Extrait du livre CSS avancées - Vers HTML 5 et CSS 3
Raphaël Goetter

Pseudo-classes et pseudo-éléments CSS 3

Les spécifications CSS 3 sont très riches de pseudo-éléments et pseudo-classes inédites qui apportent une souplesse supplémentaire dans la sélection des éléments selon leur contexte. Ainsi, il devient possible de cibler uniquement le dernier enfant d'un conteneur, le premier enfant d'un certain type, uniquement les éléments pairs ou impairs, ceux qui ne respectent *pas* une condition, etc.

> **COMPATIBILITÉ** **Nouvelles pseudo-classes**
>
> J'ai choisi de classer ces pseudo-classes par ordre de compatibilité : les premières sont reconnues dès Firefox 3.0 et Safari 2.0, les dernières ne sont comprises qu'à partir de Firefox 3.5 et Safari 3.0. Il est à noter que ces pseudo-classes CSS 3 ne seront toutes reconnues qu'à partir d'Internet Explorer 9 et que, d'une manière générale, toutes sont adoptées par Chrome et par Opera depuis sa version 9.

Signalons que depuis CSS 3, une convention d'écriture a été proposée par le W3C pour distinguer les pseudo-classes des pseudo-éléments. Ainsi, ces derniers s'écrivent dorénavant à l'aide d'un double deux-points (`::first-line`, `::first-child`, `::after`, `::before`), tout en autorisant une rétrocompatibilité avec l'écriture CSS 2.

:lang

La pseudo-classe `:lang` permet de cibler un élément dont la langue correspond à une certaine valeur indiquée entre parenthèses.

```
html:lang(fr) q {
  background-color: gray;
}
```

À moins d'avoir dans l'idée de mettre en forme certaines parties différemment selon la langue du document, comme les guillemets à la française sur des citations, l'intérêt de ce sélecteur est

assez limité dans le cadre d'un site monolingue ; mais il acquiert une redoutable efficacité sur les sites polyglottes.

En outre, il a l'avantage d'être reconnu sur tous les navigateurs modernes à partir d'Internet Explorer 8. Il permettrait par conséquent de cibler tous les navigateurs actuels – et uniquement ceux-là – à l'aide d'un sélecteur qui débuterait par `:lang(fr)`. Cette astuce suppose bien entendu que l'attribut HTML `lang` soit appliqué à un élément de structure principal (généralement la balise `<html>`) et qu'il ait pour valeur `"fr"` !

```
#kiwi { /* Pour tout le monde, dont IE6/IE7 */
  float: left;
  width: 300px;
}
:lang(fr) #kiwi {  /* Pour les navigateurs modernes et IE8+ */
  display: table-cell;
  float: none;
  width: auto;
}
```

Notez enfin que `:lang()` hérite de la langue du parent ou de l'ancêtre : si `<div lang="fr">` contient un paragraphe, alors le sélecteur `p:lang(fr)` trouve sa cible.

L'exemple précédent peut donc parfaitement être remplacé par la syntaxe suivante :

```
#kiwi:lang(fr) {
  …
}
```

:empty

Reconnu à partir de Firefox 3.0, Chrome, Opera 9.5, Safari 2.0 et Internet Explorer 9, le sélecteur `:empty` fait référence à un élément vide de tout contenu (balise ou texte).

Ainsi, `<p></p>` sera concerné par le sélecteur `:empty`, contrairement à `<p><span></span></p>`, `<p>kiwi</p>` ou encore `<p> </p>`.

Cette pseudo-classe trouve habituellement son utilité au sein de tableaux de données dynamiques où des cellules vides de contenu peuvent ainsi se distinguer des autres :

```
td:empty {
  background-color: gray;
}
```

:root

Comme `:empty`, la pseudo-classe `:root` est reconnue depuis Firefox 3.0, Opera 9.5, Safari 2.0 et annoncée sur IE9. Cependant son intérêt est extrêmement limité : en HTML, ce sélecteur désigne uniquement, et toujours, l'élément racine `<html>`. La seule différence avec le sélecteur `html` est que le poids de `:root` est supérieur.

```
:root { /* une image de fond sera appliquée sur l'élément <html> */
  background-image: url(kiwi.jpg);
}
```

:target

Si votre page web contient des ancres internes, c'est-à-dire des éléments nommés par des `id`, et des liens pointant vers ces éléments, alors la pseudo-classe `:target` désigne l'élément ciblé par l'ancre en question.

