Le CSS complet

Ce cours couvre les principales propriétés CSS avec des explications claires et des exemples concrets pour chaque propriété.

Background

Les propriétés `background` permettent de définir l'apparence de l'arrière-plan d'un élément.

`background`

Définition complète de l'arrière-plan avec une seule propriété. Vous pouvez spécifier l'image, la couleur, la position, etc.

Exemple :

```
background: #f3f3f3 url("image.jpg") no-repeat center center fixed;
```

`background-color`

Spécifie la couleur de fond de l'élément.

Valeurs possibles :

- Noms de couleurs (`red`, `blue`, etc.)
- Valeurs hexadécimales (`#ff0000`)
- `rgba()` pour inclure l'opacité.

Exemple :

```
background-color: rgba(255, 0, 0, 0.5);
```

`background-image`

Permet d'ajouter une image en arrière-plan.

Valeurs possibles :

- `url('image.jpg')`
- `none`

Exemple :

```
background-image: url("background.jpg");
```

`background-position`

Positionne l'image de fond.

Valeurs possibles :

- `left`, `center`, `right` pour l'horizontale
- `top`, `center`, `bottom` pour la verticale
- Valeurs en `%` ou en pixels.

Exemple :

```
background-position: top left;
```

`background-repeat`

Définit si l'image doit se répéter ou non.

Valeurs possibles :

- `repeat`, `repeat-x`, `repeat-y`, `no-repeat`

Exemple :

```
background-repeat: no-repeat;
```

`background-attachment`

Définit si l'image de fond doit rester fixe ou se déplacer avec le contenu.

Valeurs possibles :

- `scroll`, `fixed`, `local`

Exemple :

```
background-attachment: fixed;
```

Border

Les propriétés `border` permettent de définir la bordure d'un élément.

`border`

Définit la largeur, le style et la couleur des bordures en une seule ligne.

Exemple :

```
border: 2px solid #000;
```

border-width

Définit l'épaisseur de la bordure.

Valeurs possibles :

- `thin`, `medium`, `thick`
- Valeurs en pixels, `em`, `rem`.

Exemple :

```
border-width: 5px;
```

border-style

Spécifie le style de la bordure.

Valeurs possibles :

- `none`, `solid`, `dashed`, `dotted`, `double`, etc.

Exemple :

```
border-style: dashed;
```

border-color

Définit la couleur de la bordure.

Valeurs possibles :

- Nom de couleur, hexadécimal, `rgba()`.

Exemple :

```
border-color: #ff0000;
```

border-radius

Permet d'arrondir les coins d'un élément.

Valeurs possibles :

- Valeurs en pixels ou pourcentage.

Exemple :

```
border-radius: 10px;
```

Couleur

Les propriétés de couleur définissent les couleurs d'éléments variés comme le texte, les bordures, et l'arrière-plan.

`color`

Définit la couleur du texte.

Valeurs possibles :

- Noms de couleurs (`red`, `blue`)
- Hexadécimal, `rgba()`.

Exemple :

```
color: #333;
```

`opacity`

Définit la transparence d'un élément.

Valeurs possibles :

- Un nombre entre `0` (transparent) et `1` (opaque).

Exemple :

```
opacity: 0.5;
```

Divers

`cursor`

Change l'apparence du curseur lorsque l'utilisateur survole l'élément.

Valeurs possibles :

- `auto`, `pointer`, `text`, `move`, etc.

Exemple :

```
cursor: pointer;
```

`visibility`

Définit si l'élément est visible ou caché.

Valeurs possibles :

- `visible`, `hidden`, `collapse`

Exemple :

```
visibility: hidden;
```

`z-index`

Contrôle l'ordre de superposition d'un élément.

Valeurs possibles :

- Nombre entier (positif ou négatif).

Exemple :

```
z-index: 10;
```

Génération de contenu

`content`

Permet d'ajouter du contenu via CSS, principalement utilisé avec `::before` et `::after`.

