CSS - Sélecteurs avancés

Table des matières

DOM et rendu d’une page web

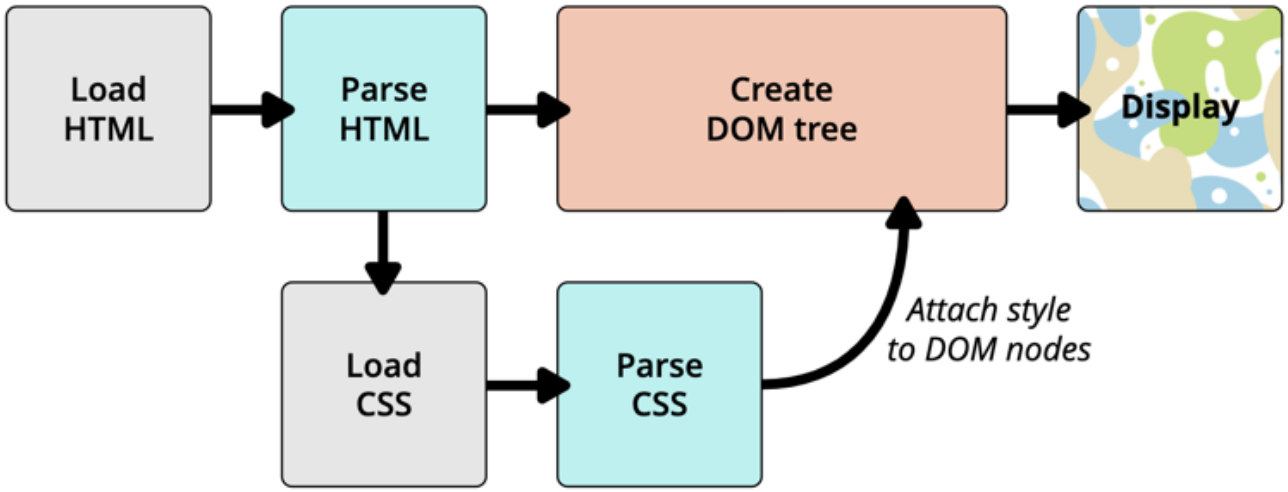


Figure 1. Rendu d’une page HTML [1]

L’arbre DOM

Modèle DOM

Quand une page HTML est **téléchargée** par le navigateur, elle est **parsée** puis **interprétée** sous la forme **d’un arbre** dans lequel élément et contenu sont représentés par **un nœud**.

Cet arbre est le Document Object Model [2] (abbrégé DOM).

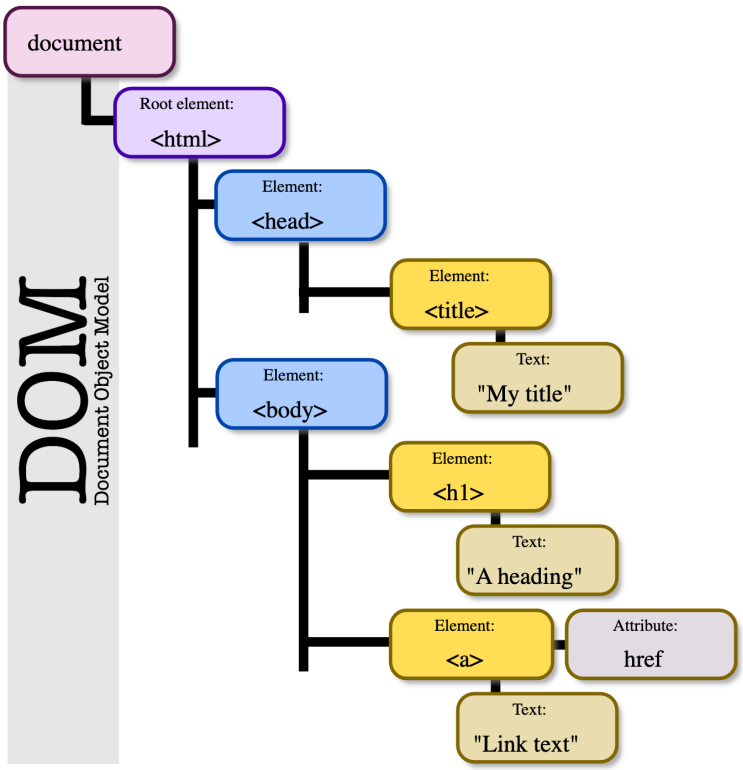


Figure 2. Représentation DOM d'une page HTML [3].

[1] Credits image : https://developer.mozilla.org/en-US/docs/Learn_web_development/Getting_started/Web_standards/How_browsers_load_websites/rendering.svg

[2] Wikipedia - Document Object Model [https://fr.wikipedia.org/wiki/Document_Object_Model]

[3] Par Birger Eriksson — Travail personnel, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18034500

Sélecteurs CSS

Les **sélecteurs CSS**[1][2] permettent de déterminer sur quels éléments s'applique un ensemble de **règles CSS**.

- Élément (`nom de l'élément`)
- Classe (`.nom_de_la_classe`)
- Identifiant (`#nom_de_l_identifiant`)
- Sélecteur universel : (*)
- attribut (`[nom_d_attribut="valeur"]`)

Sélecteur par élément

Applique le style à tous les élements correspondant au type

Syntaxe

```
nom_d_element {
    /* propriétés CSS */
}
```

```
1  <style>
2  p {
3      color: red;
4  }
5  </style>
6
7  <h1>Titre</h1>
8  <p>Paragraphe</p>
9  <section>
10     <h2>Titre 2</h2>
11     <p>Paragraphe 2</p>
12 </section>
```

Example 1. Tous les paragraphes ont le texte en rouge

Titre

Paragraphe

Titre 2

Paragraphe 2

Avantage : il permet de sélectionner des éléments sans avoir à modifier le code

> HTML
>
> **Inconvénient :** le style n'est pas forcément modulaire quand on change le type de l'élément. Dans ce cas, il vaut mieux décorréler le style de l'élément en utilisant une classe CSS.

