2. Sélecteurs

Un sélecteur W3C représente une structure. Cette structure peut être vue par exemple comme une condition (p.ex. dans une règle CSS) qui détermine quels éléments de l'arbre documentaire sont sélectionnés par le sélecteur, ou comme une description plate du fragment HTML ou XML correspondant à cette structure.

Les sélecteurs W3C vont de la simple représentation du nom d'un élément jusqu'à des représentations contextuelles complexes.

La table qui suit résume la syntaxe des sélecteurs W3C :

Séquence	Signification	Décrit en section	Défini originellement en CSS niveau
*	tout élément	Sélecteur universel	2
E	tout élément de type E	Sélecteur de type d'élément	1
E[foo]	tout élément E portant l'attribut "foo"	Sélecteurs d'attribut	2
E[foo="bar"]	tout élément E portant l'attribut" foo" et dont la valeur de cet attribut est exactement "bar"	Sélecteurs d'attribut	2
E[foo~="bar"]	tout élément E dont l'attribut "foo" contient une liste de valeurs séparées par des espaces, l'une de ces valeurs étant exactement égale à "bar"	Sélecteurs d'attribut	2
E[foo^="bar"]	tout élément E dont la valeur de l'attribut "foo" commence exactement par la chaîne "bar"	Sélecteurs d'attribut	3
E[foo$="bar"]	tout élément E dont la valeur de l'attribut "foo" finit exactement par la chaîne "bar"	Sélecteurs d'attribut	3
E[foo*="bar"]	tout élément E dont la valeur de l'attribut "foo" contient la sous-chaîne "bar"	Sélecteurs d'attribut	3
E[lang|="en"]	tout élément E dont l'attribut 'lang" est une liste de valeurs séparées par des tirets et commençant (à gauche) par "en"	Sélecteurs d'attribut	2

E:root	un élément E, racine du document	Pseudo-classes structurelles	3
E:nth-child(n)	un élément E qui est le n-ième enfant de son parent	Pseudo-classes structurelles	3
E:nth-last-child(n)	un élément E qui est le n-ième enfant de son parent en comptant depuis le dernier enfant	Pseudo-classes structurelles	3
E:nth-of-type(n)	un élément E qui est le n-ième enfant de son parent et de ce type	Pseudo-classes structurelles	3
E:nth-last-of-type(n)	un élément E qui est le n-ième enfant de son parent et de ce type en comptant depuis le dernier enfant	Pseudo-classes structurelles	3
E:first-child	un élément E, premier enfant de son parent	Pseudo-classes structurelles	2
E:last-child	un élément E, dernier enfant de son parent	Pseudo-classes structurelles	3
E:first-of-type	un élément E, premier enfant de son type	Pseudo-classes structurelles	3
E:last-of-type	un élément E, dernier enfant de son type	Pseudo-classes structurelles	3
E:only-child	un élément E, seul enfant de son parent	Pseudo-classes structurelles	3
E:only-of-type	un élément E, seul enfant de son type	Pseudo-classes structurelles	3
E:empty	un élément E qui n'a aucun enfant (y compris noeuds textuels purs)	Pseudo-classes structurelles	3
E:link E:visited	un élément E qui est la source d'un hyperlien dont la cible n'a pas encore été visitée (:link) ou a déjà été visitée (:visited)	Les pseudo-classes de lien	1
E:active E:hover E:focus	un élément E pendant certaines actions de l'usager	Les pseudo-classes d'action Usager	1 and 2

E:target	un élément E qui est la cible de l'URL d'origine contenant lui-même un fragment identifiant.	La pseudo-classe :target	3
E:lang(c)	un élément E dont le langage (humain) est c (le langage du document spécifie comment le langage humain est déterminé)	La pseudo-classe :lang()	2
E:enabled E:disabled	un élément d'interface utilisateur E qui est actif ou inactif.	Les pseudo-classes d'état d'élément d'interface	3
E:checked E:indeterminate	un élément d'interface utilisateur E qui est coché ou dont l'état est indéterminé (par exemple un bouton-radio ou une case à cocher)	Les pseudo-classes d'état d'élément d'interface	3
E:contains("foo")	un élément E dont le contenu textuel concaténé contient la sous-chaîne "foo"	La pseudo-classe de contenu	3
E::first-line	la première ligne formatée d'un élément E	The :first-line pseudo-element	1
E::first-letter	le premier caractère formaté d'un élément E	Le pseudo-élément ::first-letter	1
E::selection	la partie d'un élément E qui est actuellement sélectionnée/mise en exergue par l'usager	Les pseudo-éléments fragments d'éléments d'interface	3
E::before	le contenu généré avant un élément E	Le pseudo-élément ::before	2
E::after	le contenu généré après un élément E	Le pseudo-élément ::after	2
E.warning	*Uniquement en HTML*. Identique à E[class~="warning"].	Sélecteurs de classe	1
E#myid	un élément E dont l'ID est égal à "myid".	Sélecteurs d'ID	1
E:not(s)	un élément E qui n'est pas représenté par le sélecteur simple s	La pseudo-classe de négation	3
E F	un élément F qui est le descendant d'un élément E	Combinateur de descendance	1
E > F	un élément F qui est le fils d'un	Combinateur filial	2

	élément E		
E + F	un élément F immédiatement précédé par un élément E	Combinateur d'adjacence directe	2
E ~ F	un élément F précédé par un élément E	Combinateur d'adjacence indirecte	3

Exemple : en CSS, la signification réelle de chaque sélecteur est déterminée à partir de la table ci-dessus en faisant précéder le mot "sélectionne" au contenu de chaque cellule de la colonne "Signification".

3. Sensibilité à la casse

La sensibilité à la casse des noms d'éléments d'un langage de documents donné dans les sélecteurs W3C dépend du langage de documents en question. Par exemple, en HTML, les noms d'éléments ne sont pas senbibles à la casse, alors qu'ils le sont en XML

La sensibilité à la casse des noms d'attribus et des valeurs d'attributs dépend également du langage de documents.

4. Syntaxe des sélecteurs

Un sélecteur est une chaîne d'une ou plusieurs séquences de sélecteurs simples séparés par des combinateurs.