Valeurs possibles :

- Texte, URL, compteurs.

Exemple :

```
::before {
  content: "Avant: ";
}
```

`quotes`

Définit le style des guillemets utilisés dans un contenu généré automatiquement.

Valeurs possibles :

- Liste de paires de guillemets.

Exemple :

```
quotes: "«" "»" "‹" "›";
```

Impression

page-break-before

Permet de forcer un saut de page avant l'élément lors de l'impression.

Valeurs possibles :

- `auto`, `always`, `avoid`, `left`, `right`

Exemple :

```
page-break-before: always;
```

orphans

Définit le nombre minimum de lignes d'un paragraphe à laisser au bas d'une page.

Valeurs possibles :

- Nombre entier.

Exemple :

```
orphans: 3;
```

widows

Définit le nombre minimum de lignes d'un paragraphe à laisser en haut d'une nouvelle page.

Exemple :

```
widows: 2;
```

Selecteurs CSS

Les sélecteurs CSS sont des éléments essentiels dans la construction de styles pour une page web. Ils permettent de cibler et de styliser des éléments HTML de manière précise. Voici un guide détaillé des principaux types de sélecteurs CSS, avec des explications pédagogiques et des exemples pratiques pour chaque cas.

Type Selectors

Le sélecteur de type cible tous les éléments d'un type HTML spécifique. C'est le moyen le plus simple de styliser des éléments spécifiques tels que les paragraphes, les titres, les listes, etc.

Exemple :

```css
p {
  font-size: 16px;
}
```

Ce sélecteur cible tous les éléments `p` (paragraphes) de la page et leur applique une taille de police de 16 pixels.

Explication :

- **Avantage** : Simple à utiliser et utile pour appliquer des styles globaux à des éléments courants.
- **Limite** : Il ne cible pas d'éléments spécifiques si plusieurs types similaires sont présents dans le document.

Universal Selector

Le sélecteur universel (`*`) cible tous les éléments de la page, sans distinction. Il est souvent utilisé pour appliquer des styles de réinitialisation à tous les éléments d'une page.

Exemple :

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

Ce sélecteur réinitialise toutes les marges et les remplissages de tous les éléments à 0.

Explication :

- **Avantage** : Utile pour réinitialiser les styles par défaut du navigateur sur tous les éléments.
- **Limite** : Il peut avoir des effets non souhaités s'il est utilisé sans précaution.

Class Selectors

Le sélecteur de classe cible les éléments qui ont une classe spécifique, définie via l'attribut `class`. C'est un moyen flexible et courant pour styliser plusieurs éléments en même temps.

Exemple :

```
.btn-primary {
  background-color: blue;
  color: white;
}
```

Ce sélecteur cible tous les éléments ayant la classe `btn-primary` et applique un fond bleu et un texte blanc.

Explication :

- **Avantage** : Peut être appliqué à plusieurs éléments, permet une grande réutilisation des styles.
- **Limite** : Les noms de classes doivent être bien choisis pour éviter les conflits.

ID Selectors

Le sélecteur d'ID cible un élément avec un identifiant unique, spécifié avec l'attribut `id`. Contrairement aux classes, un ID ne peut être utilisé qu'une seule fois par page.

Exemple :

```
#header {
  background-color: grey;
}
```

Ce sélecteur cible l'élément ayant l'ID `header` et lui applique un arrière-plan gris.

Explication :

- **Avantage** : Permet de cibler un élément unique de manière précise.
- **Limite** : Un ID ne peut être utilisé qu'une seule fois, ce qui limite sa flexibilité.

Attribute Selectors

Les sélecteurs d'attributs permettent de cibler des éléments en fonction de la présence ou des valeurs d'attributs spécifiques. Il existe plusieurs variantes pour affiner cette sélection :

1. **[attribut]** : Cible les éléments qui ont l'attribut spécifié, quelle que soit sa valeur.
2. **[attribut="valeur"]** : Cible les éléments dont l'attribut a exactement la valeur spécifiée.

