Les fiches récap de l'école O'clock

Css

La syntaxe CSS

Dernière modification: 18 décembre 2022

La syntaxe CSS

Structure d'une règle CSS

```
selecteur{
  propriete: valeur;
}
```

- **selecteur** : Cible (élément HTML)
- **declaration** : couple propriété-valeur
- **propriété** : Ce que l'on souhaite modifier
- **valeur** : Comment souhaite-t-on le modifier

Mise en pratique

```
<p>Lorem ipsum dolor sit amet</p>
```

```
p {
  background: black;
}
```

Les sélecteurs peuvent être cumulés grâce à une `,` et une règle CSS peut également avoir plusieurs ensemble `propriete: valeur;`

```
p,
h2 {
  background: black;
  color: white;
}
```

Commentaires

Les commentaires en CSS sont déclarés avec `/* ... */`

```
/* un commentaire en CSS */
p,
h2 {
  background: black;
  color: white;
}
/*
Idéal pour laisser des instructions
ou des rappels
*/
```

Sélécteurs

Balises

Cible un élément par le nom de la balise

```
a { /* Tous les liens */ }
p { /* Tous les paragraphes */ }
li { /* Tous les éléments de liste */ }
```

IDs

En se basant sur l'attribut HTML `id` il est possible de cibler l'élément grâce à `#` en CSS

```html
<div id="chapeau">Le chapeau de l'article</div>
```

```css
#chapeau {
  color: blue;
}
```

Classes

En se basant sur l'attribut HTML `class` il est possible de cibler les éléments grâce à `.` en CSS

```html
<div class="important">Un texte très important</div>
```

```css
.important {
  color: red;
}
```

Classes multiples

Il est possible de mettre plusieurs classes pour un seul et même élément. Il suffit pour cela de mettre un espace entre les classes dans l'attribut class :

```
class="classe1 classe2 classe3"
```

C'est très utile lorsque plusieurs éléments ont les même styles sauf quelques propriétés.
On définit les styles communs sur une classe commune, et on définit les styles différents, sur une classe spécifique.

```html
<div class="important rouge"> Un texte très utile et rouge </div>
<div class="important jaune"> Un texte très utile et jaune </div>
```

```
.important {
  font-weight: bold;
  font-size: 20px;
}
.important.rouge {
  color: red;
}
.important.yellow {
  color: yellow;
}
```

Sélecteur descendant

En utilisant un espace :
https://developer.mozilla.org/fr/docs/Web/CSS/S%C3%A9lecteurs_descendant

```
header a { /* ... */ }
```

Les `<a>` se trouvant dans un `<header>`

```
<header>
  <a href="">Je serai ciblé</a>
  <p>
    Contenu de paragraphe
    <a href="">Je serai également ciblé</a>
  </p>
</header>
```

Sélecteur enfant

En utilisant le symbole `>` :
https://developer.mozilla.org/fr/docs/Web/CSS/S%C3%A9lecteurs_enfant

```
header > a { /* ... */ }
```

Les `<a>` étant **directement** enfants de `<header>`

```
<header>
  <a href="">Je serai ciblé</a>
  <p>
    Contenu de paragraphe
    <a href="">Je ne serai pas ciblé</a>
  </p>
</header>
```

Sélecteur voisin direct

En utilisant le symbole `+` :
https://developer.mozilla.org/fr/docs/Web/CSS/S%C3%A9lecteur_de_voisin_direct

```
h2 + p { /* ... */ }
```

Un `<p>` étant **directement** voisin de `<h2>`

```
<header>
  <h2>titre</h2>
  <p>paragraphe ciblé</p>
  <p>paragraphe ignoré</p>
  <p>paragraphe ignoré</p>
</header>
```

Sélecteur d'attribut

En utilisant la notation `[attribut]` :
https://developer.mozilla.org/fr/docs/Web/CSS/S%C3%A9lecteurs_d_attribut

```
input[type="text"] { /* ... */ }

a[href="http://exemple.com"] { /* ... */ }
```

Combinaisons

En combinant les différents sélecteurs existant il est possible de cibler des éléments très précisément

Dans toutes les `<div>` , trouve les `<p>` avec l'attribut `class` *important*

```
div p.important{ /* ... */ }
```

Trouve l'élément possédant l'attribut `id` puis trouve tous ces `<h2>`, à l'intérieur, trouve toutes les `<span>` avec l'attribut `class` *note*

```
#chapeau h2 span.note{ /* ... */ }
```

Pseudo-classes

Permet de cibler des éléments en fonction de leur état (survolé, mis en focus, etc...)

```
a {
  color: black;
}
a:hover {
  color: red;
}
```

Lorsqu'un lien sera survolé, son texte deviendra rouge

> Pour en apprendre plus sur les Pseudo-classes, cette page présente d'autres exemples ou bien MDN propose une liste exhaustive des pseudo-classes disponibles

Héritage

Le principe est simple, la plupart des propriétés relatives au **texte**, passeront d'un élément parent à ses enfants

- font-size
- font-weight
- font-family
- line-height

```
body {
  font-size: 16px;
}
/* les `p` hériterons de la valeur de `font-size` car ils sont les enfants
de `body` */
p {}
```

Priorité ou 'c'est moi le plus fort'

Tous les sélecteurs n'ont pas la même force

```
<p id="chapeau" class="important">Lorem ipsum dolor sit amet</p>
```

```
#chapeau { color: blue; }
.important { color: red; }
p { color: green; }
```

On aurait tendance à penser que la dernière règle l'emporterait mais dans la pratique ici notre paragraphe sera de couleur bleue. `color` vaudra `blue` car bien que `p {}` ou `.important {}` soit déclaré après `#chapeau {}` , ils sont plus 'faibles'.

