R102

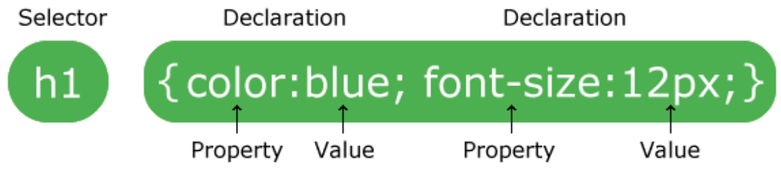
DÉVELOPPEMENT D'INTERFACES WEB

Jean Carle <jean.carle@univ-lille.fr>, 2025

CSS : CASCADING STYLE SHEET

- Mise en forme des contenus
- Notion de hiérarchie, priorité, héritage

SYNTAXE



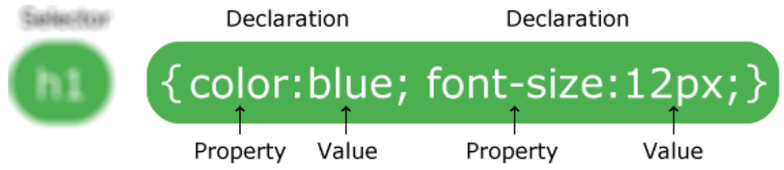
Définition : **Règle** = 1 sélecteur avec son bloc de **déclarations**

- En cas de doublon ou recouvrement de règles :
  - **Priorité à la dernière lue**
  - Si sélecteur différent : calcul de priorité (nommée *spécificité*)
- Commentaires

```
/* Commentaire de 1 ou
plusieurs lignes */
```

⚠ Ne pas oublier le `;` de fin de déclaration (optionnel uniquement sur la dernière déclaration d'un bloc)

DÉCLARATION



Chaque déclaration est composée de 3 éléments :

1. **Propriété** : Nom d'une caractéristique à modifier
   - https://www.w3schools.com/cssref/
   - https://developer.mozilla.org/fr/docs/Web/CSS/Reference
2. un séparateur : " `:` "
3. **Valeur** : Valeur pour la propriété associée
   - https://www.w3schools.com/cssref/css_units.asp
   - https://developer.mozilla.org/fr/docs/Learn/CSS/Building_blocks/Values_and_units

RÈGLES @

= Permet de fournir des informations ou des instructions au CSS sur la façon dont il doit se comporter ou réaliser certaines opérations.

C’est une sorte de configuration d’un ensemble de règles CSS.

- Commence par `@`, se termine par `;` ou par un bloc de règles CSS entre accolades.
- Exemple :
  - @charset "utf-8";
  - @import url("bluish.css") speech;

REQUÊTES MÉDIA

- @import url requête_média;
- @media requête_média;

```
@media screen and (min-width: 780px) and (max-width: 979px)
{ /* Liste des règles pour ce type d'écran */
    p {font-size: 150%;}
    ...
}

@media print
{ /* Liste des règles pour l'impression papier */
    p {font-size: inherit; …}
    ⋮
}
```

LES SÉLECTEURS

Cible où vont s’appliquer les déclarations du bloc associé.

Il existe de très nombreux sélecteurs.

- https://www.w3.org/Style/css3-selectors-updates/WD-css3-selectors-20010126.fr.html#selectors
- https://developer.mozilla.org/fr/docs/Web/CSS/Reference#s%C3%A9lecteurs

Testeur de sélecteur du W3school : https://www.w3schools.com/cssref/trysel.asp

SÉLECTEUR UNIVERSEL

Cible = tout

Permet l’initialisation de tous les éléments

css

```css
* {
    margin: 0;
    padding: 0;
    border: 0;
}
```

🚫 À éviter 🚫

SÉLECTEUR DE TYPE

Cible = un élément HTML (balise)

css

```
p { color: red; }
h6 { font-size: xx-large; }
```

html

```
<h6>Le titre de niveau 6</h6>
<p>Le paragraphe</p>
```

LE TITRE DE NIVEAU 6

Le paragraphe

SÉLECTEUR DE CLASSE

Cible tous les éléments possédant la classe donnée

css

```css
.surbrille {
    border: dashed yellow;
    color: red;
}

p.surbrille {
    border: solid lightgreen;
}
```

html

```html
<h3 class="surbrille">Un titre de niveau 3 encadré</h3>
<p>Un paragraphe simple</p>
<h3>Un simple titre de niveau 3</h3>
<p class="surbrille">Un paragraphe encadré</p>
```

UN TITRE DE NIVEAU 3 ENCADRÉ

Un paragraphe simple

UN SIMPLE TITRE DE NIVEAU 3

Un paragraphe encadré

Question: Pourquoi les deux paragraphes n’ont pas le même fond ?