3. **[attribut~="valeur"]** : Cible les éléments dont l'attribut contient la valeur spécifiée comme un mot individuel.
4. **[attribut|="valeur"]** : Cible les éléments dont l'attribut commence par la valeur spécifiée suivie d'un tiret.
5. **[attribut^="valeur"]** : Cible les éléments dont l'attribut commence par la valeur spécifiée.
6. **[attribut$="valeur"]** : Cible les éléments dont l'attribut se termine par la valeur spécifiée.
7. *[attribut="valeur"]** : Cible les éléments dont l'attribut contient la valeur spécifiée n'importe où dans sa chaîne.

Exemple :

```
input[type="email"] {
  border: 2px solid green;
}
```

Ce sélecteur cible tous les éléments `input` dont l'attribut `type` a la valeur `email`, et leur applique une bordure verte.

Explication :

- **Avantage** : Très précis et puissant, utile pour cibler des éléments en fonction de leurs attributs HTML.
- **Limite** : Peut devenir complexe si de nombreux attributs sont à gérer.

Descendant Selector

Le sélecteur descendant cible tous les éléments qui sont des descendants (directs ou indirects) d'un élément spécifique. Il est écrit en plaçant deux sélecteurs séparés par un espace.

Exemple :

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

Ce sélecteur cible tous les éléments `p` qui sont des descendants de `div` et applique une couleur rouge au texte.

Explication :

- **Avantage** : Très utile pour appliquer des styles à des éléments situés à l'intérieur d'un conteneur spécifique.
- **Limite** : Moins performant si la page contient de nombreux éléments imbriqués.

Child Selector

Le sélecteur enfant cible uniquement les enfants directs d'un élément spécifique. Il est écrit en utilisant le signe `>` entre deux sélecteurs.

Exemple :

```
ul > li {
  list-style-type: none;
}
```

Ce sélecteur cible tous les éléments `li` qui sont des enfants directs d'un élément `ul` et supprime les puces de liste.

Explication :

- **Avantage** : Très précis et permet de cibler uniquement les enfants directs.
- **Limite** : Ne cible pas les descendants plus profonds dans la hiérarchie.

Adjacent Sibling Selector

Le sélecteur de frères adjacents cible l'élément qui suit immédiatement un autre élément dans le DOM. Il est représenté par le signe `+`.

Exemple :

```
h1 + p {
  margin-top: 0;
}
```

Ce sélecteur cible tous les éléments `p` qui suivent directement un élément `h1` et leur applique une marge supérieure de 0.

Explication :

- **Avantage** : Utile pour styliser des éléments qui se suivent immédiatement.
- **Limite** : Ne s'applique qu'au premier élément suivant.

General Sibling Selector

Le sélecteur de frères généraux cible tous les éléments frères (suivant) d'un élément spécifique. Il est représenté par le signe `~`.

Exemple :

```css
h2 ~ p {
  color: green;
}
```

Ce sélecteur cible tous les éléments `p` qui sont des frères d'un élément `h2` et leur applique une couleur verte.

Explication :

- **Avantage** : Permet de cibler tous les frères d'un élément donné, sans restriction sur leur position.
- **Limite** : Cible tous les frères, sans distinction de proximité.

Grouping Selectors

Le sélecteur de groupe permet d'appliquer le même style à plusieurs éléments à la fois. Il est représenté par une liste de sélecteurs séparés par des virgules.

Exemple :

```css
h1,
h2,
h3 {
  font-family: Arial, sans-serif;
}
```

Ce sélecteur applique la même police à tous les éléments `h1`, `h2` et `h3`.

Explication :

- **Avantage** : Réduit la répétition du code en appliquant des styles similaires à plusieurs éléments.
- **Limite** : Tous les sélecteurs doivent être bien réfléchis pour éviter de surcharger la feuille de style.

