

# Les bases du CSS

Ibrahim ALAME

29 janvier 2024

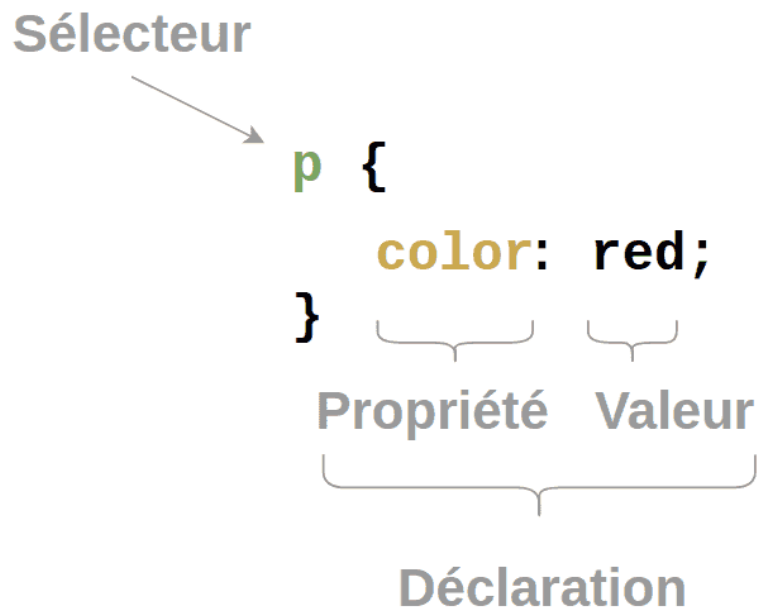
## 1 Présentation du chapitre

### 1.1 Rappels sur le CSS

Les feuilles de style en cascade, pour **CSS (Cascading Style Sheets)** est un langage qui décrit la présentation des documents **HTML**. Le principe du **CSS** est de sélectionner certains éléments pour leur appliquer un style particulier.

### 1.2 Les règles CSS

Tout le **CSS** se divise a minima de cette manière si il n'est pas appliqué côté **HTML** (ce qui est déconseillé comme nous le verrons) :



Le sélecteur permet de sélectionner le contenu **HTML** sur lequel la règle doit s'appliquer. Ici, par exemple, la règle s'appliquera à tous les éléments **HTML** **p**, c'est-à-dire à tous les paragraphes. Après le sélecteur, nous avons forcément une paire d'accolades qui contiennent la ou les propriétés à modifier pour les éléments sélectionnés.

- La propriété est ce que l'on souhaite définir sur l'élément. Par exemple ici sa couleur. Il faut toujours utiliser : pour séparer la propriété et la ou les valeurs à lui donner.

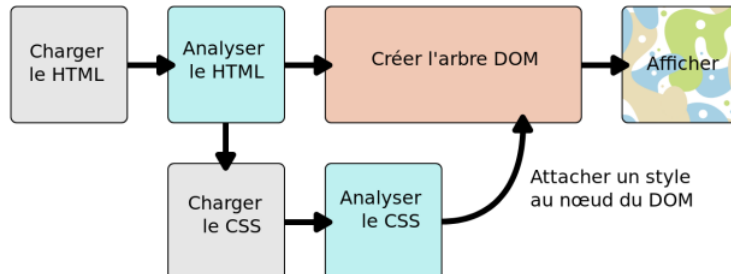
- La valeur est la mise en forme à appliquer pour une propriété donnée. Ici, de mettre la couleur en rouge. Il faut toujours terminer par ; pour passer à la modification suivante, par exemple :

```
p {  
  color: red;  
  width: 100px;  
}
```

### 1.3 Comment le CSS fonctionne t-il ?

Pour afficher un document le navigateur va réaliser les principales étapes suivantes :

1. Il va charger le code HTML avec une requête HTTP.
2. Il va convertir le HTML en objet l'DOM (document object model). Il s'agit du modèle en arbre de tous les éléments HTML de la page qui est mis en mémoire vive pour pouvoir le modifier dynamiquement. Nous y reviendrons
3. Le navigateur va récupérer toutes les ressources contenues dans les liens indiqués dans le HTML dont les feuilles de style CSS. Il effectue donc des requêtes HTML supplémentaires.
4. Le navigateur analyse les règles CSS pour créer un arbre de rendu (ou render tree) qui contient tous les styles à appliquer aux différents sélecteurs.
5. Enfin le navigateur affiche le document en prenant donc en compte le HTML et le CSS (et éventuellement le JavaScript, nous y reviendrons plus tard). Il s'agit de la phase de peinture (painting).