Sélecteur par classe

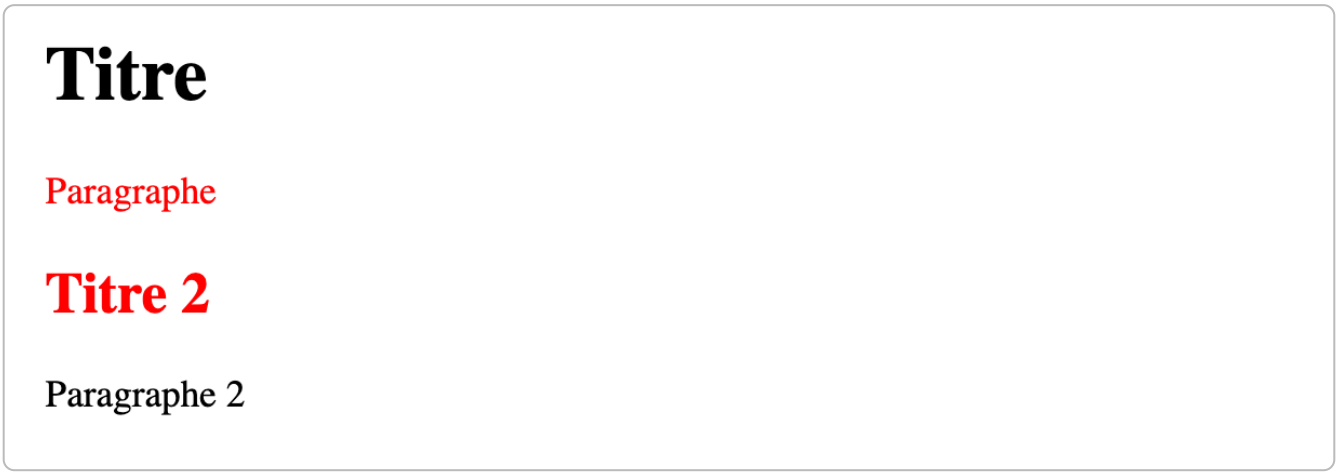
Sélectionner tous les éléments portant un attribut `class` dont la valeur correspond à la directive CSS.

Syntaxe

```
.nom_de_classe {
    /* propriétés CSS */
}
```

```
1  <style>
2      .red {
3          color: red;
4      }
5  </style>
6
7  <h1>Titre</h1>
8  <p class="red">
9      Paragraphe
10 </p>
11 <section>
12     <h2 class="red">Titre 2</h2>
13     <p>
14         Paragraphe 2
15     </p>
16 </section>
```

Example 2. Tous les éléments avec la classe .red sont rouges



Les classes permettent de sélectionner des éléments indistinctement de leur type. Augmentant ainsi la **portabilité** et la **maintenabilité** du style.

Sélecteur par identifiant

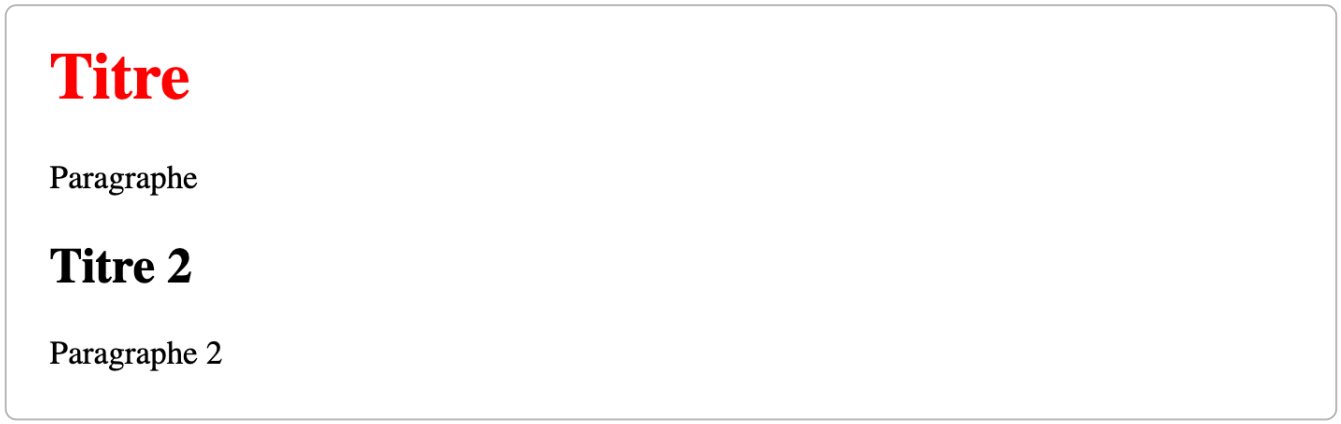
Applique le style à l'élément dont l'identifiant porte la valeur

Syntaxe

```
#nom_de_l_id {
    /* propriétés CSS */
}
```

```
1  <style>
2  #titre_principal {
3      color: red;
4  }
5  </style>
6
7  <h1 id="titre_principal">Titre</h1>
8  <p>Paragraphe</p>
9  <section>
10     <h2>Titre 2</h2>
11     <p>Paragraphe 2</p>
12 </section>
```

Example 3. L'élément avec l'id "titre_principal" est en rouge



Avantage : il permet de sélectionner un élément unique indistinctement du type de l'élément

Inconvénient : le style ne peut pas être réutilisé car l'id est unique. Préférer les `class` si le style a besoin d'être réutilisé.

* sélecteur universel

Applique le style à n'importe quel élément, quel que soit son type.