Combinateurs CSS

Les combinateurs CSS permettent de définir des relations entre les éléments HTML, facilitant ainsi la sélection précise de ceux-ci. Les combinateurs déterminent comment les sélecteurs interagissent entre eux et comment les styles s'appliquent à des éléments en fonction de leur position dans le DOM. Voici un aperçu détaillé des principaux combinateurs avec des exemples pratiques pour chaque cas.

Child Combinator (>)

Le combinateur enfant cible les éléments enfants directs d'un élément parent. Il est représenté par le symbole `>` et ne s'applique qu'aux enfants directs, pas aux descendants plus profonds.

Exemple :

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

Ce sélecteur cible tous les éléments `p` qui sont des enfants directs d'un élément `div` et applique une couleur rouge au texte.

Explication :

- **Avantage** : Utile pour styliser uniquement les enfants directs sans affecter les autres descendants.
- **Limite** : Il ne cible pas les éléments plus profonds dans la hiérarchie, contrairement au combinateur descendant.

Column Combinator (||)

Le combinateur de colonne cible les éléments dans une colonne spécifique d'un élément multicolonne. Il est représenté par le symbole `||` et permet de styliser les éléments dans une structure de colonnes, comme celles créées avec `column-count`.

Exemple :

```
article {
  column-count: 3;
}

article||col-2 {
  background-color: #eee;
}
```

Ce sélecteur cible tous les éléments dans la deuxième colonne d'un élément `article` multicolonne et applique un fond gris clair.

Explication :

- **Avantage** : Utile dans les mises en page multicolonnes pour styliser des éléments par colonne.
- **Limite** : Moins utilisé dans les mises en page modernes où d'autres techniques comme Flexbox ou Grid sont plus courantes.

Descendant Combinator (Espace)

Le combinateur descendant cible les éléments descendants, qu'ils soient des enfants directs ou non. Il est représenté par un espace entre deux sélecteurs.

Exemple :

```
div p {
  font-size: 1.2em;
}
```

Ce sélecteur cible tous les éléments `p` qui sont des descendants de `div` (directs ou indirects) et leur applique une taille de police de 1,2 fois la taille normale.

Explication :

- **Avantage** : Très flexible, car il permet de styliser des éléments à n'importe quel niveau de profondeur.
- **Limite** : Peut être moins performant si la structure du DOM est très complexe avec de nombreux niveaux imbriqués.

Namespace Separator (`|`)

Le séparateur de namespace est utilisé pour cibler les éléments dans un espace de noms spécifique, notamment dans les documents XML ou SVG. Il est représenté par le symbole `|`.

Exemple :

```
svg|circle {
  fill: blue;
}
```

Ce sélecteur cible tous les éléments `circle` dans un document SVG et leur applique une couleur de remplissage bleue.

Explication :

- **Avantage** : Utile pour styliser des éléments dans des espaces de noms spécifiques, comme SVG pour le graphisme vectoriel.
- **Limite** : Nécessite une bonne compréhension des espaces de noms et n'est pas utilisé dans les documents HTML standards.

Next-sibling Combinator (`+`)

Le combinateur de frères adjacents cible un élément qui suit immédiatement un autre élément. Il est représenté par le signe `+`.

Exemple :

```
h1 + p {
  font-weight: bold;
}
```

Ce sélecteur cible le premier élément `p` qui suit immédiatement un élément `h1` et applique un style de texte gras.

Explication :

- **Avantage** : Utile pour styliser les éléments qui suivent immédiatement un autre élément.
- **Limite** : Ne cible que le premier élément suivant, et non tous les éléments frères.

Selector List (Liste de Sélecteurs)

Une liste de sélecteurs permet d'appliquer le même style à plusieurs éléments. Les sélecteurs sont séparés par des virgules, et chaque sélecteur dans la liste reçoit les mêmes styles.