Dans l'exemple suivant, l'arrière-plan du titre `<h2>` devient jaune lorsque l'on clique sur le lien qui y mène (figure 8-26) :

Partie HTML

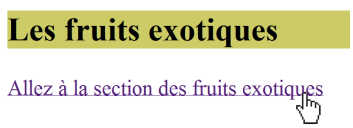
```
<h2 id="exotic_fruits">Les fruits exotiques</h2>
…
<a href="#exotic_fruits">Allez à la section des fruits exotiques</a>
```

Partie CSS

```
h2:target {
  background: #CCCC66;
}
```

Figure 8-26

Illustration de la pseudo-classe `:target`



Tout comme `:hover` et `:focus`, `:target` est considérée comme une pseudo-classe dynamique, dans la mesure où elle interagit avec l'action du visiteur. Ce sélecteur est reconnu à partir de Firefox 3.0, Chrome 2, Opera 9.5, Safari 2.0 et voit le jour sur IE9. Il est notamment employé sur Wikipédia pour la mise en exergue des notes de bas de page lorsqu'elles sont atteintes, mais nous allons voir d'autres exemples pratiques en fin de ce chapitre.

QUELQUES EXEMPLES PRATIQUES **Pseudo-classe :target**

J'ai rassemblé sur le site IE7nomore.com quelques cas d'usages pour le moins originaux de la pseudo-classe `:target`. N'hésitez pas à visionner et exploiter les codes source de ces pages pour aller plus loin.

Vous y trouverez par exemple des effets de transitions dans une galerie d'images :

- http://www.ie7nomore.com/fun/slideshow/
- http://www.ie7nomore.com/fun/scroll/

... mais aussi un menu déroulant fonctionnant au clic :

- http://www.ie7nomore.com/fun/menu/

:not

La pseudo-classe de négation `:not()` cible un élément qui ne correspond *pas* au sélecteur déterminé entre les parenthèses (figure 8-27). Par exemple, `:not(a)` désignera tous les éléments de la page à l'exception des liens et `a:not(:visited)` pointera tous les liens sauf ceux déjà visités.

Dans la pratique, ce sélecteur se révélera utile si vous souhaitez mettre en forme un groupe d'éléments mais avec des exclusions spécifiques.

Par exemple, pour désigner tous les éléments d'une liste sauf le premier, nous pourrions écrire :

```
li:not(:first-child) {
  background: #CCCC66;
}
```

Figure 8-27

Illustration de la pseudo-classe :not

Sans surprise, la pseudo-classe de négation est reconnue depuis Firefox 3.0, Chrome 2, Opera 9.5, Safari 2.0 et adoptée par Internet Explorer 9.

:last-child

Le sélecteur `:last-child`, annoncé en CSS 3 (tandis que son aîné `:first-child` date de CSS 2.1) désigne, comme vous l'avez deviné, un élément dernier enfant de son parent.

L'action de supprimer la marge basse du dernier paragraphe d'un bloc pourrait s'écrire de la sorte :

```
.bloc p:last-child {
  margin-bottom: 0;
}
```

Comme pour la plupart des autres pseudo-classes CSS 3, tous les navigateurs actuels reconnaissent `:last-child`, mais il faudra patienter jusqu'à la démocratisation d'Internet Explorer 9 pour l'employer à tour de bras.

:nth-child

Reconnue à partir de Firefox 3.5, Chrome 3, Opera 9.5, Safari 3 et IE9, la pseudo-classe `:nth-child()` s'applique au(x) n-ème(s) enfant(s) d'un élément.

Les valeurs contenues au sein de la parenthèse de `:nth-child()` peuvent être :

- un chiffre (entier positif ou négatif) – le premier enfant correspond à la valeur « 1 » ;
- une formule de type *an+b*, avec *a* et *b* deux chiffres ; *n* prendra toutes les valeurs à partir de zéro ;
- les mots-clés `even` ou `odd` qui symboliseront tous les fils pairs (`even`) ou impairs (`odd`) d'un parent, ce qui est idéal pour avoir des styles appliqués en alternance aux lignes d'un tableau, par exemple. Notez que `odd` a la même signification que *2n+1* (figure 8-28) et que `even` a la même que *2n*.

Voici quelques exemples concrets pour illustrer ce sélecteur :

```
li:nth-child(2) {…} /* le deuxième enfant d'une liste s'il s'agit d'un <li> */
li:nth-child(2n) {…} /* chaque <li> sur deux = les éléments pairs */
li:nth-child(even) {…}  /* les <li> pairs (idem que précédent) */
li:nth-child(2n+3) {…}  /* le 3e, le 5e, le 7e... */
```

Figure 8-28

Illustration de :nth-child(odd)

Sachez qu'une valeur négative de *n* est autorisée et offre des possibilités intéressantes : ainsi, le sélecteur `:nth-child(-n+3)` cible les trois premiers éléments uniquement (figure 8-29). Plutôt pratique, non ?

