Blocs et sélecteurs

Nous nous sommes déjà familiarisés avec les ciblages les plus élémentaires par le biais d'une balise native HTML, d'une classe ou d'un identifiant. Par le jeu des relations d'héritages, nous avons également vu que l'on pouvait composer des ciblages plus précis en mélangeant balises, classes et identifiants. Il reste encore de nombreuses possibilités à explorer que nous allons passer en revue dans les paragraphes suivants.

I. Les sélecteurs

Sélecteurs
Catégorie d'éléments permettant de cibler un élément spécifique du balisage HTML pour lui appliquer un style. Ils peuvent être de type simple (comme le sélecteur universel *) ou de type attribut ou encore de type id.

Attribut
Élément permettant d'ajouter une information supplémentaire. *Exemple : l'élément « img » n'aurait aucun sens sans l'attribut « src ».*

Certains ciblages, tel **nth-of-type**, seront très pratiques, par exemple pour styliser les rangées d'un tableau sans avoir à intervenir dans le HTML. S'il n'est pas nécessaire de connaître sur le bout des doigts la manière dont on utilise ces éléments, il est intéressant de savoir ce que l'on peut réaliser pour le mettre en œuvre après quelques recherches.

Les sélecteurs que nous allons étudier peuvent se décomposer en quatre familles :

1. les sélecteurs par parenté ou adjacence ;

2. les sélecteurs d'attributs ;

3. les pseudo-classes ;

4. les pseudo-éléments.

A. Les sélecteurs par parenté et adjacence

1. X +Y

Le sélecteur adjacent correspond à tout élément Y immédiatement précédé par un élément X. L'exemple ci-dessous permet d'appliquer un style au premier paragraphe qui suit une liste non ordonnée.

```
ul + p {
    color: red;
}
```

Fig. 1

HTML

```
<ul>
    <li><a href="lien-relatif.html">lien relatif</a></li>
    <li><a href="lieu.gif">vers image gif</a></li>
    <li><a href="https://www.foo.com">lien absolu</a></li>
    <li><a href="image.jpg">vers image jpg</a></li>
</ul>
<p> je suis le frère de ul</p>
```

Fig.2

Rendu

Fig.3

2. X > Y

Contrairement au sélecteur descendant, le sélecteur d'enfant cible seulement le descendant direct. Considérons le code suivant.

CSS

```
.main-menu li {
    color:blue;

}
.main-menu > li {
    list-style: none;
    color:red;
}
```

Fig.4

HTML

```
<ul class="main-menu">
    <li> Enfant direct de main-menu
        <ul class="sub-menu">
            <li> Enfant non direct de main-menu </li>
        </ul>
    </li>
    <li> Enfant direct de main-menu  </li>
    <li> Enfant direct de main-menu  </li>
    <li> Enfant direct de main-menu  </li>
</ul>
```

Fig. 5

Le sélecteur « **.main-menu > ul** » cible tous les éléments **li** qui sont descendants directs d'un bloc ayant pour classe **.main-menu**. Contrairement à « **.main-menu li** », il ne ciblera pas l'élément **li** descendant de **.sub-menu**.

Rendu

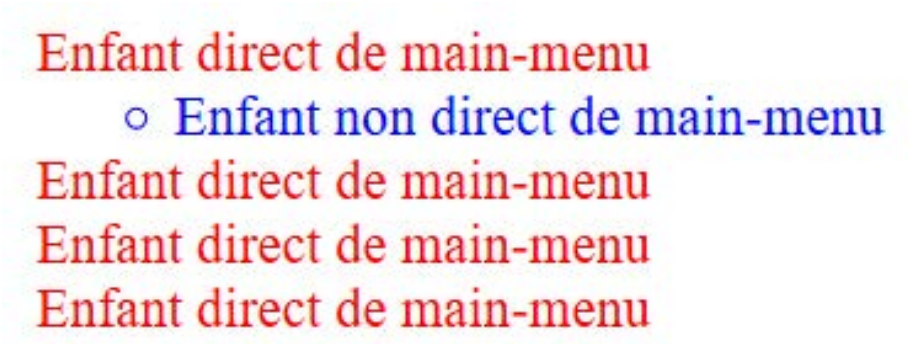

Fig. 6

3. Syntaxe : X~Y

Permet d'ajouter un style à tous les éléments qui suivent un élément particulier.