Syntaxe

```
* {
    /* propriétés CSS */
}
```

```
/*
    Remplace le comportement par défaut de
    `width` et `height` pour s'appliquer
    au niveau de la zone de bordure
    plutôt que la zone de contenu
*/
* {
    box-sizing: border-box;
}
```

Utilisé principalement pour faire un **"reset" CSS**. C'est à dire initialiser les valeurs de certains éléments pour qu'ils aient le même comportement sur l'ensemble des navigateurs.

Sélecteur d'attribut

Applique le style aux élements dont un attribut correspond à la valeur [3]

Syntaxe

```
{sélecteur}[{nom_d_attribut}{opérateur}"{valeur}"]{
    propriétés CSS
}
```

```
body {
    padding: 2rem;
}

/*
Pour les éléments qui possèdent
un attribut "titre"
*/
[title] {
    color: red;
}
```

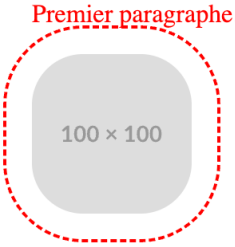
```
/*
Pour les images dont l'attribut
"alt" n'est pas présent.
Entoure de pointillés rouges
*/
img:not([alt]) {
    outline: 2px dashed red;
    outline-offset: 1rem;
    border-radius: 2rem;
}

/*
Pour les liens dont l'attribut
"href" commence par "http://"
ou "https://".
Ajoute un émoji  après les liens externes
*/
:is(a[href^="https://"],a[href^="http://"])::after {
    content: " "
}

/*
Ajoute un émoji avant les
liens externes en HTTP
*/
a[href^="http://"]::before {
    content: " "

}

/*
Ajoute un émoji avant les
liens externes en HTTPS
*/
a[href^="https://"]::before {
    content: " "
}

.dark {
    padding: 3rem;
    background-color: black;
}

/*
Pour les images dont le nom
de fichier termine par ".png".
Ajoute un arrière-plan clair
pour être sûr que les dessins
en noir sur transparence sont
bien lisibles */
img[src$=".png"] {
```

```
    background-color: honeydew;
}
```

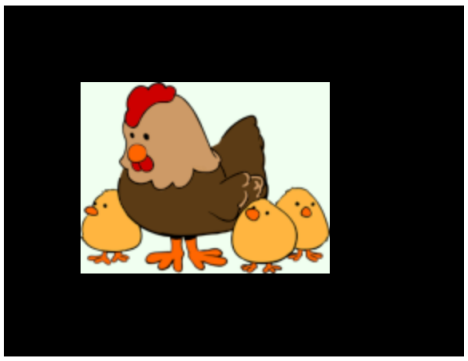
```
<h1>Titre</h1>

<p title="Premier paragraphe">
    Premier paragraphe.
</p>

<img src="https://placehold.co/100"/>

<ul>
    <li>
        <a href="#un_lien_de_page">
            Lien de page
        </a>
    </li>
    <li>
        <a href="http://lien_externe.com">
            Lien externe non sécurisé
        </a>
    </li>
    <li>
        <a href="https://lien_externe.com">
            Lien externe sécurisé
        </a>
    </li>
</ul>

<div class="dark">
    <img
            src="assets/chicken-159496_120.png"
            alt="Une poulet et ses poussins">
</div>
```

Titre

Premier paragraphe.



- Lien de page
- ⚠️ Lien externe non sécurisé 🔗
- 🔐 Lien externe sécurisé 🔗



Souvent très pratique pour styler des liens, images, ou champs de saisie selon leurs attributs.

Voir la documentation MDN [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Attribute_selectors] pour l'ensemble des opérateurs possibles

[1] MDN - CSS Sélecteurs [https://developer.mozilla.org/fr/docs/Web/CSS/CSS_selectors]

[2] MDN - CSS : Liste des sélecteurs [https://developer.mozilla.org/fr/docs/Web/CSS/Selector_list#liste_de_s%C3%A9lecteurs_invalide]

[3] MDN - CSS : sélecteur d'attribut [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Attribute_selectors]

Combinateurs

- descendants : " " (espace)
- enfants : >
- voisins globaux : ~
- voisin direct : +

Les combinateurs ne permettent que de sélectionner des **éléments enfants** (pas parents) et **voisins suivants** (pas précédents). Cependant, on peut contourner cette limitation avec la pseudo-classe `:has()`