## 2 Déclarer des règles CSS

Il existe trois manières d'appliquer du CSS à un document HTML. Cependant une seule est recommandée et nous la verrons en dernier.

### 2.1 Styles en ligne

La première méthode est déconseillée mais vous serez amené à la voir souvent. Il s'agit de mettre du CSS directement dans le HTML. Par exemple :

```
<p style="color:red;">Element paragraphe avec du style CSS en ligne.</p>
```

Ce n'est pas bon pour de nombreuses raisons :

- La première est qu'en programmation, un des concepts fondamentaux est la séparation des préoccupations. Cela signifie qu'il ne faut pas mélanger du code qui n'ont pas du tout les mêmes finalités. Ici le **HTML** est pour la structure du **document** et le **CSS** pour la mise en forme du **document**. Ils n'ont pas la même finalité et doivent donc être séparés.
- Deuxième raison, la lisibilité et la maintenabilité du code s'en trouve fortement réduite car la combinaison de règles devient compliquer à comprendre et à mettre à jour.
- Troisième raison, l'un des autres concepts très important en programmation est de rester **DRY (don't repeat yourself)**. Cela signifie qu'il faut éviter au maximum de répéter le même code à différents endroits.

Ici, si vous voulez appliquer le même style à plusieurs paragraphes qui sont dans différents fichiers **HTML** ou dans des parties différentes vous devrez vous répéter. Cela aura pour conséquence de diminuer la maintenabilité du code car lors d'une mise à jour il faudra se rappeler tous les endroits où vous avez mis des règles **CSS**.

La seule exception est pour les emails car les clients de messagerie sont plus nombreux et moins bien développés que les navigateurs. Pour s'assurer que le **CSS** soit bien compatible avec tous les clients de messagerie, il est commun de mettre les règles **CSS** en ligne.

## 2.2 Feuilles de style interne

Une feuille de style interne est l'utilisation de règles **CSS** à l'intérieur de balises `<style>` dans le fichier **HTML** dans la partie `<head>`. Pour les mêmes raisons, ce n'est pas recommandé.

Voici un exemple :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Appliquer des styles CSS</title>
    <style>
      div {
        color: blue;
      }
    </style>
  </head>
  <body>
    <div>Div avec feuille de style interne</div>
  </body>
</html>
```

## 2.3 Feuilles de style externe

Cette méthode permet l'utilisation de fichiers **CSS** externes. C'est la méthode recommandée. Vous pouvez facilement appliquer les mêmes styles dans plusieurs pages **HTML** en important la même feuille de styles dans les différents fichiers. Il suffit de créer un fichier **CSS** avec l'extension **.css** puis de l'inclure avec un élément **link** :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Feuille de styles externe</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <p>Un paragraphe avec du contenu.</p>
  </body>
</html>
```

Dans le fichier CSS, vous pouvez mettre directement les règles :

```
h1 {
  color: blue;
}

p {
  color: red;
}
Copier
```

Voici un exemple avec les trois méthodes pour appliquer des styles **CSS** :

## 3 Les classes, les ids et les sélecteurs

### 3.1 Les sélecteurs CSS

Nous avons déjà vu ce qu'était un sélecteur : il permet de sélectionner un ou plusieurs éléments **HTML** sur lesquels appliquer une ou plusieurs règles **CSS**.

Cette leçon va être plus détaillée que la vidéo pour vous donner un aperçu approfondi des règles. Nous vous conseillons de les survoler pour le moment, nous y reviendrons au fur et à mesure dans les chapitres suivants.

Il existe différents types de sélecteurs : le sélecteur universel, les sélecteurs de type, les classes, les ids, les sélecteurs d'attribut, les combinateurs et les pseudo-classes.