Une séquence de sélecteurs simples est une chaîne de sélecteurs simples qui ne sont pas séparés par des combinateurs. Ele commence toujours par un sélecteur de type d'élément ou un sélecteur universel. Aucun autre sélecteur de type d'élément ou sélecteur universel n'est autorisé dans la séquence.

Un sélecteur simple est soit un sélecteur de type d'élément, un sélecteur universel, un sélecteur d'attribut, un sélecteur d'ID, un sélecteur de contenu ou une pseudo-classe. Un pseudo-élément peut suivre la dernière séquence de sélecteurs simples.

Les combinateurs sont : l'espace, ">", "+" et "~". Un espace peut être présent entre un combinateur et les sélecteurs simples qui l'entourent. Seuls les caractères "espace" (code Unicode 32), "tab" (9), "saut de ligne" (10), "retour chariot" (13), and "saut de page" (12) peuvent composer un espace. Les autres caractères de type espace, comme "em-space" (8195) et "ideographic space" (12288), ne peuvent composer un espace.

5. Groupes de sélecteurs

Quand plusieurs sélecteurs sont associés aux mêmes déclarations, ils peuvent être groupés en les séparant par des virgules.

Exemples CSS :

Dans cette exemple, les trois règles CSS sont réduites à une seule :

```
h1 { font-family: sans-serif }
h2 { font-family: sans-serif }
h3 { font-family: sans-serif }
```

est équivalent à :

```
h1, h2, h3 { font-family: sans-serif }
```

Attention : cette équivalence est vérifiée dans cet exemple car les trois sélecteurs initiaux sont des sélecteurs valides. Si l'un ou plus de ces sélecteurs est invalide, le groupe composé des trois sélecteurs est lui-même invalide, invalidant l'ensemble de la règle ; il n'y a alors pas équivalence.

6. Sélecteurs simples

6.1 Sélecteur de type d'élément

Un sélecteur de type d'élément est le nom d'un type d'élement du langage de documents. Un sélecteur de type d'élement représente une occurence d'un élément du type donné dans l'arbre du document.

Exemple(s):

Le sélecteur suivant représente un élément `h1` dans l'arbre du document :

```
h1
```

6.2 Sélecteur universel

Le sélecteur universel, dénoté"`*`", represente le nom qualifié de tout type d'élément. Il représente donc tout élément de l'arbre du document dans tout espace de noms (y compris les éléments sans espace de noms déclaré) si aucune espace de noms par défaut n'a été déclaré. Si un espace de noms par défaut a été déclaré, consultez ci-dessous Sélecteur universel et Espaces de noms.

Si le sélecteur universel n'est pas le seul composant d'une séquence de sélecteurs simples, le `*` peut être omis. Par exemple:

- `*[LANG=fr]` et `[LANG=fr]` sont équivalents,
- `*.warning` et `.warning` sont équivalents,
- `*#myid` et `#myid` sont équivalents.

Attention : il est recommandé de ne pas omettre le `*`, representant le sélecteur universel.

6.3 Sélecteurs d'attribut

Les sélecteurs W3C permettent de représenter les attributs attachés à un élément.

6.3.1 Représentation des attributs et des valeurs d'attributs

Quatre différents sélecteurs d'attribut sont disponibles :

`[att]`
: Représente l'attribut `att`, quelque soit la valeur de l'attribut.

`[att=val]`
: Représente l'attribut `att`, sa valeur étant exactement égale à "val".

`[att~=val]`
: Représente l'attribut `att`, sa valeur étant une liste de mots séparés par des espaces, l'un de ces mots étant exactement "val". En cas d'usage de ce sélecteur, les mots dans la valeur ne doivent pas être séparés par des espaces (puisqu'ils sont séparés par des espaces).

`[att|=val]`
: Représente l'attribut `att`, sa valeur étant une liste de mots séparés par des tirets et commençant par "val". La comparaison se fait toujours au début de la valeur de l'attribut. Ceci est principalement destiné à la sélection de sous-codes linguistiques (p.ex. l'attribut `LANG` en HTML) conformément à la RFC 1766 ([RFC1766]).

Les valeurs d'attributs doivent être des identificateurs ou des chaînes de caractères. La sensibilité à la casse des noms d'attributs et de leurs valeurs dépend du langage de documents.

Exemple(s):

Par exemple, le sélecteur suivant représente un élément `h1` portant l'attribut `title`, quelque soit sa valeur:

```
h1[title]
```

Exemple(s):

Dans l'exemple suivant, le sélecteur représente un élément `span` dont l'attribut `class` a exactement la veleur "exemple":

```
span[class=exemple]
```

Plusieurs sélecteurs d'attribut peuvent être utilisés pour représenter plusieurs attributs d'un élément, ou plusieurs fois le même attribut.

Exemple(s):

Ici, le sélecteur représente un élément `span` dont l'attribut `hello` a exactement la valeur "Cleveland" et dont l'attribut `goodbye` a exactement la valeur "Columbus":

```
span[hello="Cleveland"][goodbye="Columbus"]
```

Exemple(s):

Les sélecteurs suivants illustrent les différences entre "=" et "~=". Le premier sélecteur représentera, par exemple, la valeur "copyright copyleft copyeditor" pour l'attribut `rel`. Le second sélecteur représentera uniquement un élément `a` dont l'attribut `href` a exactement la veleur "http://www.w3.org/".

```
a[rel~="copyright"]
a[href="http://www.w3.org/"]
```

Exemple(s):

Le sélecteur suivant représente un élément quelconque dont l'attribut `lang` a la valeur "fr" (p.ex. dont la langue est le Français).

```
*[lang=fr]
```

Exemple(s):

Le sélecteur qui suit représente un élément quelconque dont l'attribut `lang` a une valeur commençant par "en", par exemple "en", "en-US" ou "en-cockney":

```
*[lang|="en"]
```

Exemple(s):

De même, les sélecteurs qui suivent représentent un élément `DIALOGUE` ayant deux valeurs différentes pour le même attribut `character`:

```
DIALOGUE[character=romeo]

DIALOGUE[character=juliet]
```

6.3.2 Sélecteurs de sous-chaîne dans la valeur d'un attribut

Trois différents sélecteurs d'attribut sont disponibles pour la représentation d'une séquence dans la valeur d'un attribut :

`[att^="val"]`
: Représente l'attribut `att`, sa valeur commençant exactement par le préfixe "val"

`[att$=ident]`
: Représente l'attribut `att`, sa valeur finissant exactement par le préfixe "ident"

`[att*="val"]`
: Représente l'attribut `att`, sa valeur contenant au moins une fois la sous-chaîne "val"

Les valeurs d'attributs doivent être des identificateurs ou des chaîne de caractères. La sensibilité à la casse des noms d'attributs et de leurs valeurs dépend du langage de documents.