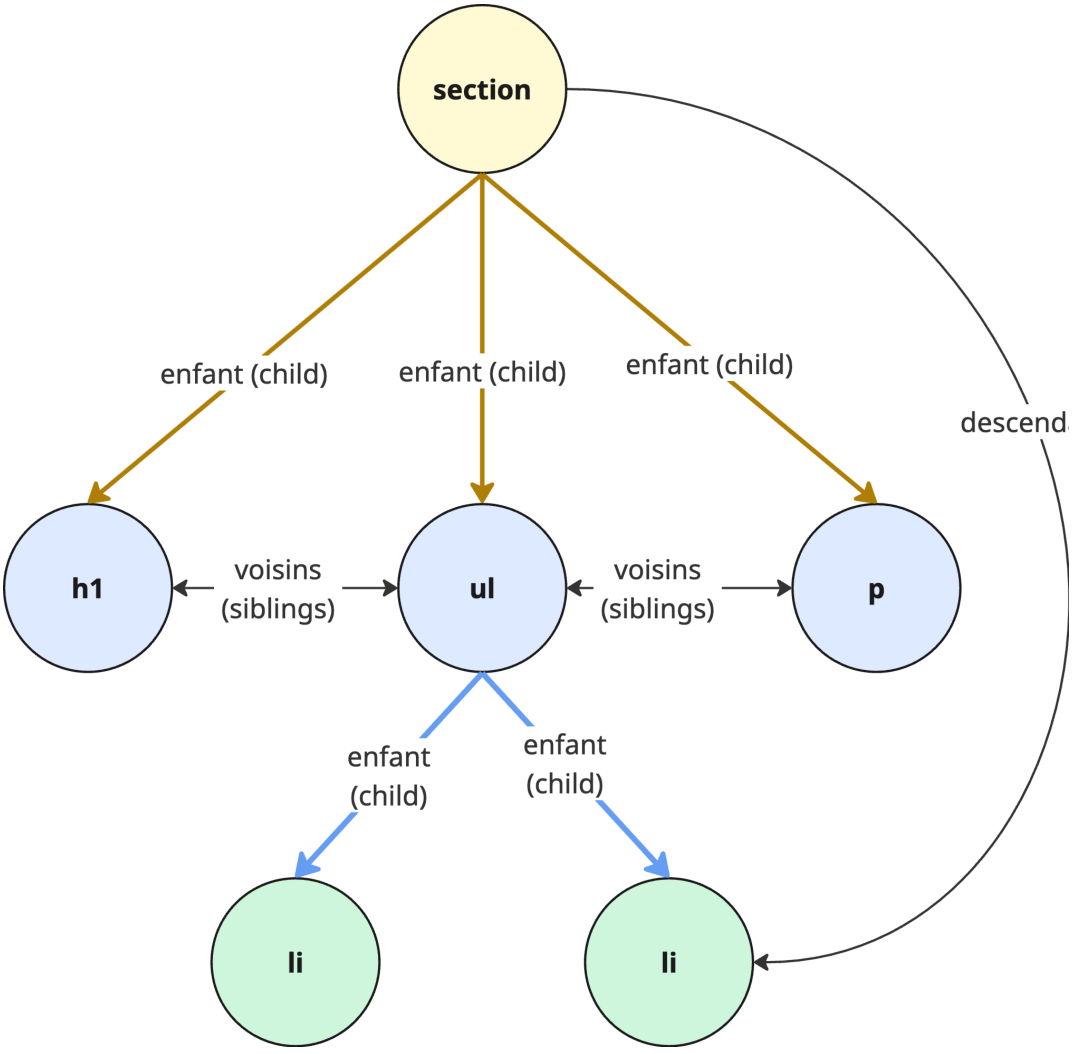


Figure 3. Relation entre les éléments de DOM

Plusieurs sélecteurs partageant la même déclaration

```
①
span {
    border: red 2px solid;
}
div {
    border: red 2px solid;
}

②
span,
div {
    border: red 2px solid;
}

③
:is(span, div) {
    border: red 2px solid;
}
```

```
<section>

    <div>Div</div>
    <p>Paragraphe contenant <span>un &lt;span&gt; ici</span>
        et <span>un autre par ici</span>.</p>
</section>
```

① Duplication de déclaration avec des sélecteurs différents

② Plusieurs sélecteurs séparés par une virgule se partagent la même déclaration[1] [2]

③ Pseudo-classe CSS fonctionnelle `:is()`

Cette technique rend le code plus compact et **plus modulaire** en **évitant les répétitions**.

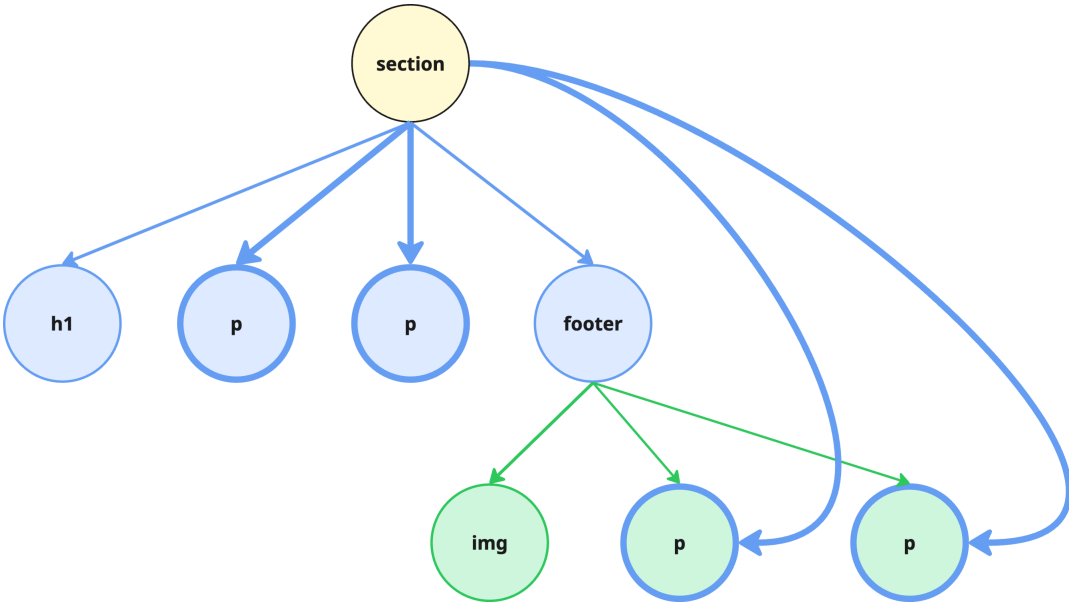
Privilégier la syntaxe de groupement avec `span , div { }`

> Lorsque plusieurs sélecteurs partagent les mêmes déclarations, ils peuvent être regroupés dans une liste séparée par des virgules. Les listes de sélecteurs peuvent aussi être passées en paramètre à certaines pseudo-classes CSS fonctionnelles. Des espaces peuvent apparaître avant et/ou après la virgule.