Nous allons tous les voir sauf les pseudo-classes que nous verront plus tard.

### 3.2 Le sélecteur universel \*

Le sélecteur universel permet d'appliquer des règles à tous les éléments de la page.

```
* {  
  color: red;  
}
```

Vous vous en servirez rarement.

### 3.3 Les sélecteurs de type

Les sélecteurs de type CSS des éléments en fonction du nom de leur type.

Lorsqu'un sélecteur de type est utilisé seul, il ciblera tous les éléments de ce type.

Ainsi pour mettre tous les paragraphes en rouge :

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

### 3.4 Les classes

Le sélecteur classe commence par un point . suivi du nom de la classe.

Il sélectionnera tous les éléments HTML qui ont un attribut class qui inclut le nom de la classe.

Un élément ayant un attribut class peut avoir plusieurs classes qui sont alors séparées par des espaces.

Par exemple côté HTML si nous avons :

```
<p class="rouge">Paragraphe 1.</p>  
<p class="gras rouge">Paragraphe 2.</p>  
<p class="gras">Paragraphe 3.</p>  
<p>Paragraphe 4.</p>
```

Et côté CSS :

```
.rouge {  
  color: red;  
}  
.gras {  
  font-weight: bold;  
}
```

Alors seuls les deux premiers paragraphes qui ont la classe rouge seront sélectionnés et se verront appliquer la règle CSS color : red.

Les deuxième et troisième paragraphes qui ont la classe gras seront sélectionnés et se verront appliquer la règle CSS font-weight : bold.

### 3.5 Les ids

Le sélecteur ID (identifiant) permet de cibler un élément grâce à la valeur de son attribut id. Un sélecteur Id commence par # suivi du nom de l'identifiant. Par exemple côté HTML si nous avons :

```
<p id="par1">Paragraphe 1.</p>
```

Et côté CSS :

```
#par1 {  
  color: red;  
}
```

### 3.6 Les sélecteurs d'attribut

Le sélecteur d'attribut permet de cibler un élément selon la présence d'un attribut ou selon la valeur donnée d'un attribut.

Nous donnons quelques exemples mais n'entrons pas dans les détails car ils ne sont pas fréquemment utilisés :

```
/* Sélectionne les éléments a avec un attribut title */  
a[title] {  
  color: red;  
}
```

Cela sélectionnerait :

```
<a href="#" title="test">hello</a>
```

Autre exemple :

```
a[href="https://dyma.fr"] {  
  color: red;  
}
```

Sélectionnerait :

```
<a href="https://dyma.fr">Dyma</a>
```

### 3.7 Le groupement

Il est possible de grouper plusieurs sélecteurs en les séparant par une virgule pour appliquer les mêmes règles à un ensemble de sélecteurs.

Par exemple :

```
div, span {  
  color: red;  
}
```

Sélectionnerait les deux premiers éléments :

```
<div>Je suis rouge</div>
<span>Je suis aussi rouge</span>
<p>Je ne suis pas rouge</p>
```

## 3.8 Les combinateurs

Il existe des combinateurs qui permettent de définir finement les relation entre les sélecteurs pour un ensemble de règles.

### 3.8.1 Le combinateur de descendance

Lorsque que l'on met un espace entre deux sélecteurs cela signifie que l'on veut sélectionner les éléments sélectionnés par le second sélecteur et qui sont imbriqués dans les éléments sélectionnés par le premier sélecteur.

Cela à l'air dur dit comme cela, mais c'est très simple ! Prenons quelques exemples :

```
div span {
  color: red;
}
```

Donnera :

```
<div>Je ne suis pas rouge</div>
<span>Je ne suis pas rouge</span>
<div>
  <span>
    Je suis rouge
  </span>
</div>
```

### 3.8.2 Le combinateur enfant

Le combinateur `>` cible seulement les éléments correspondant au second sélecteur qui sont imbriqués directement dans les éléments ciblés par le premier sélecteur.

```
div > span {
  color: red;
}
```

Donnera :

```
<div>
  <span>Je suis rouge</span>
</div>
```

```

    <p><span>Pas rouge</span></p>
  </div>
</div>
<span>Pas rouge</span>