Donc ici, `#chapeau { color: blue; }` est plus que `.important { color: red; }` qui lui même est plus fort que `p { color: green; }`

> **`id > class > <balise>`**

Unités

Couleurs

Afin d'exprimer une couleur en CSS il existe plusieurs solutions

Couleurs nommées

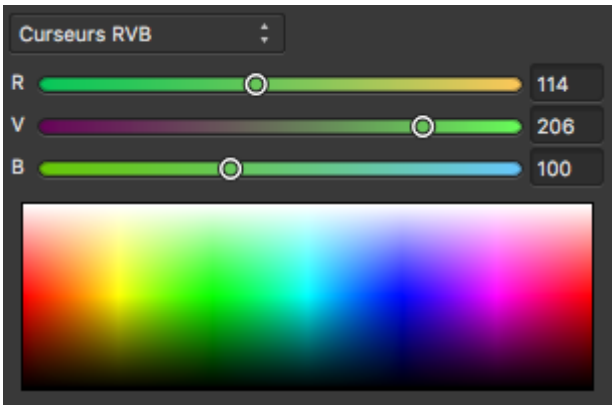
- `red`
- `blue`
- `black`
- `white`

- ...

La liste fournie par MDN contient plus d'une centaine de références; à chaque mot-clé correspond une couleur avec une valeur RVB associée.

RGB & RGBa

RGB ou Red Green Blue



256 valeurs possible dans le Rouge, le Vert et le Bleu soit la bagatelle de **16 777 216** de combinaisons et donc de couleurs possible

En CSS, le RGB s'utilise comme ceci

```
p { color: rgb(114, 206, 100); }

/* Noir */
p { color: rgb(0, 0, 0); }

/* Blanc */
p { color: rgb(255, 255, 255); }

/* Rouge */
p { color: rgb(255, 0, 0); }
```

Il est possible d'exprimer une notion d'alpha (transparence) avec la variante `rgba`. L'alpha se note de 0 à 1.

```
p { color: rgba(114, 206, 100, 0.8); }
```

Hexadécimal

Bien que la notation hexadécimale puisse être déroutante elle est en réalité extrêmement simple à comprendre.

Partant du principe

Base 2 **Binaire**	0	1											
Base 10 **Décimale**	0	1	2	3	4	5	6	7	8	9			
Base 16 **Hexadécimale**	0	1	2	3	4	5	6	7	8	9	A	B	C	D

L'hexadécimal s'exprime avec 16 valeurs possibles et non 10.

Décimale	**Hexadécimale**
0	0
3	3
8	8
13	D
16	10
20	14
30	1E
100	64
150	96
200	C8
255	FF

Même si cette notion n'est pas indispensable, ça peut toujours être utile lors d'un dîner « Absolument, `64 + 64 = C8` ! »

Les couleurs hexadécimales expriment donc du RGB mais sous un autre format

Blanc en RGB `255, 255, 255` en hexadécimal `#FFFFFF`

Toujours pas clair ?

Blanc	Valeur R	Valeur G	Valeur B
RGB	255	255	255
#	FF	FF	FF

Mieux non ?

```
/* Blanc */
p { color: #FFFFFF; }

/* Rouge */
p { color: #FF0000; }

/* Noir */
p { color: #000000; }
```

Évidement, il est inutile de recomposer les couleurs de tête, les couleurs hexadécimales sont très répandues.

La notation hexadécimale permet aussi de gérer la transparence d'une couleur.

🤸 **Pour aller plus loin :** https://developer.mozilla.org/fr/docs/Web/CSS/color_value

---

Tailles

Il est possible d'exprimer les tailles grâce à plusieurs unités

Les unités absolues :

pixels (px)

Cette unité permet de fixer des tailles, des positions ou des espacements par rapport à la résolution de l'affichage.

```
p{
  border-bottom: black solid 2px;
  font-size: 16px;
  margin-top: 20px;
  width: 300px;
}
```

Les unités relatives :

%

Unité relative à la taille de l'élément parent :

```
div{
  width: 300px;
}
div p{
  width: 50%;
  /* la largeur dépend de la largeur du parent, ici elle fera 150px (50% de 300px)*/
}
```

em

Unité relative à la valeur du `font-size` du parent :

```
body{ font-size: 20px }
h1{ font-size: 2em;} /* 40px (2 fois 20px hérité de body) */
h2{ font-size: 1.5em;} /* 30px (1,5 fois 20px hérité de body) */
p{ font-size: 0.75em;} /* 16px (75% de 20px hérité de body) */
```

rem

Unités relative à la valeur du `font-size` de l'élément racine (le plus souvent, la balise html) :

```
html{ font-size: 10px; }
body{ font-size: 2rem } /* 20px (2 fois 10px hérité de html) */
h1{ font-size: 2rem;} /* 20px (2 fois 10px hérité de html) */
h2{ font-size: 1.5rem;} /* 15px (1,5 fois 10px hérité de html) */
p{ font-size: 0.75rem;} /* 7.5px (75% de 10px hérité de html) */
```