CSS

```css
.example4 div{
margin:0;
padding:10px;
color:#000;
}

.example4 div~p{
color:#fff;
margin:20px;
width:200px;
padding:5px;
border:1px solid #333;
background:#006644;
}
```

Fig.7

HTML

```html
<div class="example4">
    <div>je suis l'élément particulier div</div>
    <p> je suis un p qui suit le div (l'élement particulier)</p>
    <p>je suis un p qui suit le div (l'élement particulier)</p>
    <span>je suis un span</span>
</div>

<p>je suis un p qui ne suit pas le div (l'élement particulier)</p>
```

Fig.8

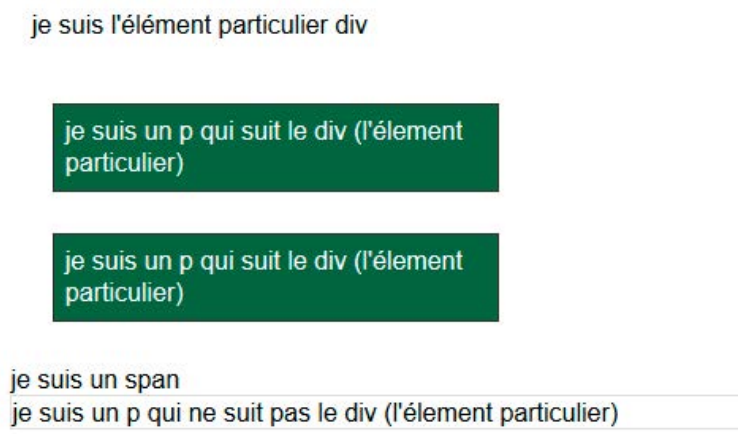

Fig. 9

B. Les sélecteurs par attributs

Syntaxe : [attrib]

Ou

[attrib = "valeur"]

1. [attrib]

Correspond à tout élément ayant un attribut défini.

***Exemple** : un lien ayant l'attribut title.*

CSS

```css
a[title] {
color: green;
}
```

Fig. 10

HTML

```html
<ul>
    <li><a href="monlien.html"> Je n'ai pas de titre </a></li>
    <li> <a href="monlien.html" title="j'ai un titre"> j'ai un titre</a></li>
    <li><a href="monlien.html"> Je n'ai pas de titre  </a></li>
</ul>
```

Fig. 11

Rendu

- Je n'ai pas de titre
- j'ai un titre
- Je n'ai pas de titre

Fig. 12

2. [attrib=« valeur »]

Permet de sélectionner tous les éléments dont les attributs ont une valeur précise, par exemple le code ci-dessous applique un style à tous les éléments **a** qui pointent vers http://www.foo.com.

CSS

```css
a[href="http://www.foo.com"] {
    color: #1f6053;
}
```

Fig. 13

HTML

```html
<ul>
    <li><a href="monlien.html"> Je pointe vers monlien.html </a></li>
    <li><a href="http://www.foo.com"> je pointe vers foo.com</a></li>
    <li><a href="monlien.html"> Je pointe vers monlien.html  </a></li>
</ul>
```

Fig. 14

On obtient :

- Je pointe vers monlien.html
- je pointe vers foo.com
- Je pointe vers monlien.html

Fig. 15

3. [attr*=« valeur »]

Ce sélecteur permet de sélectionner un élément dont l'attribut comporte au moins une fois la valeur définie.

```css
.example3{
    margin:0;
    padding:10px;
    color:#000;
}
.example3[title*="val"]{
    color:#fff;
    background:#990000;
}
```

Fig. 16

```html
<p class="example3"> je n'ai pas d'attribut title</p>
<p class="example3" title="comment"> j'ai un attribut title mais
il ne contient pas "val"</p>
<p class="example3" title="val"> j'ai un attribut title contenant
au moins "val"</p>
<p class="example3" title="evaluer"> j'ai un attribut title
contenant au moins "val" également</p>
<p class="example3" title="eval"> j'ai un attribut title
contenant au moins "val" également</p>
```

Fig. 17

Dans cet exemple, les 3 derniers paragraphes sont concernés.