Exemple(s) :

Le sélecteur suivant représente un élément HTML `object`, ciblant une image.

```
object[type^="image/"]
```

Le sélecteur suivant représente un élément XML `foo1` portant l'attribut `bar`, la valeur de cet attribut finissant par "cpg".

```
foo1[bar$="cpg"]
```

Le sélecteur suivant représente un paragraphe HTML dont la valeur de l'attribut `title` contient la sous-chaîne "hello".

```
p[title*="hello"]
```

6.4 Sélecteurs de classe

En HTML, les auteurs peuvent utiliser la notation point (`.`) comme alternative à la notation `~=` notation dans le cas de l'attribut `class`. En HTML, `div.value` et `div[class~=value]` ont donc la même signification. La valeur de l'attribut doit suivre immédiatement le "`.`".

Exemple(s):

On pourra représenter un élément quelconque portant `class~="pastoral"` de la manière suivante:

```
*.pastoral
```

ou tout simplement

```
.pastoral
```

Le sélecteur suivant représente un élément `h1` portant `class~="pastoral"`:

```
h1.pastoral
```

Exemple(s):

Le sélecteur suivant représente un élément `p` dont l'attribut `class` contient une liste de valeurs séparées par des espaces et contenant "pastoral" et "marine":

```
p.pastoral.marine
```

Il est absolument identique à :

```
p.marine.pastoral
```

Ce sélecteur représente par exemple un élément `p` avec `class="pastoral blue aqua marine"` ou `class="marine blue pastoral aqua"` mais pas avec `class="pastoral blue"`.

6.5 Sélecteurs d'ID

Les langages de documents peuvent contenir des attributs qui sont déclarés être de type ID. Les attributs de type ID sont spéciaux en ce sens que deux tels attributs ne peuvent pas avoir la même valeur dans un document, sans considération du type des éléments qui les portent ; quelque soit le

langage, un attribut `ID` peut être utilisé pour identifier de façon unique les éléments. En HTML, tous les attributs ID sont nommés "id" ; les applications XML peuvent nommer les attributs ID différemment, mais avec les mêmes contraintes.

L'attribut ID d'un langage de documents permet aux auteurs d'assigner un identificateur à une instance d'élément dans l'arbre documentaire. Les sélecteurs W3C d'ID représentent une instance d'élément en se basant sur son identificateur. Un sélecteur d'ID contient un "#" immédiatement suivi de la valeur d'ID.

Exemple(s):

Le sélecteur d'ID qui suit représente un élément `h1` dont l'attribut `id` a la valeur "chapter1" :

```
h1#chapter1
```

Le sélecteur d'ID ci-dessous représente tout élément dont l'attribut `id` a la valeur "chapter1" :

```
#chapter1
```

Le sélecteur d'ID suivant représente tout élément dont l'ID a la valeur "z98y".

```
*#z98y
```

.

6.6 Pseudo-classes

Le concept de pseudo-classe est introduit pour permettre une sélection basée sur des informations qui résident à l'extérieur de l'arbre documentaire ou qui ne peuvent pas être exprimées à l'aide des autres sélecteurs simples.

Une pseudo-classe contient toujours deux-points (`:`) suivis du nom de la pseudo-classe et optionnellement d'une valeur entre parenthèses.

Les pseudo-classes sont autorisées dans toutes les séquences de sélecteurs simples contenues dans un sélecteur. Les pseudo-classes sont autorisées partout dans une séquence de sélecteurs simples, après le sélecteur de type ou le sélecteur universel (éventuellement omis) initial. Les noms des pseudo-classes sont insensibles à la casse. Quelques pseudo-classes sont mutuellement exclusives, alors que d'autres peuvent être appliquées simultanément au même élément. Des pseudo-classes peuvent être dynamiques, en ce sens qu'un élément peut acquérir ou perdre une pseudo-classe lorsqu'un usager interagit avec le document.

6.6.1 Pseudo-classes dynamiques

Les pseudo-classes dynamiques classent les éléments selon des caractéristiques autres que le nom, les attributs ou le contenu, caractéristiques qui en principe ne peuvent pas être déduites de l'arbre documentaire.

Les pseudo-classes dynamiques n'apparaissent pas dans le source du document ni dans l'arbre documentaire.

Les pseudo-classes de liens : :link et :visited

Les Agents Utilisateurs restituent les liens non visités différemment des liens visités. Les Sélecteurs W3C offrent les pseudo-classes `:link` et `:visited` pour les distinguer :

- La pseudo-classe `:link` caractérise les liens qui n'ont pas encore été visité,.
- La pseudo-classe `:visited` s'applique une fois que le lien a été visité par l'usager.

Note. *Après un certain délai, les Agents Utilisateurs peuvent choisir de ramener un lien à l'état (non visité) ':link'.*

Les deux états sont mutuellement exclusives.

Exemple(s):

Le sélecteur suivant représente des liens portant la classe `external` et déjà visités :

```
a.external:visited
```

Les pseudo-classes d'action usager :hover, :active, and :focus

Les Agents Utilisateurs interactifs changent parfois la restitution en réponse à des actions de l'usager. Les Sélecteurs W3C offrent trois pseudo-classes pour la sélection d'un élément sur lequel l'usager agit.

- La pseudo-classe `:hover` s'applique quand l'utilisateur désigne un élément (avec un périphérique de pointage), mais ne l'active pas. Par exemple, un Agent Utilisateur visuel pourrait appliquer cette pseudo-classe quand le curseur (le pointeur de la souris) se trouve au-dessus d'une boîte générée par l'élément. Les Agents Utilisateurs ne supportant pas les media interactifs n'ont pas à supporter cette pseudo-classe. Certains Agents Utilisateurs conformes et supportant les media interactifs peuvent ne pas être capable de supporter cette pseudo-classe (par exemple, un stylet de pointage).
- La pseudo-classe `:active` s'applique pendant qu'un élément est activé par l'utilisateur. Par exemple, entre le moment où l'utilisateur appuie sur le bouton de la souris et celui où il le relache.
- La pseudo-classe `:focus` s'applique pendant qu'un élément a le focus (accepte les évènements clavier ou souris, ou d'autres formes d'entrée).