https://developer.mozilla.org/fr/docs/Web/CSS/Selector_list

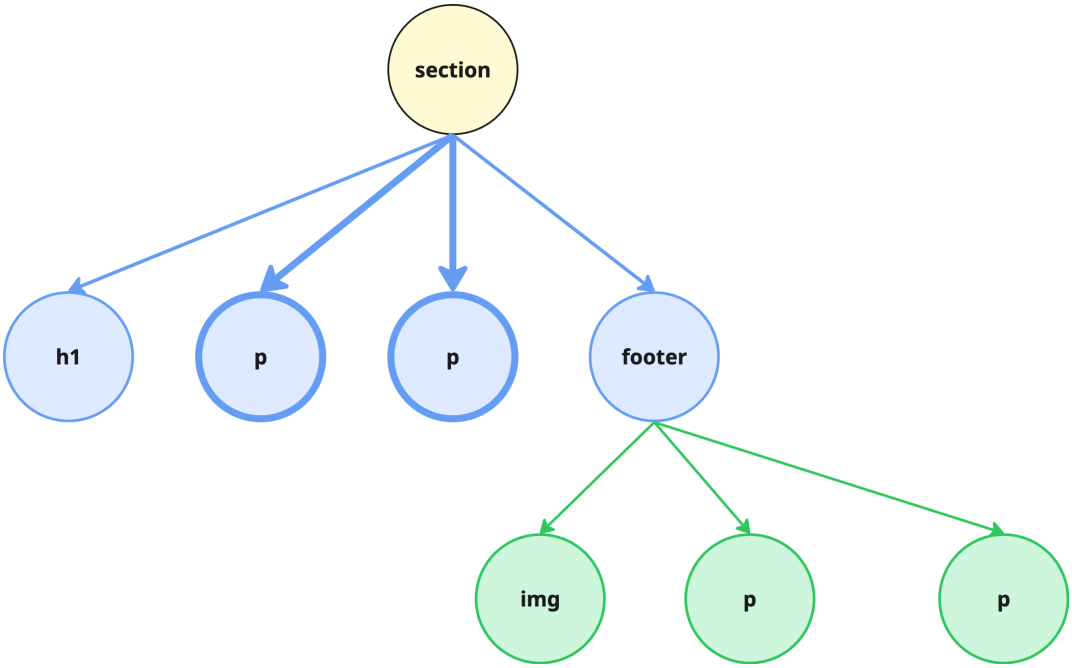
Descendants

- descendants : " " (espace)



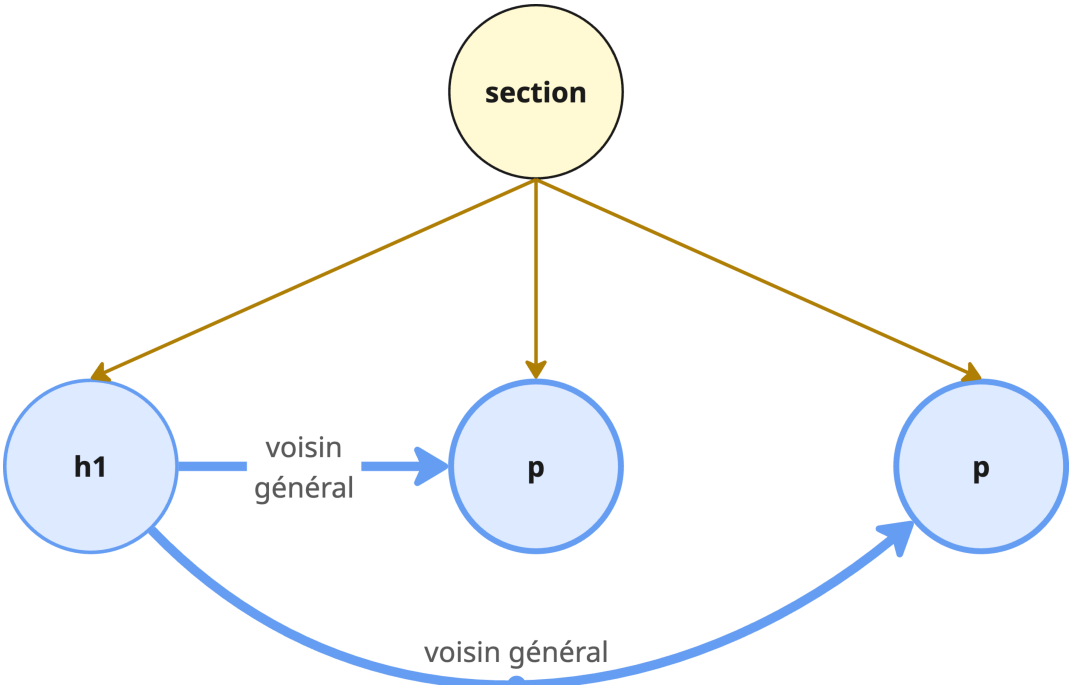
Enfants (direct)