Le rendu :

je n'ai pas d'attribut title

j'ai un attribut title mais il ne contient pas "val"

j'ai un attribut title contenant au moins "val"

j'ai un attribut title contenant au moins "val" également

j'ai un attribut title contenant au moins "val" également

Fig. 18

Prenons un autre exemple, si on a :

```
a[href*="foo"] {
    color: #1f6053;
}
```

Fig.19

*L'étoile indique donc que « **foo** » doit se trouver quelque part dans la valeur de l'attribut. Ainsi, ce code prend également en compte foo.com, cours.foo.com, elections.tomsyfoo.com.*

HTML

```
<ul>
    <li><a href="monlien.html"> Je pointe vers monlien.html</a></li>
    <li><a href="http://www.cours.foo.com"> Je pointe vers cours.foo.com</a></li>
    <li><a href="http://www.foo.com"> je pointe vers foo.com</a></li>
    <li><a href="http://www.football.com"> Je pointe vers football.com</a></li>
</ul>
```

Fig.20

On obtient :

- Je pointe vers monlien.html
- Je pointe vers cours.foo.com
- je pointe vers foo.com
- Je pointe vers football.com

Fig.21

a. [attr^=« valeur »]

Ce sélecteur permet de sélectionner un élément dont l'attribut commence exactement par la valeur définie.

Par exemple :

```
a[href^="http://"] {
    text-decoration :none ;
    color : #0a0 ;
}
```

Fig.22

Cible tous les liens dont la valeur de l'attribut commence par « http:// ».

On peut par exemple discriminer des liens internes (en relatif) de liens d'ancres ou encore de liens en https.

```
<ul>
    <li><a href="lien-relatif.html">Je suis un lien relatif</a></li>
    <li><a href="http://www.foo.com/portrait.jpg">Je suis un lien absolu en http://</a></li>
    <li><a href="https://www.foo.com">Je suis un lien absolu en https://</a></li>
    <li><a href="#paragraphe"> Je suis un lien d'ancre</a></li>
</ul>
```

Fig. 23

Donne :

- Je suis un lien relatif
- Je suis un lien absolu en http://
- Je suis un lien absolu en https://
- Je suis un lien d'ancre

Fig. 24

Exemple 2 :
CSS

```
.example{
    margin:0;
    padding:10px;
    color:#000;
}
.example[title^="ess"]{
    color:#fff;
    background:#333;
}
```

Fig. 25

HTML

```
<p class="example"> je n'ai pas d'attribut title</p>

<p class="example" title="comment"> j'ai un attribut title mais
il ne commence pas par "ess"</p>

<p class="example" title="essai"> j'ai un attribut title
commençant par "ess"</p>

<p class="example" title="esson"> j'ai un attribut title
commençant par "ess" également</p>
```

Fig. 26

Dans cet exemple, les 2 derniers paragraphes sont concernés.

Rendu

Fig.27

4. [attr$=« valeur »]

Ce sélecteur permet de sélectionner un élément dont l'attribut finit exactement par la valeur définie.

Exemple :
CSS

```css
.example2{
    margin:0;
    padding:10px;
    color:#000;
}
.example2[title$="sai"]{
    color:#fff;
    background:#045FB4;
}
```

Fig.28

HTML

```html
<p class="example2"> je n'ai pas d'attribut title</p>
<p class="example2" title="comment"> j'ai un attribut title mais
il ne finit pas par "sai"</p>
<p class="example2" title="essai"> j'ai un attribut qui finit
par "sai""</p>
<p class="example2" title="esson"> j'ai un attribut title mais
il ne finit pas par "sai"</p>
```

Fig.29

Rendu

je n'ai pas d'attribut title

j'ai un attribut title mais il ne finit pas par "sai"

j'ai un attribut qui finit par "sai""

j'ai un attribut title mais il ne finit pas par "sai"

Fig.30

Dans cet exemple, le troisième paragraphe est concerné.