Seuls les éléments dont la propriété 'user-input' (Cf. [UI]) a pour valeur "enabled" peuvent devenir `:active` ou acquérir `:focus`.

Ces pseudo-classes ne sont pas mutuellement exclusives. Un élément peut déclencher plusieurs d'entre elles en même temps.

Exemple(s):

```
a:link /* liens non visites */
a:visited /* liens visites   */
a:hover /* survol        */
a:active /* liens actifs   */
```

Exemple(s):

Un exemple de pseudo-classes dynamiques combinées :

```
a:focus
a:focus:hover
```

Le dernier sélecteur représente les éléments `A` qui ont la pseudo-classe :focus et la pseudo-classe :hover.

Note. *Un élément peut être à la fois ':visited' et ':active' (ou ':link' et ':active').*

6.6.2 La pseudo-classe :target

Certains URIs se réfèrent à un point bien déterminé dans une ressource. Ce type d'URI se termine par "#" suivi par un identificateur d'ancre (appelé l'identificateur de fragment).

Les URIs avec un identificateur de fragment ciblent un élément donné dans le document, appelé élément cible. Par exemple, voici un URI ciblant une ancre nommée section_2 dans un document HTML :

```
http://somesite.com/html/top.html#section_2
```

Un élément cible peut être représenté par la pseudo-classe `:target` :

```
p.bar:target
```

représente un élément `p` de classe bar qui est la cible deURI.

Exemple CSS d'usage de la pseudo-classe `:target` :

```
*:target { color : red }

*:target:before { content : url(target.png) }
```

6.6.3 La pseudo-classe :lang

Si le langage de documents spécifie comment est déterminée la langue humaine d'un élément, il est possible d'écrire des sélecteurs qui représentent un élément sur la base de sa langue. Par exemple, en HTML [HTML40], la langue est déterminée par la combinaison de l'attribut `lang`, de l'élément `meta`, et potentiellement d'informations venant du protocole (comme les en-têtes HTTP). XML utilise un attribut appelé `XML:LANG`, et il peut y avoir d'autres méthodes spécifiques au langage de documents pour déterminer la langue.

La pseudo-classe `:lang(C)` représente un élément dans la langue C. C est ici un code linguistique comme spécifié par HTML 4.0 [HTML40] et RFC 1766 [RFC1766]. Il est sélectionné de la même manière que par le sélecteur d'attribut linguistique `[att |= "C"]`.

Exemple(s):

Les deux premiers sélecteurs ci-dessous représentent un document HTML qui est en Français ou en Allemand. Les deux sélecteurs suivants représentent des citations `q` contenues dans un élément quelconque dont la langue est le Français ou l'Allemand.

```
html:lang(fr)
html:lang(de)
:lang(fr) > q
:lang(de) > q
```

6.6.4 Les pseudo-classes d'état d'élément d'inteface

Les pseudo-classes :enabled et :disabled

La pseudo-classe `:enabled` permet aux auteurs de modifier le style des éléments d'interface qui sont actifs - que l'utilisateur peut sélectionner/activer d'une manière ou d'une autre (par exemple en cliquant sur un bouton avec la souris). Une telle pseudo-classe est utile parce qu'il est autrement impossible de spécifier programmatiquement l'apparence par défaut de disons un élément `input` actif sans spécifier également son allure s'il est inactif.

Parallèlement à `:enabled`, `:disabled` permet aux auteurs de spécifier exactement comment un élément d'interface inactif ou inactivé doit être restitué.

Notez que la plupart des éléments ne sont ni actifs ni inactifs. Un élément est actif si l'utilisateur peut l'activer ou lui transférer le focus. Un élément est inactif s'il peut être actif, mais que l'utilisateur ne peut à ce moment précis l'activer ou lui transférer le focus.

La pseudo-classe :checked

La pseudo-classe `:checked` ne s'applique aux éléments qui ont 'user-input: enabled' ou 'user-input : disabled' (voyez [UI] à propos de la propriété 'user-input'). L'utilisateur peut changer l'état des éléments radio et cases à cocher. Quelques entrées de menu sont "cochées" quand l'utilisateur les sélectionne. Quand de tels éléments passent dans l'état "allumé", la pseudo-classe `:checked` s'applique. La pseudo-classe `:checked` s'applique initialement à de tels éléments qui possèdent l'attribut HTML4 `selected` tels que décrits dans la Section 17.2.1 de HTML4, mais bien entendu l'utilisateur peut faire passer à "éteint" ces éléments auquel car la pseudo-classe `:checked` ne s'applique plus. Alors que la pseudo-classe `:checked` est dynamique par nature, et est modifiée par l'action de l'utilisateur, elle s'applique à tous les media puisqu'elle peut aussi se fonder sur la présence de l'attribut sémantique HTML4 `selected`.

La pseudo-classe :indeterminate

La pseudo-classe `:indeterminate` s'applique aux éléments qui ont 'user-input: enabled' ou 'user-input: disabled' (voyez [UI] à propos de la propriété 'user-input'). Les éléments radio et cases à cocher peuvent être changés d'état par l'utilisateur, mais sont parfois dans un état indéterminé, ni coché ni non-coché. Cela peut être causé par un attribut de l'élément, ou par une manipulation du DOM. La pseudo-classe `:indeterminate` s'applique à de tels éléments. Alors que la pseudo-classe `:indeterminate` est dynamique par nature, et est modifiée par l'action de l'utilisateur, elle s'applique à tous les media puisqu'elle peut aussi se fonder sur la présence de l'attribut sémantique HTML4 `selected`.

Les composants d'un radio-groupe sans choix pré-sélectionné sont un exemple d'état `:indeterminate`.