- enfants : >



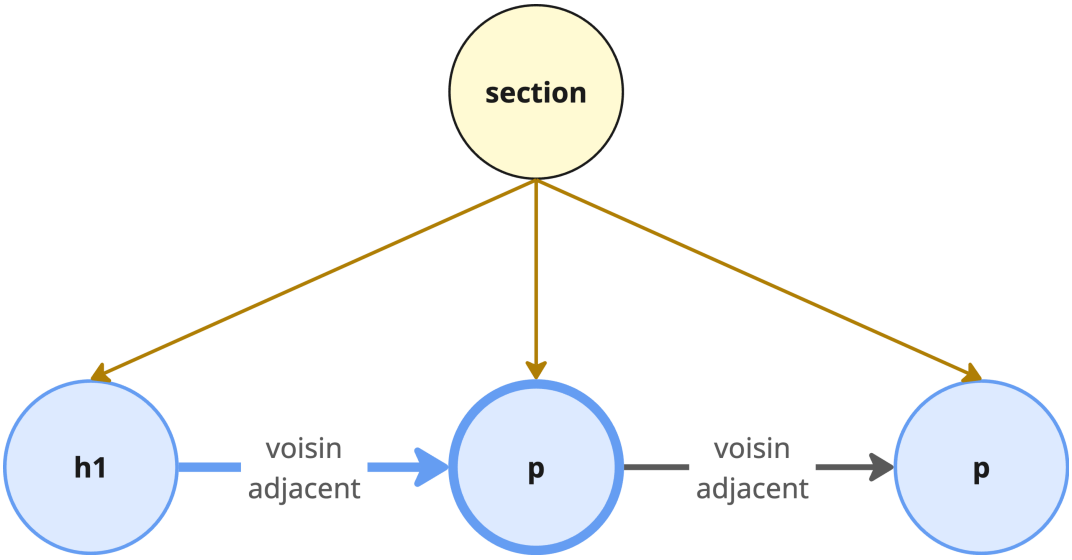
Voisins globaux

- voisins globaux : ~

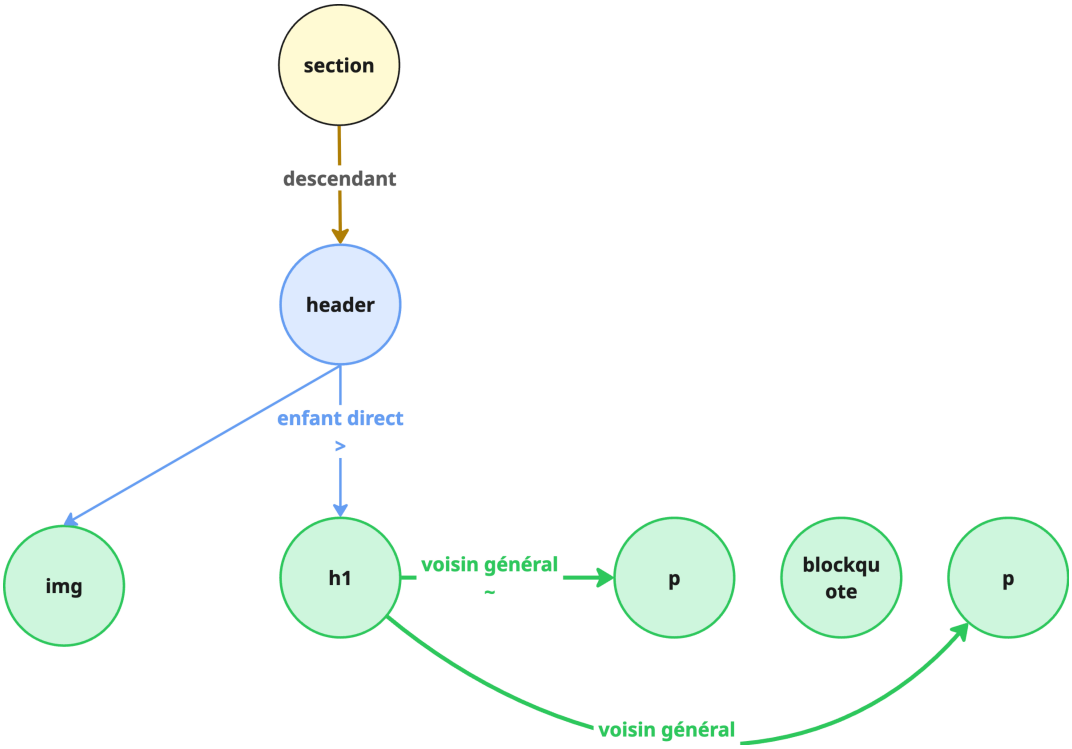


Voisin direct

- voisin direct : +



Exemple complexe




```
section header > h1 ~ p {
  color: blue;
}
```

```
<section>
  <header>
    <img src="#" alt="image" />
    <h1>Titre de la carte</h1>
    <p>paragraphe 1</p>
    <blockquote>Citation</blockquote>
    <p>paragraphe 2</p>
  </header>
    <p>paragraphe 3</p>
</section>
```

Sandbox [https://developer.mozilla.org/en-US/play?id=nbRq4UEH24%2BtzXnNFIOEoD6hAuQwydec8ukG5zYNaZM-jWNUjvNZPDqsF5OzBbeNtT8z1cNzoQIuN0YHF]

[1] On peut mettre les sélecteurs sur une seule ou plusieurs lignes

[2] Si un sélecteur est incorrect toute la déclaration l'est

Pseudo-classes

Les pseudo-classes [1] permettent de compléter un sélecteur pour styliser **leur état** ou **leur position dans l'arbre DOM**.

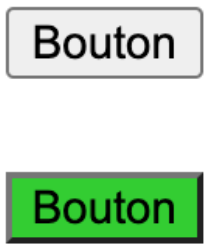
La liste étant très longue, seuls certaines pseudo-classes significatives sont présentes dans ce cours.

Les autres seront laissées à la discrétion des lecteur-ices.

Exemple d'utilisation de `:hover`

L'arrière-plan du bouton devient vert quand on passe la souris dessus

```
<style>
    button:hover{
        background:limegreen;
    }
</style>
<button>Bouton</button>
```



Exemple d'utilisation de `:visited`

Le lien devient rouge quand il a été visité

```
<style>
    a:visited{
        color:darkred;
    }
</style>

<a href="https://developer.mozilla.org">
    Documentation Mozilla
</a>
```

Documentation Mozilla

Documentation Mozilla

[1] MDN - CSS : pseudo-classes [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Pseudo-classes#pseudo-classes_de_liens]

Pseudo-classes d’actions utilisateur-ice

> Ces pseudo-classes [1] nécessitent une interaction de l’utilisateur·ice pour s’appliquer, comme le fait de maintenir un pointeur de souris au-dessus d’un élément.

Quelques exemples

- `:active` : L’état actif d’un élément (ex. cliquer sur un bouton)
- `:hover` : quand la souris passe au dessus de l’élément.
- `:focus` : quand un élément obtient le focus (ex. par navigation clavier ou quand on clique dans un champ de saisie)
- `:focus-within` : quand l’élément ou un de ses descendants a le focus

[1] MDN - Pseudo-classes d’action utilisateur-ice [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Pseudo-classes#pseudo-classes_daction_utilisateur%C2%B7ice]

Pseudo-classes de champs de formulaires

Relatifs à l'état d'éléments de formulaire [1] (ex. champs requis, désactivé, optionnel).