Figure 8-29

Illustration de :nth-child(-n+3)

TESTEZ EN LIGNE **:nth-child**

Rien n'est mieux que quelques exercices pour bien comprendre le principe de cette pseudo-classe déroutante. Le site CSSTricks propose un outil en ligne où vous pouvez observer en direct le comportement obtenu selon les différentes valeurs de `:nth-child` introduites :

▸ http://css-tricks.com/examples/nth-child-tester/

:nth-of-type

Variante du sélecteur précédent, `:nth-of-type` ne compte et ne sélectionne que les éléments du même type.

Pour bien comprendre la différence avec le sélecteur `:nth-child`, prenons un exemple de code HTML :

```
<div>
  <p></p>
  <span></span>
  <p></p>
  <p></p>
</div>
```

Le sélecteur `div p:nth-child(2);` se lit « le deuxième élément enfant de `<div>` à condition qu'il s'agisse d'un paragraphe ». Or, dans notre exemple, le deuxième enfant du `<div>` n'est pas un paragraphe mais un élément `<span>` ; aucun élément ne sera donc ciblé.

Le sélecteur `div p:nth-of-type(2);` se lit « le deuxième élément paragraphe enfant de `<div>` ». Le paragraphe succédant au `<span>` sera par conséquent concerné et ciblé.

:only-child

`:only-child` représente un élément qui n'a aucun frère. Il est pris en compte par tous les navigateurs ainsi que sur Internet Explorer 9.

Prenons pour exemple le cas d'une liste dynamique où le nombre d'éléments peut varier. Nous souhaiterions obtenir l'apparence suivante :

- Si la liste comporte plusieurs éléments, alors ceux-ci sont affichés les uns à côté des autres.
- Si la liste ne comporte qu'un seul élément, alors il doit occuper toute la largeur.

Ce comportement se traduirait en langage CSS de cette façon :

```
li {
  display: inline;
  background: orange;
}
li:only-child {
  display: block;
}
```

Dans l'exemple suivant, une image est parfois associée à un paragraphe au sein d'un bloc (figure 8-30). L'affichage des blocs ne comportant pas d'image est géré via ce code CSS :

```
img {
  float: left;
}
p {
  margin: 0 0 0 60px;
}
```

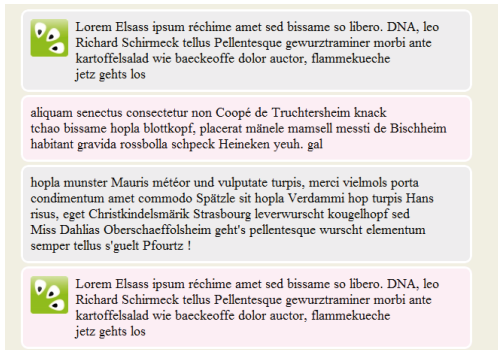
```
p:only-child {
  margin: 0;
}
```

Visualiser l'exemple en ligne

- http://www.ie7nomore.com/fun/only-child/

Figure 8-30

Illustration de :only-child



:only-of-type

La pseudo-classe `:only-of-type` renvoie tous les éléments qui sont seuls de leur type parmi leurs frères. Par exemple, une cellule de tableau `<td>` unique au sein de sa ligne est concernée par le sélecteur `td:only-of-type`, même si un élément frère `<th>` s'y trouve également :

```
<tr>
  <th>titre</th>
  <td>contenu</td>
</tr>
```

Pour ce qui est de sa compatibilité, comme de coutume, cet élément est reconnu par tous les navigateurs actuels, sauf Internet Explorer avant sa version 9.

:first-of-type et :last-of-type

`:first-of-type` désigne le premier élément de son type parmi ses frères, à l'inverse de `:last-of-type`.

Par exemple, le sélecteur suivant représente la dernière cellule de données `<td>` d'une ligne d'un tableau :

```
tr > td:last-of-type {
  text-align: right;
}
```

Ces sélecteurs sont interprétés à partir de Firefox 3.5, Chrome 3, Opera 9.5, Safari 3 et sur Internet Explorer 9.

:enabled, :disabled et :checked

Les spécifications CSS 3 prévoient de pouvoir mettre en forme les éléments de formulaires selon leur contexte. Ainsi, `:enabled` désigne les éléments actifs, `:disabled` les éléments inactifs et `:checked` les éléments cochés.