Exemple 2 :

CSS

```css
a[href$=".jpg"] {
    color: red;
    text-decoration: none;
}
```

Fig.31

HTML

```html
<ul>
    <li><a href="lien-relatif.html">lien relatif</a></li>
    <li><a href="lieu.gif">vers image gif</a></li>
    <li><a href="https://www.foo.com">lien absolu</a></li>
    <li><a href="image.jpg">vers image jpg</a></li>
</ul>
```

Fig.32

Le rendu

Fig.33

5. [attrib~=« valeur »]

Le symbole tilde permet de cibler un élément dont l'attribut défini possède la valeur définie exactement, celle-ci peut être séparée par une espace.

Par exemple :
CSS

```css
a[class~="external"] {
    text-decoration :none ;
    border: 1px solid black;
    display: block;
    padding: 5px;
    background-color:#ededed;
}

a[class~="image"] {
    color: red;
    text-decoration :none ;
}
```

Fig.34

On cherche les éléments qui ont pour classe « external » (règle 1) ; on cherche les éléments qui ont pour classe « image » .

HTML

```
<ul>
    <li><a href="lien-relatif.html" class="internal">lien relatif</a></li>
    <li><a href="lieu.gif" class="image">vers image</a></li>
    <li><a href="https://www.foo.com" class="external">lien externe</a></li>
    <li><a href="https://www.foo.com/image.jpg" class="external image">vers image externe</a></li>
</ul>
```

Fig.35

Rendu

- lien relatif
- vers image
- lien externe
- vers image externe

Fig.36

6. [attrib|=« valeur »]

Cette fois-ci, la valeur définie peut être séparée par d'autres valeurs à l'aide de tirets.

Exemple :
CSS

```
p{
    margin:0;
    padding:10px;
    color:#000;
}
p[class|="val"]{
    color:#fff;
    background:#990000;
    border:2px solid #ddffdd;
}
```

Fig.37

On cherche des éléments qui sont des paragraphes (règle 1) ; on cherche des éléments qui sont des paragraphes et qui possèdent un attribut dont la valeur est « val » ou qui commence par « val- ».

HTML

```
<p class="classe">class="class"</p>
<p class="val-jaune">class="val-jaune"</p>
<p class="val-rouge">class="val-rouge"</p>
<p class="uneautre">class="uneautre" </p>
```

Fig.38

Rendu

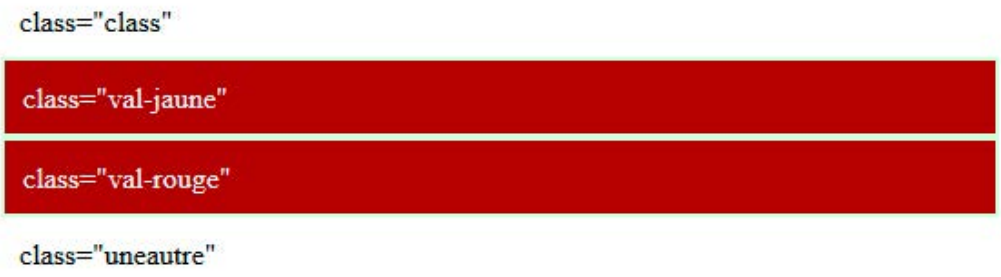

Fig. 39

C. Les pseudo-classes

Les pseudo-classes permettent de cibler un élément sans avoir à ajouter un nom de classe dans le html. On cible par exemple le premier enfant ou le dernier enfant d'un élément. En CSS3, la syntaxe d'une pseudo-classe s'écrit **:nom-de-la-pseudo-classe** .

1. :hover

Vous voulez appliquer un style particulier lorsque l'utilisateur survole un élément ? L'élément **:hover** fera l'affaire. Vous devez déjà le connaître pour les liens.

HTML

```
<p class="over">
Je change de couleur au survol !
</p>
```

Fig. 40

CSS

```
.over:hover {
background: #ffe3e3;
}
```

Fig. 41

Rendu avant le survol

Je change de couleur au survol !

Fig. 42

Rendu au survol

Je change de couleur au survol !

Fig. 43

2. :not(selecteur)

La pseudo-classe de négation a une utilité particulière. On sélectionne un élément par différence, par ce qu'il n'est pas.