SÉLECTEUR D'ID

Cible = un élément possédant l'identifiant donné

Au contraire de la classe, un ID ne peut être utilisé qu'une seule fois par document.

css

```css
#premier {
    background-color: lightgreen;
}

h3#entete {
    background-color: yellow;
    color: red;
}
```

html

```html
<h3 id="entete">Un titre identifié</h3>
<p>Un paragraphe simple</p>
<h3>Un simple titre de niveau 3</h3>
<p id="premier">Un paragraphe identifié</p>
```

UN TITRE IDENTIFIÉ

Un paragraphe simple

UN SIMPLE TITRE DE NIVEAU 3

Un paragraphe identifié

SÉLECTEUR D'ATTRIBUT

= Permet de cibler des éléments en fonction de la présence ou non d'un attribut.

css

```css
p[cat] { color: green; } /* l'attribut existe */
p[cat^="dog"] { color: blue; } /* commence par */
p[cat~="dog"] { color: red; } /* valeur dans la liste */
p[cat="dog"] { color: yellow; } /* valeur exacte */
```

html

```html
<p>Un paragraphe simple</p>
<p cat>Un paragraphe avec police verte</h3>
<p cat="matou dog chat">Un paragraphe avec police rouge<
<p cat="doggy">Un paragraphe avec police bleue</p>
<p cat="dog">Un paragraphe avec police jaune</p>
```

Un 1er paragraphe simple

Un paragraphe avec police verte

Un paragraphe avec police rouge

Un paragraphe avec police bleue

Un paragraphe avec police jaune

LES PSEUDO-CLASSES `:`

= Permet d'indiquer l'état spécifique dans lequel un élément ciblé doit être pour utiliser la mise en forme définie.

S'applique à un sélecteur pour appliquer des styles selon l'état, la position

- d'état : fonction de l'état de l'élément ciblé
- de position : compter et cibler les éléments pairs, impair, le nième, …
- d'information (meta données ou autres)

```
h1:hover {
  background-color: #F89B4D;
  font-variant-caps: small-caps;
}
p:nth-child( 2 ) { background-color: lightgreen; }
```

```
p:lang(fr) { ... }
a:visited { ... }
a:focus:hover { ... }
```

UN TITRE

Avec un premier paragraphe

Avec un second paragraphe

LES PSEUDO-ÉLÉMENTS ::

= mot-clé ajouté à un sélecteur pour mettre en forme certaines parties de la sélection ciblée.

```css
.demo::first-letter {
  color: red;
  font-size: 130%;
}
```

```html
<p class="demo">Un premier paragraphe</p>
<p>Un paragraphe normal</p>
<p class="demo">Un autre paragraphe</p>
```

Un premier paragraphe

Un paragraphe normal

Un autre paragraphe

COMBINAISONS DE SÉLECTEURS

- " `,` " Regroupement : `div,p` = `div` et `p` partage les mêmes déclarations.
- " `+` " Voisin adjacent : `div+p` = cible le `p` qui suit immédiatement un `div` et qui a le même élément parent
- " `~` " Voisins : `div~p` = cible tous les `p` qui suivent (immédiatement ou non) un `div` et qui ont le même élément parent
- " `>` " Enfants : `div>p` = cible tous les `p` qui ont un `div` comme parent direct
- " ` ` " Descendants : `div p` = cible tous les `p` qui ont un `div` comme ancêtre

INTÉGRATION DES STYLES

Trois méthodes d’intégration

1. en ligne
2. interne
3. externe

STYLE EN LIGNE

Directement dans les balises html au moyen de l'**attribut** `style`

```
<p  style="color: blue;">Un paragraphe bleu</p>
<p  style="color: red; font-style: oblique 40deg;">Un autre rouge penché</p>
```

