire du fait de l'ajout de la déclaration !important. Ainsi, la couleur d'arrière-plan de l'élément en question (l'élément #leftSide) sera vert au lieu de rouge.

Lecture Web :
http://openweb.eu.org/articles/cascade_css