6.6.5 Les pseudo-classes structurelles

Les Sélécteurs W3C introduisent la notion de pseudo-classes structurelles qui permettent la sélection sur la base d'informations supplémentaires se trouvant dans l'arbre documentaire mais qui ne peuvent être représentées par d'autres sélecteurs simples ou par des combinateurs.

Veuillez noter que le texte seul PCDATA n'est pas compté lors du calcul de la position d'un élément dans la liste des enfants de son parent. Lors du calcul de la position d'un élément dans la liste des enfants de son parent, la numérotation commence à 1.

La pseudo-classe :root

La pseudo-classe `:root` représente un élément qui est la racine du document. En HTML 4, il s'agit de l'élément `html`. En XML, c'est ce qui est spécifié par la DTD, le schéma, l'espace de noms pour le document XML en question.

La pseudo-classe :first-child

Identique à `:nth-child(1)`. La pseudo-classe `:first-child` représente un élément qui est le premier fils d'un autre élément.

Exemple(s):

Dans l'exemple qui suit, le sélecteur représente un élément `p` qui est le premier fils d'un élément `div`.

```
div > p:first-child
```

Ce sélecteur peut représenter un `p` à l'intérieur d'un `div` du fragment suivant ::

```
<p> The last P before the note.</p>
<div class="note">
 <p> The first P inside the note.</p>
</div>
```

mais ne peut pas représenter le second `p` du fragment suivant :

```
<p> The last P before the note.</p>
<div class="note">
 <h2>Note</h2>
 <p> The first P inside the note.</p>
</div>
```

Les deux sélecteurs qui suivent sont équivalents :

```
* > a:first-child /* a is first child of any element */
a:first-child /* Same */
```

La pseudo-classe :last-child

Identique à `:nth-last-child(1)`. La pseudo-classe `:last-child` pseudo-class représente un élément qui est le dernier fils d'un autre élément.

Exemple:

Le sélecteur suivant représente une entrée de liste `li` qui est le dernier enfant d'une liste ordonnée `ol`.

```
ol li:last-child
```

La pseudo-classe :first-of-type

Identique à `:nth-of-type(1)`. La pseudo-classe `:first-of-type` pseudo-class représente un élément qui est le premier enfant de son type dans la liste des enfants de son élément parent.

Exemple:

Le sélecteur qui suit représente un titre de définition `dt` à l'intérieur d'une liste de définitions `dl`, cette `dt` étant le premier enfant de son type dans la liste des enfants de l'élément parent.

```
dl dt:first-of-type
```

C'est une description valide pour les deux premiers `dt` dans l'exemple qui suit mais pas pour le troisième ::

```
<dl><dt>gigogne</dt>
 <dd><dl><dt>fus&eacute;e</dt>
 <dd>multistage rocket</dd>
 <dt>table</dt>
 <dd>nest of tables</dd>
 </dl></dd>
</dl>
```

La pseudo-classe :last-of-type

Identique à `:nth-last-of-type(1)`. La pseudo-classe `:last-of-type` pseudo-class représente un élément qui est le premier enfant de son type dans la liste des enfants de son élément parent.

Exemple:

Le sélecteur suivant représente la dernière cellule de données `td` d'une ligne de tableau..

```
tr > td:last-of-type
```

La pseudo-classe :only-child

Représente un élément qui n'a aucun frère. Identique à `:first-child:last-child` ou `:nth-child(1):nth-last-child(1)`, mais avec une spécificité inférieure.

La pseudo-classe :only-of-type

Représente un élément qui n'a pas de frère avec le même nom d'élément. Identique à `:first-of-type:last-of-type` or `:nth-of-type(1):nth-last-of-type(1)`, mais avec une spécificité inférieure.

:empty pseudo-class

La pseudo-classe `empty` représente un élément qui n'a aucun enfant, y compris les noeuds textuels.

Exemples:

`p:empty` est une représentation valide du fragment ci-dessous :

```
<p></p>
```

`foo:empty` n'est pas une représentation valide des fragments ci-dessous :

```
<foo>bar</foo>

<foo><bar>bla</bar></foo>

<foo>this is not <bar>:empty</bar></foo>
```

6.6.6 La pseudo-classe de contenu

La notation de pseudo-classe `:contains("foo")` représente un élément dont le contenu textuel contient la sous-chaîne donnée. L'argument de cette pseudo-classe peut être une chaîne de caractères (entourée par des guillemets) ou un mot-clé.

L'usage de la pseudo-classe de contenu est restreint aux media statiques.

Le contenu textuel d'un élément donné est déterminé par la concaténation de tous les PCDATA contenus dans l'élément et ses sous-éléments.

Exemple:

```
p:contains("Markup")
```

est une description correcte et valide, mais partielle, de :

```
<p><strong>H</strong>yper<strong>t</strong>ext
 <strong>M</strong><em>arkup</em>
 <strong>L</strong>anguage</p>
```

Des caracatères spéciaux peuvent être insérés dans l'argument de la pseudo-classe de contenu par usage du mécanisme d'échappement pour les caractères Unicode et caractères spéciaux.

***Note** : `:contains()` est une pseudo-classe, pas un pseudo-élément. La règle CSS suivante*

appliquée au fragment HTML ci-dessus n'ajoutera pas un fond rouge au seul mot "Markup" mais ajoutera ce fond à tout le paragraphe.

```
P:contains("Markup") { background-color : red }
```

6.6.7 La pseudo-classe de negation

La pseudo-classe de négation est une notation fonctionnelle prenant un sélecteur simple (à l'exclusion de la pseudo-classe de négation elle-même) pour argument. Elle représente un élément qui n'est pas représenté par l'argument.

Exemple(s) :

Le sélecteur CSS suivant sélectionne tous les éléments `button` qui ne sont pas désactivés dans un document HTML :

```
button:not([DISABLED])
```

Le sélecteur suivant représente tous les éléments sauf les éléments `FOO` :

```
*:not(FOO)
```

Le groupe de sélecteurs suivant représente tous les éléments sauf les liens HTML :

```
html|*:not(:link):not(:visited)
```

Note: la pseudo-classe `:not()` permet d'écrire des sélecteurs inutiles. Par exemple, `:not(*|*)`, qui ne représente aucun élément du tout, or `foo:not(bar)`, qui est équivalent à `foo` mais avec une spécificité supérieure.