Exemple :

```
h1,
h2,
h3 {
  color: #333;
}
```

Ce sélecteur applique la même couleur de texte gris foncé à tous les éléments `h1`, `h2` et `h3`.

Explication :

- **Avantage** : Réduit la répétition du code en appliquant le même style à plusieurs éléments en une seule ligne.
- **Limite** : Si les styles doivent différer entre les éléments, il faut alors définir des styles séparés pour chacun.

Subsequent-sibling Combinator (~)

Le combinateur de frères suivants cible tous les éléments qui suivent un élément donné, à condition qu'ils aient le même parent. Il est représenté par le signe `~`.

Exemple :

```
h1 ~ p {
  line-height: 1.5;
}
```

Ce sélecteur cible tous les éléments `p` qui suivent un élément `h1` et leur applique une hauteur de ligne de 1,5.

Explication :

- **Avantage** : Permet de styliser tous les frères suivants, pas seulement le premier, ce qui le rend plus flexible que le combinateur `+`.
- **Limite** : Tous les éléments frères après le premier élément sélectionné sont affectés, ce qui peut entraîner des styles non désirés.

Bien sûr ! Voici une version améliorée du texte avec des exemples pour chaque pseudo-classe en CSS :

Les pseudo-classes

Les pseudo-classes en CSS permettent de sélectionner des éléments en fonction de leur état ou de leur position dans le document. Elles sont préfixées par deux-points (`:`) et sont ajoutées après le sélecteur d'élément. Voici un aperçu des différentes pseudo-classes disponibles avec des exemples d'utilisation.

`:active`

`:active` sélectionne un élément au moment où il est activé par l'utilisateur, par exemple, lorsqu'un bouton est cliqué.

```
button:active {
  background-color: red;
}
```

`:any-link`

`:any-link` sélectionne tous les liens, qu'ils aient été visités ou non.

```
a:any-link {
  color: blue;
}
```

`:autofill`

`:autofill` sélectionne un champ de formulaire qui a été rempli automatiquement par le navigateur.

```
input:autofill {
  background-color: lightyellow;
}
```

`:blank`

`:blank` sélectionne un élément qui est vide ou ne contient que des espaces blancs.

```
input:blank {
  border: 2px solid red;
}
```

:buffering

:buffering sélectionne un élément multimédia qui est en train de se charger.

```
video:buffering {
  border: 3px solid yellow;
}
```

:checked

:checked sélectionne un élément de formulaire qui est sélectionné, comme une case à cocher ou un bouton radio.

```
input:checked {
  background-color: green;
}
```

:current

:current sélectionne l'élément qui représente l'élément actuel dans une série, comme un élément de navigation.

```
li:current {
  font-weight: bold;
}
```

:default

:default sélectionne l'élément de formulaire par défaut, comme un bouton radio pré-sélectionné.

```
input:default {
  border: 2px solid blue;
}
```

:defined

:defined sélectionne les éléments qui ont une définition dans le document.

```
element:defined {
  color: purple;
}
```

:dir()

:dir() sélectionne des éléments en fonction de leur directionnalité.

```
:dir(ltr) {
  text-align: left;
}
```

:disabled

:disabled sélectionne les éléments de formulaire qui sont désactivés.

```
input:disabled {
  background-color: grey;
}
```

:empty

:empty sélectionne les éléments qui ne contiennent aucun enfant.

```
div:empty {
  display: none;
}
```

:enabled

:enabled sélectionne les éléments de formulaire qui sont activés.

```
input:enabled {
  border: 1px solid green;
}
```

:first

:first sélectionne le premier élément d'un groupe d'éléments.

```

```

```
p:first {
  font-weight: bold;
}
```

:first-child

:first-child sélectionne le premier enfant d'un élément.

```
ul > li:first-child {
  color: red;
}
```

:first-of-type

:first-of-type sélectionne le premier élément d'un type spécifique dans un élément parent.

```
p:first-of-type {
  margin-top: 0;
}
```