L'exemple qui suit modifie l'affichage de l'étiquette (`<label>`) associée à un champ lorsque celui-ci est coché par l'utilisateur (figure 8-31) :

Partie HTML

```
<input type="checkbox" id="kiwi" value="oui" />
<label for="kiwi">Vous aimez les kiwis</span>
```

Partie CSS

```
:checked + label {
  color: green;
  font-weight: bold;
}
```

Figure 8-31

Illustration de :checked

ASTUCE Une taille pour les boutons checkbox ou radio

Saviez-vous qu'il est parfaitement possible d'affecter une dimension aux éléments de type checkbox ou radio ?

Il suffit pour cela d'indiquer une valeur pour chacune des composantes `width` et `height`. Ainsi, la valeur commune de `1em` permettra aux boutons de s'agrandir en harmonie avec le reste du contenu :

```
input[type="radio"], input[type="checkbox"] {
    height: 1em;
    width: 1em;
}
```

:required et :optional

Un champ est considéré comme requis s'il est nécessaire qu'il contienne une valeur lors de la soumission du formulaire auquel il est attaché. À l'inverse, `:optional` désigne un champ qui peut demeurer vierge.

:valid, :invalid

Un élément de formulaire est valide s'il remplit toutes les exigences liées à la sémantique et aux spécifications de son type (texte, nombre, adresse électronique, intervalle, URL, etc.). Par exemple, un champ `<input>` de type `url` dont la valeur introduite par l'utilisateur ne débute pas par la chaîne `"http://"` est considéré comme invalide (figure 8-32).

Ces pseudo-classes sont reconnues sur WebKit (Chrome et Safari), ainsi que sur Internet Explorer 9 et sur Firefox, mais uniquement à partir de la version 4.

Vous pouvez par exemple décider d'attribuer une couleur de fond différente aux éléments invalides :

```
input:invalid {
  background-color: red;
}
input:valid {
  background-color: green;
}
```

Figure 8-32

Illustration d'un champ invalide

::selection

Le sélecteur `::selection` n'est pas une pseudo-classe mais un pseudo-élément, d'où la syntaxe débutant par un double « : ». Son usage est simple : il permet de modifier la couleur de fond ou de texte d'une portion de contenu que le visiteur sélectionne en glissant la souris par-dessus.

Ce sélecteur a été introduit dans CSS 3... mais en a été (temporairement ?) retiré il y a quelque temps, bien qu'il fonctionne parfaitement sur WebKit, Opera et Firefox. En voici un exemple :

```
::selection {
  background: yellow; /* Safari, Chrome, Opera */
}
::-moz-selection {
  background: yellow; /* Firefox */
}
```

Non reconnu par – faut-il le préciser ? – Internet Explorer, nul ne peut deviner aujourd'hui si ce pseudo-élément réintégrera les spécifications CSS un jour.

:contains

Tout comme le regretté élément `::selection`, la pseudo-classe `:contains()` fut annoncée puis retirée de CSS 3 malgré son attrait : elle s'appliquait à un élément pour peu qu'il contienne la chaîne de caractères explicitée dans les parenthèses.

Dans ma grande mégalomanie, je pourrais ainsi mettre en exergue tous les commentaires reçus des visiteurs d'Alsacréations qui mentionneraient le site en oubliant le « s » final :

```
:contains("alsacreation ") {
  background: red;
}
```

À ma connaissance, ce sélecteur n'est implémenté sur aucun navigateur actuel, mais le concept n'en demeure pas moins intéressant.

Exercice pratique : tableau de données

L'objectif de cet exercice est de styler un tableau de données en CSS 2 et CSS 3 en évitant tout élément superflu (classe, `id`, attribut) dans le code HTML, qui devra demeurer le plus épuré possible.

Vous observerez au sein de la structure ci-après (figure 8-33) – outre les éléments classiques d'un tableau – un attribut malheureusement sous-exploité : `scope`. Celui-ci précise si l'en-tête de cellule (`<th>`) doit être associé à sa ligne ou à sa colonne.