7. Pseudo-éléments

Les pseudo-éléments créent des abstractions hors de l'arbre documentaire qui ne sont pas spécifiables par le langage de documents. Par exemple, les langages de documents n'offrent aucun moyen d'accéder à la première lettre ou la première ligne du contenu d'un élément. Les pseudo-éléments permettent aux auteurs de se référer à ces données inaccessibles autrement. Les pseudo-éléments peuvent aussi fournir aux auteurs un moyen de se référer à du contenu qui n'existe pas dans le source du document (par exemple, les pseudo-éléments `:before` et `:after` donnent accés au contenu généré).

Un pseudo-élément est formé de deux deux-points (`::`) suivis du nom du pseudo-élément.

Attention : cette notation `::` est introduite par le présent document dans le but d'établir une discrimination entre les pseudo-classes et les pseudo-éléments. Pour des raisons de compatibilité avec les feuilles de styles existantes, les Agents Utilisateurs doivent aussi accepter la notation précédente avec un seul deux-points. Cepedant cette compatibilité n'est pas requise pour les nouveaux pseudo-élémentsp introduits dans CSS level 3.

Les pseudo-éléments peuvent apparaître une fois seulement dans la séquence de sélecteurs simples représentant les sujets du sélecteur.

Les noms des pseudo-éléments sont insensibles à la casse.

7.1 Le pseudo-élément ::first-line

Le pseudo-élément `::first-line` décrit la première ligne formatée d'un élément.

Par exemple en CSS :

```
p::first-line { text-transform: uppercase }
```

La règle ci-dessus signifie "changer les lettres de la première ligne de chaque paragraphe en majuscules". Cependant, le sélecteur `p::first-line` ne sélectionne aucun élément HTML réel. Il sélectionne un pseudo-élément que les Agents Utilisateurs conformes vont insérer au début de chaque paragraphe.

Veuillez noter que la longueur de la première ligne dépend d'un grand nombre de facteurs, y compris de la largeur de la page, la taille de fonte, etc. Ainsi, dans un paragraphe HTML ordinaire tel que :

```
<p>This is a somewhat long HTML
paragraph that will be broken into several
lines. The first line will be identified
by a fictional tag sequence. The other lines
will be treated as ordinary lines in the
paragraph.</p>
```

les lignes qui seront découpées de la manière suivante :

```
THIS IS A SOMEWHAT LONG HTML PARAGRAPH THAT
will be broken into several lines. The first
line will be identified by a fictional tag
sequence. The other lines will be treated as
ordinary lines in the paragraph.
```

peuvent être "réécrites" par les Agents Utilisateurs pour y inclure la *séquence de balises fictives* pour `::first-line`. Cette séquence de balises fictives aide à montrer comment les propriétés sont héritées.

```
<p><p::first-line> This is a somewhat long HTML
paragraph that will </p::first-line> be broken into several
lines. The first line will be identified
by a fictional tag sequence. The other lines
will be treated as ordinary lines in the
paragraph.</p>
```

Si un pseudo-element scinde un élément réel, l'effet désiré peut souvent être décrit par une séquence de balises fictives qui ferme et ré-ouvre l'élément. Ainsi, si l'on balise le paragraphe précédent avec un élément `span` :

```
<p><span class="test"> This is a somewhat long HTML
paragraph that will be broken into several
lines.</span> The first line will be identified
by a fictional tag sequence. The other lines
will be treated as ordinary lines in the
paragraph.</p>
```

l'Agent Utilisateur pourra générer les balises de début et de fin appropriées pour `span` lors de l'insertion de la séquence de balises fictives pour `::first-line`.

```
<p><p::first-line><span class="test"> This is a
somewhat long HTML
paragraph that will </span></p::first-line><span class="test"> be
broken into several
lines.</span> The first line will be identified
by a fictional tag sequence. The other lines
will be treated as ordinary lines in the
paragraph.</p>
```

Le pseudo-élément `::first-line` ne peut être attaché qu'à des éléments de niveau bloc.

Le pseudo-élément `::first-line` est similaire à un élément de niveau flot de texte, mais avec certaines restrictions, en fonction de l'usage. Seules les propriétés suivantes s'appliquent aux pseudo-éléments `::first-line` : propriétés de fontes, propriétés de couleurs, 'word-spacing', 'letter-spacing', 'text-decoration', 'vertical-align', 'text-transform', 'line-height', 'text-shadow', et 'clear'.

7.2 Le pseudo-élément ::first-letter

Le pseudo-élément `::first-letter` la première lettre formatée d'un élément.

Le pseudo-élément `::first-letter` peut être attaché à tout élément.

Le pseudo-élément `::first-letter` peut être utilisé pour des "initiales en capitales" et des "lettrines", qui sont des effets typographiques usuels. Ce type de lettre initiale est similaire à un élément de niveau flot de texte si sa propriété CSS 'float' a pour valeur 'none', mais avec certaines restrictions, en fonction de l'usage. Autrement, il est similaire à un élément flottant.

Voici les propriétés CSS qui s'appliquent aux pseudo-éléments `::first-letter` : propriétés de fontes, propriétés de couleurs, propriétés de fonds, 'text-decoration', 'vertical-align' (seulement si 'float' est 'none'), 'text-transform', 'line-height', propriétés de marges externes, propriétés de marges internes, propriétés de cadres, 'float', 'text-shadow', et 'clear'.