[1] MDN - CSS : Pseudo-classes d'entrée de formulaire [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Pseudo-classes#pseudo-classes_dentr%C3%A9e_de_formulaire]

Pseudo-classes fonctionnelles

> Ces pseudo-classes [1] acceptent une liste de sélecteurs ou une liste de sélecteurs indulgente comme paramètre.

- `:is()` correspond à tout élément qui correspond à l'un des sélecteurs de la liste fournie. La liste est tolérante.
- `:not()` négation, représente tout élément qui n'est pas représenté par son argument.
- `:where()` ajustement de spécificité correspond à tout élément qui correspond à l'un des sélecteurs de la liste fournie sans ajouter de poids de spécificité. La liste est tolérante.
- `:has()` représente un élément si l'un des sélecteurs relatifs correspond lorsqu'il est ancré par rapport à l'élément attaché.

`:has()`

`:has()` [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/:has] permet de sélectionner un élément si au moins un des sélecteurs passés en paramètre correspond à l'élément.

`:has()` : est une pseudo-classe intéressante car elle permet de contourner une limitation des sélecteurs CSS : sélectionner des **éléments parents** ou des **voisins qui précèdent** l'élément.

La pseudo-classe `:has()` [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/:has] fait partie des fonctionnalités plutôt récentes des navigateurs (baseline 2023). Elle ne fonctionnera probablement pas sur des navigateur plus anciens. Évitez de l'utiliser pour des styles ou des fonctionnalités essentielles de vos sites pour le moment.

Sélectionner un voisin précédent avec `:has()`

Sélection du voisin précédent *avec* `:has()` *pour ajouter une étoile rouge à un label quand le chaps de saisie qui le suit est requis.*

```
<style>
    label {
        display: block;
    }
    label:has(+ input:required)::after {
        content: " * ";
        color: red;
    }
</style>

<label for="answer">Réponse</label>
<input id="answer" type="text" name="answer" required>

<label for="comment">Commentaire</label>
```

```
<input id="comment" type="text" name="comment">
```

Sélectionner un parent avec :has()

Sélection d'une <section> parent si l'un de ses <input> descendant est invalide (ici un champ requis non rempli)

```
<style>
section{
    border: 1px solid grey ;
    border-radius: 1rem;
    padding: 1rem;
}
section:has( input:invalid){
    background-color: pink;
    border-color: red;
}
</style>
<section>
    <form>
        <label for="answer">Réponse</label>
        <input
            id="answer"
            type="text"
            name="answer"
            required >
    </form>
</section>
```

:not()

Sélectionne les élements qui ne correspondent pas aux sélecteurs en argument

Tous les paragraphes sans classe `.exemple` ont un texte bleu.

```
p:not(.example) {
  color: blue;
}
```

`:is()`

Groupe plusieurs sélecteurs. Similaire à la virgule.

```
/*
    Pour les liens dont l'attribut "href" commence par "http://" ou "https://".
    Ajoute un émoji  après les liens externes
*/
:is( a[href^="https://"], a[href^="http://"] )::after {
    content: " "
}
```

Peut être utile si on veut appliquer un pseudo-élément à plusieurs sélecteurs.

[1] MDN - CSS : Pseudo-classes fonctionnelles [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Pseudo-classes#pseudo-classes_-fonctionnelles]

Pseudo-classes structurelles d'arbre

Ces pseudo-classes [1] se rapportent à l'emplacement d'un élément dans l'arbre du document.

- `:root`
- `:empty`
- `:nth-child(n)`
- `:nth-last-child(n)`
- `:first-child`
- `:last-child`
- `:only-child`
- `:first-of-type`
- `:last-of-type`
- `:nth-of-type(n)`

`:root`

Représente un élément qui est la racine du document. Dans HTML, il s'agit généralement de l'élément <html>.

Souvent utilisé pour positionner des variables (custom properties) [2] à appliquer globalement.

Par exemple, une palette de couleurs.

```
<style>
    :root {
        --color-header-text: black;
        /* Couleur standard */
        /* --color-header-background: yellow; */
        /* Paramétrage temporaire lors d'Halloween */
        --color-header-background: orange;
    }
    header {
        color: var(--color-header-text);
        background-color: var(--color-header-background);
    }
</style>
<header>
    <h1>Bienvenue au BDE !</h1>
</header>
```

[1] MDN - CSS : Pseudo-classes structurelles d'arbre [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/Pseudo-classes#pseudo-classes_structurelles_darbre]

[2] Les custom properties seront abordées dans un chapitre ultérieur

Pseudo-éléments

> Un pseudo-élément CSS [1] est un mot-clé ajouté à un sélecteur qui vous permet de mettre en forme une partie spécifique du ou des éléments sélectionnés.

```
sélecteur::pseudo-élément {
  propriété: valeur;
}
```

```
<style>
    /* La première ligne de chaque élément <p>. */
    p::first-line {
        color: blue;
        text-transform: uppercase;
    }
</style>
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ut dui ut mauris feugiat venenatis. Lorem ipsum dolor
    sit amet, consectetur adipiscing elit. Donec posuere a diam eu consectetur. Cras pharetra elit a magna porttitor
    finibus. Pellentesque rhoncus id lorem in lacinia. Orci varius natoque penatibus et magnis dis parturient montes,
    nascetur ridiculus mus. Maecenas dignissim faucibus ullamcorper. Vestibulum condimentum lorem id scelerisque
    tincidunt. Fusce quis ultricies risus, id imperdiet erat. Nunc vel ligula bibendum, condimentum libero sit amet,
    congue odio. Pellentesque euismod egestas elementum. Integer et dignissim nisi.
</p>
<p>
    Aliquam erat volutpat. Nunc fermentum mauris eget diam tristique porttitor. Vestibulum sit amet eros sed mauris
    tempor pharetra vitae in magna. Donec magna erat, feugiat nec feugiat non, pellentesque aliquam turpis. Aliquam
    turpis nisl, feugiat maximus tincidunt ac, suscipit id est. In convallis eu felis a eleifend. Vivamus euismod
    fermentum leo, cursus condimentum tortor viverra vitae. Pellentesque viverra sem tristique tincidunt feugiat.
    Praesent efficitur aliquam facilisis.
</p>
```

LOREM IPSUM DOLOR SIT AMET,
consectetur adipiscing elit. Nullam ut dui ut mauris feugiat venenatis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec posuere a diam eu consectetur. Cras pharetra elit a magna porttitor finibus. Pellentesque rhoncus id lorem in lacinia. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas dignissim faucibus ullamcorper. Vestibulum condimentum lorem id scelerisque tincidunt. Fusce quis ultricies risus, id imperdiet erat. Nunc vel ligula bibendum, condimentum libero sit amet, congue odio. Pellentesque euismod egestas elementum. Integer et dignissim nisi.