Exemple :
HTML

```
<ul class="main-menu">
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
    <li class="not-menu-link">n'est pas un lien menu</li>
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
</ul>
```

Fig. 44

CSS

```
li:not(.menu-link) {
color: blue;
}
```

Fig. 45

Rendu

- lien menu
- lien menu
- lien menu
- n'est pas un lien menu
- lien menu
- lien menu

Fig. 46

3. :first-child et :last-child

Ces pseudo-classes ciblent le premier et/ou le dernier enfant.

Exemple :
HTML

```
<ul class="main-menu">
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
    <li class="menu-link">lien menu</li>
</ul>
```

Fig. 47

CSS

```
li:first-child {
border-top:1px solid #f00;
padding-top: 2px;
}
li:last-child {
border-bottom: 1px solid #f00;
padding-bottom: 2px;
}
```

Fig. 48

Rendu

lien menu
lien menu
lien menu
lien menu
lien menu

Fig.49

4. :first-of-type et :last-of-type

Ces pseudo-classes ciblent un élément qui est le premier enfant de son type ou le dernier élément de son type.

Exemple :
HTML

```
<p>
    Lorem <em>premier élément italisé</em>ipsum dolor sit a met, <strong>premier gras</
    strong> adipiscing elit. Nunc aliquet nisl massa,<em> élément italisé</em> et rutrum
    odio tincidunt vitae. Praesent tincidunt risus sit amet diam volutpat, eu ornare
    metus pretium. Integer vitae sem non<strong> deuxième gras </strong>sit amet quis
    sem. Sed ex urna, sagittis et massa eget, pharetra aliquam <em> dernier élément
    italisé</em>lorem. Quisque ullamcorper pretium <strong>dernier gras</strong>. Donec
    condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui
    condimentum ultricies in vel eros.
</p>
```

Fig.50

CSS

```
em:first-of-type {
    color:#F00;
    background-color: #cdcdcd;}
strong:last-of-type{
    color:#F00;
    background-color: #cdcdcd;
}
```

Fig.51

Rendu

Lorem *premier élément italiséipsum* dolor sit a met, **premier gras** adipiscing elit. Nunc aliquet nisl massa, *élément italisé* et rutrum odio tincidunt vitae. Praesent tincidunt risus sit amet diam volutpat, eu ornare metus pretium. Integer vitae sem non **deuxième gras** sit amet quis sem. Sed ex urna, sagittis et massa eget, pharetra aliquam *dernier élément italisélorem*. Quisque ullamcorper pretium **dernier gras**. Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in ve eros.

Fig.52

5. :nth-child(expression) et :nth-last-of-child(expression)

Très pratique, ce sélecteur permet de cibler tous les éléments en se basant sur leur position dans la liste des enfants. On commence par le début (**:nth-child**), ou par la fin (**:nth-last-of-child**).

(expression) peut être un nombre, une expression numérique ou un mot clé tel que **odd** (concerne les enfants impairs) ou **even** (concerne les enfants pairs), un chiffre **n** pour les multiples de chiffres.

Exemple 1 (en commençant par le début) :
HTML : on utilise 4 tableaux ayant chacun la classe suivante exampleTable1, example Table2, etc. Le tableau lui-même est de la forme :

```
<table class="exampleTable1">
        <caption>tous les enfants aux numéros pairs</caption>
        <tr>
            <td>1ere ligne</td>
        </tr>
        <tr>
            <td>2eme ligne</td>
        </tr>
        <tr>
            <td>3eme ligne</td>
        </tr>
        <tr>
            <td>4eme ligne</td>
        </tr>
        <tr>
            <td>5ere ligne</td>
        </tr>
        <tr>
            <td>6eme ligne</td>
        </tr>
        <tr>
            <td>7eme ligne</td>
        </tr>
        <tr>
            <td>8eme ligne</td>
        </tr>
    </table>
```

Fig.53

CSS

```css
table, caption{
        width:250px;
        border:1px solid #444;
        float:left
}

.exampleTable1 tr:nth-child(even){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
.exampleTable2 tr:nth-child(odd){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
.exampleTable3 tr:nth-child(3n){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
.exampleTable4 tr:nth-child(7){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
```