L'exemple CSS 2 suivant génèrera une initiale "lettrine" courant sur deux lignes :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
 <HEAD>
  <TITLE>Drop cap initial letter</TITLE>
  <STYLE type="text/css">
   P               { font-size: 12pt; line-height: 12pt }
   P::first-letter { font-size: 200%; font-style: italic;
                     font-weight: bold; float: left }
   SPAN            { text-transform: uppercase }
  </STYLE>
 </HEAD>
 <BODY>
  <P><SPAN>The first</SPAN> few words of an article
    in The Economist.</P>
 </BODY>
</HTML>
```

Cet exemple pourra être formaté de la manière qui suit :

La séquence de balises fictives est :

```
<P>
<SPAN>
<P::first-letter>
T
</P::first-letter>he first
</SPAN>
few words of an article in the Economist.
</P>
```

Veuillez noter que les balises du pseudo-élément `::first-letter` entourent le contenu (i.e. le caractère initiale), alors que la balise de début de `::first-line` est insérée immédiatement la balise de début de l'élément auquel il est attaché.

Pour réaliser le formatage traditionnel des "lettrines", les Agents Utilisateurs pourront approximer les tailles de fontes, par exemple pour aligner les lignes de base. De même, le contour des glyphes peut être pris en compte lors du formatage.

Les ponctuations (c'est-à-dire les caractères définis dans Unicode [UNICODE] dans les classes de ponctuation "open" (Ps), "close" (Pe), et "other" (Po)), qui précèdent la première lettre doivent être incluses, comme dans :

Le pseudo-élément `::first-letter` sélectionne des parties d'élément seulement.

Quelques langues ont des règles spécifiques de traitement de certaines combinaisons de lettres. En Néerlandais, par exemple, si la combinaison de lettres "ij" apparait en début de mot, les deux lettres doivent être prises en compte dans le pseudo-élément `::first-letter`.

L'exemple suivant illustre comment des pseudo-éléments superposés peuvent interagir. La première lettre de chaque élément `P` sera verte avec une taille de fonte de '24pt'. Le reste de la première ligne formatée sera 'blue' alors que le reste du paragraphe sera 'red'.

```
P { color: red; font-size: 12pt }
P::first-letter { color: green; font-size: 200% }
P::first-line { color: blue }

<P>Some text that ends up on two lines</P>
```

En supposant qu'un saut de ligne se produira avant le mot "ends", la séquence de balises fictives pour ce fragment pourra alors être:

```
<P>
<P::first-line>
<P::first-letter>
S
</P::first-letter>ome text that
</P::first-line>
ends up on two lines
</P>
```

Veuillez noter que les éléments ::first-letter se trouvent à l'intérieur de l'élément `::first-line`. Les propriétés assignées à `::first-line` sont héritées par `::first-letter`, mais sont surclassées si la même propriété est assignée à `::first-letter`.

7.3 Les pseudo-éléments fragments d'éléments d'interface

Le pseudo-élément ::selection

Le pseudo-élément `::selection` s'applique à la portion du document qui a été mise en exergue par l'utilisateur. Cela s'applique aussi, par exemple, au texte sélectionné dans un champ de texte éditable. Seuls les éléments qui ont une propriété 'user-select' différente de 'none' peuvent avoir un `::selection`. Ce pseudo-élément ne doit pas être confondu avec la pseudo-classe `:checked` (qui s'appelait auparavant `:selected`).

Malgré que le pseudo-élément `::selection` soit dynamique par nature, et soit modifié par l'action de l'utilisateur, il est raisonnable de s'attendre à ce que, quand un Agent Utilisateur re-restitue sur un medium statique (comme une page imprimée) ce qui était initialement restitué sur un medium dynamique (comme un écran), l'Agent Utilisateur puisse souhaiter transférer la

`::selection` courante à cet autre medium, at disposer de tout le formatage et la restitution appropriés également. Cela n'est pas requis - certains Agents Utilisateurs peuvent omettre le pseudo-élément `::selection` pour les media statiques.

Voici les propriétés CSS qui s'appliquent aux pseudo-éléments `::selection` : couleur, fond, contour. La couleur de 'background-image' sur `::selection` peut être ignorée.

7.4 Les pseudo-éléments ::before et ::after

Les pseudo-éléments `::before` et `::after` peuvent être utilisés pour décrire du contenu généré devant ou après le contenu d'un élément. Ils sont expliqués dans le Module CSS 3 Contenu Généré et Marqueurs (module 14).

Quand les pseudo-éléments `::first-letter` et `::first-line` sont combinés avec `::before` et `::after`, ils 'appliquent à la première lettre iy ligne de l'élément y compris le texte inséré.

8. Combinateurs

8.1 Combinateur de descendance

Parfois, les auteurs veulent des sélecteurs pour décrire un élément qui est le descendant d'un autre élément dans l'arbre documentaire (par exemple "un élément `EM` qui est contenu dans un élément `H1`") . Les combinateurs de descendance expriment ce type de relation. Un combinateur de descendance est un espace séparant deux séquences de sélecteurs simples. Un sélecteur de la forme "`A B`" représente un élément `B` qui est un descendant quelconque d'un élément ancêtre `A`.

Exemple(s):

Par exemple, considérons le sélecteur suivant :

```
h1 em
```

C'est une description correcte et valide, mais partielle, du fragment suivant :

```
<h1>This <span class="myclass">headline
is <em>very</em> important</span></h1>
```

Le sélecteur suivant :

```
div * p
```

représente un élément `p` qui est le petit-fils ou un descendant ultérieur d'un élément `div`. Veuillez noter les espaces entourant le "*".

Le sélecteur suit suit, qui combine des sélecteurs de descendance et des sélecteurs d'attributs, représente un élément qui (1) possède l'attribut `href` assigné (2) est à l'intérieur d'un `p` lui-même dans un `div`.

```
div p *[href]
```

8.2 Combinateur filial

Un combinateur filial décrit une relation de filiation entre deux éléments. Un combinateur filial est composé du caractère ">" et sépare deux séquences de sélecteurs simples.

Exemple(s):

Le sélecteur suivant représente un élément `p` qui est le fils de `body` :

```
body > p
```

Exemple(s):

L'exemple qui suit combine des combinateurs de descendance et des combinateurs filiaux :

```
div ol>li p
```

Il représente un élément `p` qui est le descendant d'un `li` ; l'élément `li` doit être le fils d'un élément `ol` ; l'élément `ol` doit être le descendant d'un `div`. Veuillez noter que les espaces optionnels autour du ">" ont été omis.

Pour toute information sur la sélection du premier fils d'un élément, veuillez s'il-vous-plait voir la section sur la pseudo-classe `:first-child` ci-dessus.

8.3 Combinateurs d'adjacence

Il y a deux différents combinateurs d'adjacence : le combinateur d'adjacence directe et le combinateur d'adjacence indirecte.

8.3.1 Combinateur d'adjacence directe

Les combinateurs d'adjacence directe sont composés du caractère "+" séparant deux séquences de sélecteurs simples. Les éléments représentés par les deux séquences partagent le même parent dans l'arbre documentaire et l'élément représenté par la première séquence précède immédiatement l'élément représenté par la seconde.

Exemple(s):

Ainsi, le sélecteur suivant représente un élément `p` suivant immédiatement un élement `math` :