Partie HTML

```
<table summary="Le prix imaginaire de certains fruits sur le marché">
  <caption>Prix des fruits</caption>
  <tr>
    <td></td>
    <th scope="col">Origine</th>
    <th scope="col">Prix</th>
  </tr>
  <tr>
    <th scope="row">Kiwi</th>
    <td>France</td>
    <td>42€</td>
  </tr>
  <tr>
    <th scope="row">Pastèque</th>
    <td>Taïwan</td>
    <td>1337€</td>
  </tr>
  <tr>
    <th scope="row">Papaye</th>
    <td>Roumanie</td>
    <td>1.3€</td>
  </tr>
  <tr>
    <th scope="row">Melon</th>
    <td>France</td>
    <td>12€</td>
  </tr>
  <tr>
```

```
    <th scope="row">Litchi</th>
    <td>Groënland</td>
    <td>15.50€</td>
  </tr>
</table>
```

Figure 8-33

Le rendu par défaut du tableau

Prix des fruits

	Origine	Prix
Kiwi	France	42€
Pastèque	Taïwan	1337€
Papaye	Roumanie	1.3€
Melon	France	12€
Litchi	Groënland	15.50€

Commençons par appliquer une mise en forme de base pour rendre ce tableau de données plus attrayant : tout d'abord, fixons une largeur de 25 em au tableau et séparons toutes les cellules de données de 2 pixels entre elles à l'aide des propriétés `border-collapse` et `border-spacing` :

```
table {
  width: 25em;
  border-collapse: separate;
  border-spacing: 2px;
}
```

Notre mission suivante est d'appliquer des marges internes de 6 px et 12 px sur chacun des éléments `<td>` et `<th>`. Puis, les cellules d'en-tête devront être alignées à gauche et tous les contenus de la dernière colonne (prix) devront être alignés à droite.

```
td, th {
  padding: 6px 12px;
}
th {
  text-align: left;
}
td:last-child, th:last-child { /* :last-child non compris par  IE8 */
  text-align: right;
}
```

N'oublions pas de décorer la légende du tableau (`<caption>`) grâce aux propriétés de couleurs, d'italique et de bordures :

```
table caption {
  color: #555;
  font-style: italic;
  padding: 10px;
  border-bottom: 1px solid #ccc;
}
```

Allons même plus loin en positionnant la légende en bas des contenus, très simplement en appliquant la propriété `caption-side` au tableau (reconnue à partir d'IE8) :

```
table {
  width: 25em;
  border-collapse: separate;
  border-spacing: 2px;
  caption-side: bottom;
}
```

Nouvelle étape, les rangées paires doivent être de couleur `#eee` et les rangées impaires de couleur `#ddd`. Pour ce faire, nous emploierons le sélecteur CSS 3 `:nth-child` reconnu par tous les navigateurs à l'exception d'Internet Explorer 8 :

```
tr:nth-child(odd) {
  background: #eee;
}
tr:nth-child(even) {
  background: #ddd;
}
```

La première rangée, contenant les en-têtes « origine » et « prix », doit être de couleur `#cba` et son texte doit être blanc :

```
tr:first-child {
  background: #cba;
  color: white;
}
```

Pour finir en beauté ce traitement esthétique, interdisons à la première cellule, vide, d'afficher un arrière-plan (figure 8-34). Plus précisément, la couleur de fond devra être blanche comme le fond de la page :

```
table td:empty {
  background: white;
}
```

Figure 8-34

Mise en forme du tableau

	Origine	Prix
Kiwi	France	42€
Pastèque	Taïwan	1337€
Papaye	Roumanie	1.3€
Melon	France	12€
Litchi	Groënland	15.50€

Prix des fruits

Définissons à présent un comportement spécifique lors du survol avec la souris :

- Les rangées doivent afficher une couleur d'arrière-plan `#bbb` et un texte de couleur blanche.
- Chaque cellule survolée doit avoir une couleur de fond `#999`.
- La première rangée (et aucune de ses cellules) ne doit pas changer d'arrière-plan au survol.

L'exercice nécessite de distinguer la première ligne des autres lors du survol. Le sélecteur d'adjacence direct (`tr + tr`) permet d'appliquer des propriétés à toutes les rangées sauf à la première. C'est exactement ce que nous allons mettre en pratique (figure 8-35) :

```
table td:empty, table td:empty:hover {
  background: white;
}
table tr + tr:hover {
  background: #bbb;
  color:white;
}
table tr + tr td:hover, table tr + tr th:hover {
  background: #777;
}
```

Au terme de ce travail pratique, le tableau de données et la plupart des effets appliqués sont reconnus par l'ensemble des navigateurs, même par Internet Explorer 7 à l'exception de deux comportements : la gestion des lignes paires et impaires (`:nth-child`) et le positionnement de la légende (`caption-side`). Rien de vraiment bloquant, n'est-ce pas ?

Figure 8-35

Effets de survol des cellules

	Origine	Prix
Kiwi	France	42€
Pastèque	Taïwan	1337€
Papaye	Roumanie	1.3€
Melon	France	12€
Litchi	Groënland	15.50€

Prix des fruits

Visualiser le résultat en ligne

▸ http://www.ie7nomore.com/fun/tablepimp/