Fig.54

Rendu des tableaux 1 et 2

tous les enfants aux numéros pairs	tous les enfants aux numéros impairs
1ere ligne	1ere ligne
2eme ligne	2eme ligne
3eme ligne	3eme ligne
4eme ligne	4eme ligne
5ere ligne	5ere ligne
6eme ligne	6eme ligne
7eme ligne	7eme ligne
8eme ligne	8eme ligne

Fig.55

Rendu des tableaux 3 et 4

tous les enfants multiple de 3	l'enfant numéro 7
1ere ligne	1ere ligne
2eme ligne	2eme ligne
3eme ligne	3eme ligne
4eme ligne	4eme ligne
5ere ligne	5ere ligne
6eme ligne	6eme ligne
7eme ligne	7eme ligne
8eme ligne	8eme ligne

Fig.56

Exemple 2 (on commence par la fin) :
HTML : même principe

CSS

```
table, caption{
        width:250px;
        border:1px solid #444;
        float:left
}
.exampleTable1 tr:nth-last-child(even){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
.exampleTable2 tr:nth-last-child(odd){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
.exampleTable3 tr:nth-last-child(3n){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
.exampleTable4 tr:nth-last-child(7){
    text-shadow:1px 1px 2px #000;
    color:blue;
}
```

Fig.57

Rendu des tableaux 1 et 2

tous les enfants pairs à partir de la fin
8e ligne
7eme ligne
6eme ligne
5eme ligne
4ere ligne
3eme ligne
2eme ligne
1ere ligne

tous les enfants aux numéros impairs à partir de la fin
8e ligne
7eme ligne
6eme ligne
5eme ligne
4ere ligne
3eme ligne
2eme ligne
1ere ligne

Fig. 58

Rendu des tableaux 3 et 4

tous les enfants multiple de 3 à partir de la fin
8e ligne
7eme ligne
6eme ligne
5eme ligne
4ere ligne
3eme ligne
2eme ligne
1ere ligne

l'enfant numéro 7 à partir de la fin
8e ligne
7eme ligne
6eme ligne
5eme ligne
4ere ligne
3eme ligne
2eme ligne
1ere ligne

Fig. 59

Remarque :

À partir de ce que nous avons vu, à quoi correspondent **:nth-child(1)** et **:nth-last-of-child(1)** ?

Tout simplement à **:first-child** et **:last-child**.

6. :only-child

Cette pseudo-classe permet de cibler les éléments qui sont enfants uniques.

Exemple :
HTML

```
<article>
   <p>Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem
   quis dui condimentum ultricies in vel eros.
   </p>
   <p>Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem
   quis dui condimentum ultricies in vel eros.
   </p>
</article>
<article>
   <p>Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem
   quis dui condimentum ultricies in vel eros.
   </p>
</article>
```

Fig.60

CSS

```
article {
    border:1px solid #000;
}
article p:only-child {
color: red;
}
```

Fig.61

Rendu

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Fig.62

Dans cet exemple, seul le paragraphe de la deuxième balise article sera en rouge.

7. :only-of-type

Cible tous les éléments qui n'ont pas de frères au sein de l'élément parent.

HTML : identique à l'exemple précédent

CSS

```css
article {
    border:1px solid #000;
}
p:only-of-type {
font-weight: bold;
}
```