```
math + p
```

Exemple(s):

Le sélecteur suivant est conceptuellement similaire à celui de l'exemple précédent, à ceci près qu'il ajoute un sélecteur d'attribut. Ainsi, il ajoute une contrainte à l'élément `h1` qui doit avoir `class="opener"` :

```
h1.opener + h2
```

8.3.2 Combinateur d'adjacence indirecte

Les combinateurs d'adjacence indirecte sont composés du caractère "~" séparant deux séquences de sélecteurs simples. Les éléments représentés par les deux séquences partagent le même parent dans l'arbre documentaire et l'élément représenté par la première séquence précède (pas nécessairement immédiatement) l'élément représenté par la seconde.

Exemple(s):

```
h1 ~ pre
```

représente un élément `pre` suivant un `h1`. C'est une description correcte et valide, mais partielle, de :

```
<h1>Definition of the function a</h1>
<p>Function a(x) has to be applied to all figures in the table.</p>
<pre>function a(x) = 12x/13.5</pre>
```

Propriétés du texte

Propriété	Valeurs (exemples)	Description
font-family	*police1, police2, police3*, serif, sans-serif, monospace	Nom de police
@font-face	*Nom et source de la police*	Police personnalisée
font-size	1.3em, 16px, 120%...	Taille du texte
font-weight	bold, normal	Gras
font-style	italic, oblique, normal	Italique
text-decoration	underline, overline, line-through, blink, none	Soulignement, ligne au-dessus, barré ou clignotant
font-variant	small-caps, normal	Petites capitales
text-transform	capitalize, lowercase, uppercase	Capitales
font	-	Super propriété de police. Combine : font-weight, font-style, font-size, font-variant, font-family.
text-align	left, center, right, justify	Alignement horizontal
vertical-align	baseline, middle, sub, super, top, bottom	Alignement vertical (cellules de tableau ou éléments inline-block uniquement)
line-height	18px, 120%, normal...	Hauteur de ligne
text-indent	25px	Alinéa
white-space	pre, nowrap, normal	Césure
word-wrap	break-word, normal	Césure forcée
text-shadow	5px 5px 2px blue *(horizontale, verticale, fondu, couleur)*	Ombre de texte

Propriétés de couleur et de fond

Propriété	Valeurs (exemples)	Description
color	*nom*, rgb(rouge,vert,bleu), rgba(rouge,vert,bleu,transparence), #CF1A20...	Couleur du texte
background-color	*Identique à color*	Couleur de fond
background-image	url('image.png')	Image de fond
background-attachment	fixed, scroll	Fond fixe
background-repeat	repeat-x, repeat-y, no-repeat, repeat	Répétition du fond
background-position	*(x y)*, top, center, bottom, left, right	Position du fond
background	-	Super propriété du fond. Combine : background-image, background-repeat, background-attachment, background-position
opacity	0.5	Transparence

Propriétés des boîtes

Propriété	Valeurs (exemples)	Description
width	150px, 80%...	Largeur
height	150px, 80%...	Hauteur
min-width	150px, 80%...	Largeur minimale
max-width	150px, 80%...	Largeur maximale
min-height	150px, 80%...	Hauteur minimale
max-height	150px, 80%...	Hauteur maximale
margin-top	23px	Marge en haut
margin-left	23px	Marge à gauche
margin-right	23px	Marge à droite
margin-bottom	23px	Marge en bas
margin	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge. Combine : margin-top, margin-right, margin-bottom, margin-left.
padding-left	23px	Marge intérieure à gauche
padding-right	23px	Marge intérieure à droite
padding-bottom	23px	Marge intérieure en bas
padding-top	23px	Marge intérieure en haut
padding	23px 5px 23px 5px *(haut, droite, bas, gauche)*	Super-propriété de marge intérieure. Combine : padding-top, padding-right, padding-bottom, padding-left.
border-width	3px	Épaisseur de bordure
border-	*nom*, rgb(rouge,vert,bleu),	Couleur de bordure

color	rgba(rouge,vert,bleu,transparence), #CF1A20...	
border-style	solid, dotted, dashed, double, groove, ridge, inset, outset	Type de bordure
border	3px solid black	Super-propriété de bordure. Combine border-width, border-color, border-style. Existe aussi en version border-top, border-right, border-bottom, border-left.
border-radius	5px	Bordure arrondie
box-shadow	6px 6px 0px black *(horizontale, verticale, fondu, couleur)*	Ombre de boîte

Propriétés de positionnement et d'affichage

Propriété	Valeurs (exemples)	Description
display	block, inline, inline-block, table, table-cell, none...	Type d'élément (block, inline, inline-block, none...)
visibility	visible, hidden	Visibilité
clip	rect (0px, 60px, 30px, 0px) *rect (haut, droite, bas, gauche)*	Affichage d'une partie de l'élément
overflow	auto, scroll, visible, hidden	Comportement en cas de dépassement
float	left, right, none	Flottant
clear	left, right, both, none	Arrêt d'un flottant
position	relative, absolute, static,fixed	Positionnement
top	20px	Position par rapport au haut
bottom	20px	Position par rapport au bas
left	20px	Position par rapport à la gauche
right	20px	Position par rapport à la droite
z-index	10	Ordre d'affichage en cas de superposition. La plus grande valeur est affichée par-dessus les autres.

Propriétés des listes

Propriété	Valeurs (exemples)	Description
list-style-type	disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none	Type de liste
list-style-position	inside, outside	Position en retrait
list-style-image	url('puce.png')	Puce personnalisée
list-style	-	Super-propriété de liste. Combine list-style-type, list-style-position, list-style-image.

Propriétés des tableaux

Propriété	Valeurs (exemples)	Description
border-collapse	collapse, separate	Fusion des bordures
empty-cells	hide, show	Affichage des cellules vides
caption-side	bottom, top	Position du titre du tableau

Autres propriétés

Propriété	Valeurs (exemple)	Description
cursor	crosshair, default, help, move, pointer, progress, text, wait, e-resize, ne-resize, auto...	Curseur de souris