```

Voici des exemples pour bien distinguer combinateur enfant et combinateur descendant :

### 3.8.3 Le combinateur de voisin direct

Pour sélectionner un élément uniquement si celui-ci suit un élément donné et que les deux éléments sont imbriqués dans le même élément parent, alors on utilise un `+` entre les deux sélecteurs.

```

div + span {
  color: red;
}

```

Donnera :

```

<div>Je ne suis pas rouge</div>
<span>Je suis rouge</span>
<span>Je ne suis pas rouge</span>

```

### 3.8.4 Le combinateur de voisins généraux

Le combinateur cible seulement les éléments correspondant au second sélecteur qui sont précédés par un élément ciblé par le premier sélecteur.

Les éléments doivent être de même niveau, c'est-à-dire être imbriqués dans le même parent.

Vous trouverez le caractère tilde en faisant **Alt Gr + ^**.

Voici un exemple pour bien distinguer avec le combinateur voisin :

## 4 Utiliser l'inspecteur des navigateurs

Dans cette leçon nous utiliserons les outils Chrome DevTools, mais il existe à peu près les mêmes outils sur Firefox.

### 4.1 Ouvrir l'inspecteur Chrome

Pour ouvrir l'inspecteur sélectionnez un élément puis faites clic droit puis inspecter.

Vous pouvez également sélectionner l'élément et faire **Ctrl + Maj + i** :





## 4.2 Changer l'emplacement de l'inspecteur

Ouvrez le panel en faisant **Ctrl + Maj + p** et entrez la position : **right** ou **bottom** et pressez entrée :



Vous pouvez ensuite voir les styles appliqués à un élément dans l'onglet **Styles**.

Vous pouvez également utiliser l'onglet **Elements** pour sélectionner d'autres éléments et voir les styles qui leur sont appliqués.

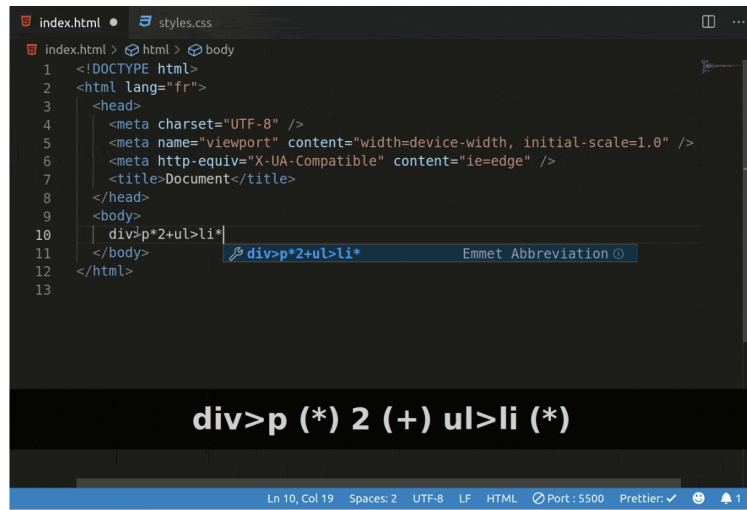
## 4.3 De nouveaux raccourcis Emmet

Nous avons vu **\*** et **;** avec Emmet.

Pour rappel le premier permet de multiplier l'élément précédent par le chiffre passé après.

Le second permet d'imbriquer un élément dans le premier.

Nous allons voir **+** qui permet d'ajouter un élément au même niveau :



The screenshot shows a code editor with two tabs: 'index.html' and 'styles.css'. The 'index.html' tab is active, displaying the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7     <title>Document</title>
8   </head>
9   <body>
10    div>p*2+ul>li*|
11  </body>
12 </html>
13
```

An Emmet abbreviation suggestion box is visible on line 10, showing the text 'div>p\*2+ul>li\*' and the label 'Emmet Abbreviation'.

Below the code editor, the text **div>p (\*) 2 (+) ul>li (\*)** is displayed.

The status bar at the bottom of the editor shows: Ln 10, Col 19 | Spaces: 2 | UTF-8 | LF | HTML | Port: 5500 | Prettier: ✓ | 1