Un paragraphe bleu

Un autre rouge penché

STYLES INTERNES

Centralisé dans l'en-tête (head) avec la **balise** `style`

```
<head>
    …
    <style>
        p {
            color : blue;
            margin: 0;
        }
        strong {
            display:block;
            color: red;
            font-style: oblique 40deg;
        }
    </style>
</head>
<body>
```

p1 : Les attributs

style

sont prioritaires

p2 : sur les styles internes.

p3 : Un 3e paragraphe

STYLES EXTERNES

```
<head>
    …
    <link rel="stylesheet" href="mon-fichier.css">
    <!-- Contient les règles précédentes -->
</head>
<body>
    <p>p1 : Les attributs <strong>style</strong> sont prioritaires</p>
    <p style="color:green" class="tomate">p2 : sur les styles internes.</p>
    <p>p3 : Un 3e paragraphe<p>
</body>
```

p1 : Les attributs ***style*** sont prioritaires

p2 : sur les styles internes.

p3 : Un 3e paragraphe

INTÉGRATION DES STYLES

Trois méthodes d'intégration

1. en ligne : 🚫 Solution à proscrire au maximum
2. interne : ❗ Solution à éviter, mais pratique parfois
3. externe : 👍 Solution la plus propre

Cas particulier avec la règle média `@import`

```
<head>
    <style>
        @import url("./css/mon-fichier.css");
    </style>
</head>
```

FUSION, HÉRITAGE ET SPÉCIFICITÉ

Comment comprendre ce qu’il se passe lorsque

- Il y en a partout (plusieurs fichiers)
- Quand des propriétés sont ciblées plusieurs fois avec des valeurs différentes

FUSION DE STYLES

= Un élément applique plusieurs propriétés différentes venant de règles différentes.

css

```css
p {color: red;}
.monstyle {font-size: 200%;}
.monstyle2 {font-style: italic;}
```

html

```html
<p>Un paragraphe rouge</p>
<div class="monstyle">Un bloc de texte important</div>
<p class="monstyle">Cette phrase est la fusion de 2 styles</p>
<p class="monstyle monstyle2">Cette phrase est la fusion de 3 styles</p>
```

Un paragraphe rouge

Un bloc de texte important

Cette phrase est la fusion de 2 styles

Cette phrase est la fusion de 3 styles

HÉRITAGE DE STYLES

Un élément peut se voir attribuer des valeurs de propriétés sans les avoir définis explicitement.
Elles proviennent des éléments "*parents*".

css

```css
p {
    background-color: lightblue;
    font-style: italic;
}
em {
    color: green;
    font-weight : bolder;
}
```

html

```html
<p>
    Paragraphe contenant une partie
    <em>mise en avant imbriqué dans le paragraphe</em>
    sans changer le fond.
</p>
```

*Paragraphe contenant une partie **mise en avant imbriqué dans le paragraphe** sans changer le fond.*

PRIORITÉ / CALCUL DE LA SPÉCIFICITÉ

Voici 3 règles qui seront appliquées au même élément :

css

```css
#demo {color: blue;}
.test {color: green; background-color: lightblue;}
p {color: red; background-color: lightgreen;}
```

html

```html
<p id="demo" class="test">Hello World!
```

Quelle sera la couleur du texte et du fond ?

CALCUL DE LA SPÉCIFICITÉ

Il existe 5 niveaux de priorité

Du ciblage le plus fort au plus faible :

- **[1]** 0 0 0 0 : Présence du mot-clé `!important` (utilisation à proscrire au maximum)
- 0 **[1]** 0 0 0 : Les styles *en ligne* (attribut *style="…"*)
- 0 0 **[1]** 0 0 : Les **ID**s
- 0 0 0 **[1]** 0 : Les **attributs**, **classes**, pseudo-classes
- 0 0 0 0 **[1]** : Les **éléments** (balises), pseudo-éléments

- La valeur la plus grande est prioritaire.
- Si égalité sur un niveau, on compte le nombre d'éléments du niveau inférieur.
- Si égalité totale, alors le dernier lu gagne.

Calculatrice : https://specificity.keegan.st

PRIORITÉ / CALCUL DE LA SPÉCIFICITÉ

Retour sur notre exemple.

css

```css
/* specificité = 100 */
#demo {color: blue;}

/* spécificité = 10 */
.test {color: green; background-color: lightblue;}

/* spécificité = 1 */
p {color: red; background-color: lightgreen;}
```

html

```html
<p id="demo" class="test">Hello World!</p>
```

Hello World!