Fig. 63

Rendu

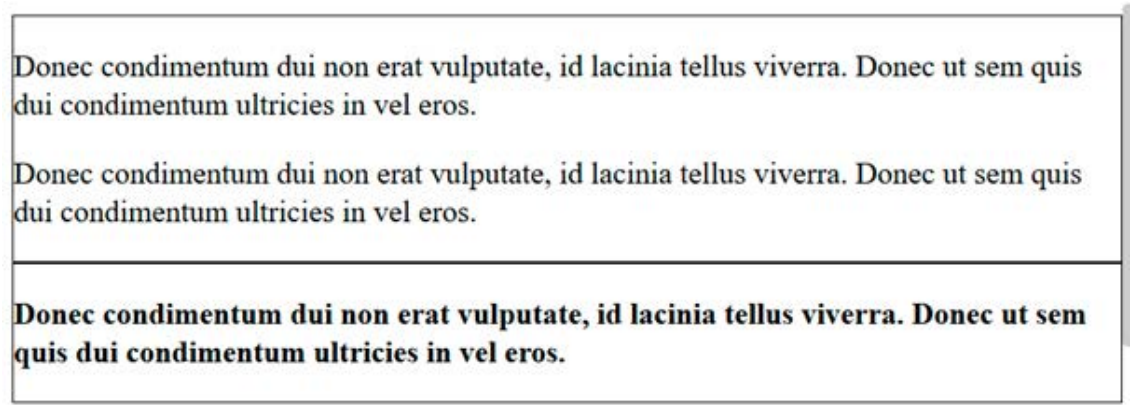

Fig. 64

Dans cet élément, seul le texte du deuxième <p> sera en gras.

8. :root

Ce sélecteur représente un élément qui est la racine d'un document. Par exemple, en HTML4, l'élément est HTML.

9. :nth-of-type(expression) et :nth-last-of-type

Ces pseudo-classes ciblent le ou les enfants d'un élément qui possède un certain type en partant du début (**:nth-of-type**) ou de la fin (**:nth-last-of-type**).

Exemple en partant du début :
HTML

```
<ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
    <li>6</li>
    <li>7</li>
</ul>
```

Fig. 65

CSS

```
ul {width: 550px}
li {width:250px;height: 50px;}
li:nth-of-type(2n) { float: left;  background-color: #ddd}
li:nth-of-type(2n+1) { float: right;  background-color:red}
```

Fig. 66

Dans cet exemple tous les éléments pairs sont gris et se positionnent à gauche et tandis que les impairs sont rouges et placés à droite.

Rendu

Fig. 67

Le même principe s'applique pour :nth-last-of-type, sauf qu'on commence par la fin.

À partir de ce que nous avons vu, à quoi correspondent :nth-first-of-type(1) et :nth-last-of-type(1) ?

Tout simplement à :first-of-type et :last-of-type.

10. :empty

Correspond aux éléments n'ayant pas d'enfant.

Exemple :
HTML

```
<p>Donec condimentum dui non erat vulputate, id lacinia tellus
viverra. Donec ut sem quis dui condimentum ultricies in vel eros.
</p>
<p>Donec condimentum dui non erat vulputate, id lacinia tellus
viverra. Donec ut sem quis dui condimentum ultricies in vel eros.
</p>
<p></p>
```

Fig. 68

On va demander aux éléments n'ayant pas d'enfant de ne pas apparaître avec display:none.

CSS

```css
p {
    background-color: #ededed;
    min-height:50px;
}
p:empty {
    display: none;
    background-color: #FF0;
}
```

Fig.69

Rendu

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Fig.70

11. :target

Elle permet de cibler un élément qui est la cible d'un lien : par exemple, une ancre.

HTML

```html
<a href="#cible">Allez à la cible</a>
...
<div id="cible">...</div><a href="#cible">Allez à la cible</a>
...
<div id="cible">...</div>

<h1>10.4 Les pseudo-élements
</h1>
```

Fig.71

CSS

```
#cible:target{
    background: lightgray;
}
```

Fig. 72

Lors du clic sur ce lien, l'élément ciblé (ici **<div id=« cible »>... **) est affiché à l'écran, même si celui-ci se trouve plus bas ou plus haut dans la page. Un scroll est donc provoqué.

La pseudo-classe **:target** permet donc d'appliquer un style à l'élément qui reçoit le lien, donc l'élément qui a l'attribut id=« cible ».

Rendu avant le clic

le lien qui amène à la cible

Ceci est la cible qui est stylisée lorsque le lien est cliqué

Fig. 73

Rendu après le clic

le lien qui amène à la cible

Ceci est la cible qui est stylisée lorsque le lien est cliqué

Fig. 74

12. :checked

```css
input[type="radio"]:checked + label{
    padding:0 20px;
    background-color: #f00;
    color:#fff;
}
```

Fig.75

On combine la pseudo classe **:checked** avec un sélecteur d'attribut pour cibler les boutons radio qui ont été cochés. Ensuite, grâce au sélecteur d'éléments adjacents, on va pointer la balise **label** des boutons radios cochés. Cela permettra de distinguer les éléments de formulaire cochés des autres.