ALIQUAM ERAT VOLUTPAT. NUNC
fermentum mauris eget diam tristique porttitor. Vestibulum sit amet eros sed mauris tempor pharetra vitae in magna. Donec magna erat, feugiat nec feugiat non, pellentesque aliquam turpis.

::before

> Le pseudo-élément CSS `::before` [2] crée un pseudo-élément qui sera le premier enfant de l'élément ciblé. Généralement utilisé pour ajouter du contenu esthétique à un élément via la propriété CSS content. Par défaut, l'élément créé est de type en-ligne (inline en anglais).

```
<style>
a[hreflang="fr"]::before{
  content:" "
}
</style>
<a href="https://wikipedia.fr" hreflang="fr">Wikipédia France</a>
```

Wikipédia France

::after

> Le pseudo-élément CSS `::after` [3] crée un pseudo-élément qui sera le dernier enfant de l'élément ciblé. Généralement utilisé pour ajouter du contenu esthétique à un élément via la propriété CSS content. Par défaut, l'élément créé est de type en-ligne (inline en anglais).

```
<style>
    /*
    Pour les liens dont l'attribut "href" commence par "http://" ou "https://".
    Ajoute un émoji  après les liens externes
    */
    :is( a[href^="https://"], a[href^="http://"] )::after {
        content: " "
    }
</style>
<ul>
    <li><a href="#un_lien_de_page"> Lien de page </a></li>
    <li><a href="http://lien_externe.com"> Lien externe non sécurisé </a></li>
    <li><a href="https://lien_externe.com"> Lien externe sécurisé </a></li>
</ul>
```

- Lien de page
- Lien externe non sécurisé
- Lien externe sécurisé

::first-letter

> Le pseudo-élément CSS `::first-letter` [4] sélectionne la première lettre de la première ligne d'un bloc, si elle n'est pas précédée par un quelconque autre contenu (comme une image ou un tableau en ligne) sur sa ligne.

```
<style>
    p::first-letter {
        font-size: 4em
    }
</style>
<p>Il était une fois, un petit chaperon rouge.</p>
<p>Fin</p>
```

Il était une fois, un petit chaperon rouge.

Fin

::first-line

> Le pseudo-élément CSS `::first-line` [5] applique la décoration à la première ligne d'un élément. La quantité de texte sur la première ligne dépend de nombreux facteurs, comme la largeur des éléments ou du document, mais aussi de la taille du texte.

```
<style>
    /* La première ligne de chaque élément <p>. */
    p::first-line {
        color: blue;
        text-transform: uppercase;
    }
</style>
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ut dui ut mauris
feugiat venenatis. Lorem ipsum dolor
    sit amet, consectetur adipiscing elit. Donec posuere a diam eu consectetur. Cras
pharetra elit a magna porttitor
    finibus. Pellentesque rhoncus id lorem in lacinia. Orci varius natoque penatibus
et magnis dis parturient montes,
    nascetur ridiculus mus. Maecenas dignissim faucibus ullamcorper. Vestibulum
condimentum lorem id scelerisque
    tincidunt. Fusce quis ultricies risus, id imperdiet erat. Nunc vel ligula
bibendum, condimentum libero sit amet,
    congue odio. Pellentesque euismod egestas elementum. Integer et dignissim nisi.
</p>
<p>
    Aliquam erat volutpat. Nunc fermentum mauris eget diam tristique porttitor.
Vestibulum sit amet eros sed mauris
    tempor pharetra vitae in magna. Donec magna erat, feugiat nec feugiat non,
pellentesque aliquam turpis. Aliquam
    turpis nisl, feugiat maximus tincidunt ac, suscipit id est. In convallis eu felis
a eleifend. Vivamus euismod
    fermentum leo, cursus condimentum tortor viverra vitae. Pellentesque viverra sem
tristique tincidunt feugiat.
    Praesent efficitur aliquam facilisis.
</p>
```

LOREM IPSUM DOLOR SIT AMET,
consectetur adipiscing elit. Nullam ut dui ut mauris feugiat venenatis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec posuere a diam eu consectetur. Cras pharetra elit a magna porttitor finibus. Pellentesque rhoncus id lorem in lacinia. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas dignissim faucibus ullamcorper. Vestibulum condimentum lorem id scelerisque tincidunt. Fusce quis ultricies risus, id imperdiet erat. Nunc vel ligula bibendum, condimentum libero sit amet, congue odio. Pellentesque euismod egestas elementum. Integer et dignissim nisi.

ALIQUAM ERAT VOLUTPAT. NUNC
fermentum mauris eget diam tristique porttitor. Vestibulum sit amet eros sed mauris tempor pharetra vitae in magna. Donec magna erat, feugiat nec feugiat non, pellentesque aliquam turpis.

[1] MDN - CSS : Pseudo-éléments [https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-elements,window=_blank]

[2] MDN - CSS : ::before [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/::before]

[3] MDN - CSS : ::after [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/::after]

[4] MDN - CSS : ::first-letter [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/::first-letter]

[5] MDN - CSS : ::first-line [https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Selectors/::first-line]