

# Les sélecteurs

## Attributs HTML

Il est possible de définir des "**points d'accroche**" sur un élément HTML grâce à deux types d'attributs :

- L'attribut `id` :
  - ne contient qu'**une seule valeur**,
  - cette valeur est **unique** à l'élément qui la porte,
  - il peut être **utilisé par le HTML** lui-même : liens internes (ancres), libellés de champs...
- L'attribut `class` :
  - un élément peut porter **plusieurs valeurs**,
  - une valeur peut être assignée à **plusieurs éléments**,
  - il n'est utilisé que pour la **sélection via CSS**.

# Les sélecteurs

## Attributs HTML

Chaque valeur peut être composée de lettres (majuscules et minuscules), de chiffres et/ou de tirets, mais doit impérativement commencer par une lettre.

# Les sélecteurs

## Types de sélecteur

### Les **sélecteurs simples** :

- Sélecteur **universel** : \*
- Par élément : <balise>
- Par **class** : .<valeur de l'attribut class>
- Par **id** : #<valeur de l'attribut id>

```
* {  
    font-style: italic;  
}
```

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

```
.myClass {  
    text-decoration: underline;  
}
```

```
#myId {  
    font-family: monospace;  
}
```

# Les sélecteurs

## Types de sélecteurs

Ces différents sélecteurs peuvent être combinés entre eux, pour former des **sélecteurs composés** :

```
p.myClass#myId {  
  font-size: 1.5rem;  
}
```

# Les sélecteurs

## Types de sélecteurs

Les **sélecteurs d'attributs** permettent de cibler un élément en fonction de la présence d'un attribut sur sa balise, et éventuellement de la valeur de cet attribut.

```
a[href], input[type="checkbox"]
```

Il est possible d'effectuer un filtrage plus fin de la valeur via des opérateurs logiques :

[https://developer.mozilla.org/fr/docs/Web/CSS/Attribute\\_selectors](https://developer.mozilla.org/fr/docs/Web/CSS/Attribute_selectors)

# Les sélecteurs

## Types de sélecteurs

Les **combinateurs CSS** permettent de créer des **sélecteurs complexes** :

- Le combineur de **descendance**, qui cible les éléments correspondants au deuxième sélecteur uniquement s'ils ont un ancêtre qui correspond au premier sélecteur :

```
ul.ma-liste li
```

- Le combineur de **descendance directe**, qui cible les éléments correspondant au deuxième sélecteur uniquement s'ils sont les descendants *directs* d'un élément correspondant au premier sélecteur :

```
ul.ma-liste > li
```

# Les sélecteurs

## Types de sélecteurs

Les **combinateurs de voisinage** :

- Le combineur de **voisin suivant**, qui cible les éléments correspondant au 2e sélecteur uniquement s'ils se trouvent après un élément correspondant au 1er sélecteur :



The image shows a CSS selector example for the 'voisin suivant' (next sibling) combinator. It consists of the text 'img ~ p' in a monospaced font, with 'img' and 'p' in red and the tilde '~' in white. The text is centered within a dark rectangular box.

- Le combineur de **voisin immédiat**, qui cible les éléments correspondant au deuxième sélecteur uniquement s'ils se trouvent *immédiatement* après un élément correspondant au premier sélecteur :



The image shows a CSS selector example for the 'voisin immédiat' (adjacent sibling) combinator. It consists of the text 'img + p' in a monospaced font, with 'img' and 'p' in red and the plus sign '+' in white. The text is centered within a dark rectangular box.

# Les sélecteurs

## Types de sélecteurs

Les **pseudo-classes** permettent de cibler un élément en fonction de :

- sa **position dans le DOM** : `:first-child`, `:first-of-type`, `:nth-child(2n)...`
- son **état**, par exemple :
  - s'il s'agit d'un **lien** : `a:hover`, `a:active`, `a:visited...`
  - s'il s'agit d'un **élément de formulaire** : `input:focus`, `select:invalid`,  
`[ type="checkbox" ]:checked...`
- Il est également possible d'**exclure un élément** correspondant à un sélecteur grâce à la pseudo-classe `:not(<selecteur>)`



# Les sélecteurs

## Types de sélecteurs

Les **pseudo-éléments** permettent quand à eux :

- le **ciblage d'une sous-partie** d'un élément, par exemple :

`h1::first-letter`

`p::first-line...`

- l'**injection de contenu** :

- `::before` (avant le contenu de l'élément),
- `::after` (après le contenu de l'élément).

Il existe encore d'autres sélecteurs plus spécifiques, permettant un ciblage plus fin et la gestion de cas complexes : <https://css4-selectors.com/selectors/>