HTML

```html
<p>Je veux marcher longtemps</p>

<input type="radio" name="radio" value="radio1">
<label for="radio">Oui</label>

<input type="radio" name="radio" value="radio2">
<label for="radio2">Non</label>
```

Fig.76

Rendu (lorsque le bouton n'est pas coché)

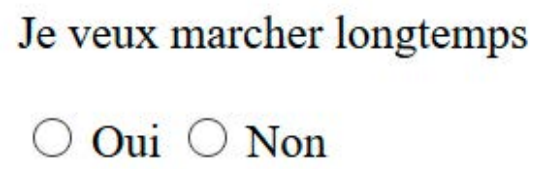

Fig.77

Rendu (lorsque le bouton est coché)

Fig.78

D. Pseudo-éléments

Un pseudo-élément permet de cibler un élément sans avoir à ajouter une balise dans le code html. Par exemple, on ajoute un élément après ou avant une balise HTML. En CSS3, la syntaxe d'un pseudo-élément s'écrit **::nom-du-pseudo-élément**.

1. ::after et ::before

::before et **::after** servent à générer du contenu autour de l'élément ciblé. Les développeurs trouvent chaque jour des idées créatives pour les utiliser de manière efficace. Nous avons déjà vu le pseudo-élément **::after** lorsque nous avons abordé les flottements, en effet, comme nous devons arrêter la flottaison des éléments, une des méthodes consiste à créer un pseudo-élément **::after** qui met fin à la flottaison grâce à **clear:both** et que l'on a coutume de nommer *clearfix* .

Ils existent en CSS2.1 sous la forme **:before** et **:after** .

Exemple
On peut utiliser par exemple before avec la propriété content pour spécifier le contenu qui est inséré. Ici, le texte «Titre :» sera ajouté avant chaque titre de niveau 1.

HTML

```
<h1>La toile</h1>
```

Fig. 79

CSS

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

Fig. 80

Rendu

Titre : La toile

Fig. 81

2. ::first-line

Applique la règle de style à la première ligne du texte de l'élément.

Exemple
Mettre la 1re ligne des éléments <p> en majuscule.

HTML

```
<p>Donec condimentum dui non erat vulputate, id lacinia tellus
viverra. Donec ut sem quis dui condimentum ultricies in vel eros.
Donec condimentum dui non erat vulputate, id lacinia tellus viverra.
Donec ut sem quis dui condimentum ultricies in vel eros.
</p>
```

Fig.82

CSS

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

Fig.83

Rendu

DONEC CONDIMENTUM DUI NON ERAT VULPUTATE, ID LACINIA TELLUS viverra. Donec ut sem quis dui condimentum ultricies in vel eros. Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Fig.84

3. ::first-letter

Applique la règle de style à la première lettre du texte de l'élément.

Exemple
La 1re lettre du ou des éléments « p » a une taille de police de 2em.

HTML

```
<p>Donec condimentum dui non erat vulputate, id lacinia tellus
viverra. Donec ut sem quis dui condimentum ultricies in vel eros.
Donec condimentum dui non erat vulputate, id lacinia tellus viverra.
Donec ut sem quis dui condimentum ultricies in vel eros.
</p>
```

Fig.85

CSS

```
p::first-letter { font-size: 2em }
```

Fig. 86

Rendu

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros. Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Fig. 87

4. ::selection

Applique la règle de style à la sélection du texte de l'élément faite par l'utilisateur (ne fonctionne pas sur tous les navigateurs).

Exemple
À la sélection, le texte sélectionné aura une couleur d'arrière-plan rouge.

HTML : identique à l'exemple précédent.

CSS

```
p::selection {
    background:red
}
```

Fig. 88

Rendu

Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros. Donec condimentum dui non erat vulputate, id lacinia tellus viverra. Donec ut sem quis dui condimentum ultricies in vel eros.

Fig. 89

II. Conclusion

Comprendre la manière de fonctionner des sélecteurs peut vous être d'une aide précieuse lorsqu'on cherche à cibler des éléments HTML. Avec l'habitude, vous saurez utiliser les plus courants et les plus pratiques.