:focus

:focus sélectionne un élément lorsqu'il reçoit le focus, comme lorsqu'un champ de formulaire est sélectionné.

```
input:focus {
  border-color: blue;
}
```

:focus-visible

:focus-visible sélectionne un élément lorsqu'il reçoit le focus via le clavier.

```
input:focus-visible {
  outline: 2px solid orange;
}
```

:focus-within

:focus-within sélectionne un élément parent lorsqu'un de ses enfants reçoit le focus.

```
form:focus-within {
  border: 1px solid purple;
}
```

:fullscreen

`:fullscreen` sélectionne un élément qui est affiché en plein écran.

```
:fullscreen {
  background-color: black;
}
```

:future

`:future` sélectionne un élément qui représente une date ou une heure future.

```
.time:future {
  color: green;
}
```

:has()

`:has()` sélectionne un élément parent en fonction de ses enfants.

```
div:has(p) {
  border: 2px solid yellow;
}
```

:host

`:host` sélectionne l'hôte d'un composant.

```
:host {
  display: block;
}
```

:host-context()

`:host-context()` sélectionne un élément en fonction de son contexte dans le document.

```

```

```
:host-context(.dark-mode) {
  background-color: black;
  color: white;
}
```

:host()

`:host()` sélectionne l'hôte d'un composant en fonction de ses attributs.

```
:host([hidden]) {
  display: none;
}
```

:hover

`:hover` sélectionne un élément lorsque le pointeur de la souris est au-dessus.

```
button:hover {
  background-color: lightblue;
}
```

:in-range

`:in-range` sélectionne un élément de formulaire dont la valeur est dans la plage autorisée.

```
input:in-range {
  border-color: green;
}
```

:indeterminate

`:indeterminate` sélectionne un élément de formulaire dont l'état est indéterminé, comme une case à cocher partiellement sélectionnée.

```
input:indeterminate {
  background-color: lightgrey;
}
```

:invalid

`:invalid` sélectionne un élément de formulaire dont la valeur est invalide.

```
input:invalid {
  border-color: red;
}
```

:is()

`:is()` sélectionne un élément qui correspond à l'un des sélecteurs spécifiés.

```
:is(h1, h2, h3) {
  color: darkblue;
}
```

:lang()

`:lang()` sélectionne des éléments en fonction de leur langue.

```
p:lang(fr) {
  font-style: italic;
}
```

:last-child

`:last-child` sélectionne le dernier enfant d'un élément.

```
ul > li:last-child {
  color: blue;
}
```

:last-of-type

`:last-of-type` sélectionne le dernier élément d'un type spécifique dans un élément parent.

```
p:last-of-type {
  margin-bottom: 0;
}
```

:left

`:left` sélectionne un élément qui est aligné à gauche.

```
div:left {
  float: left;
}
```

:link

:link sélectionne les liens qui n'ont pas encore été visités.

```
a:link {
  color: blue;
}
```

:local-link

:local-link sélectionne les liens qui pointent vers le même domaine que la page actuelle.

```
a:local-link {
  color: green;
}
```

:modal

:modal sélectionne un élément qui est affiché comme une boîte de dialogue modale.

```
div:modal {
  display: block;
}
```

:muted

:muted sélectionne un élément multimédia qui est en sourdine.

```
video:muted {
  border: 3px solid red;
}
```

:not()

:not() sélectionne les éléments qui ne correspondent pas à un sélecteur spécifique.

```

```

```
p:not(.special) {
  color: grey;
}
```

:nth-child()

:nth-child() sélectionne un enfant en fonction de sa position dans la liste des enfants.

```
li:nth-child(2) {
  color: red;
}
```

:nth-last-child()

:nth-last-child() sélectionne un enfant en fonction de sa position en partant de la fin de la liste des enfants.

```
li:nth-last-child(2) {
  color: green;
}
```

:nth-last-of-type()

:nth-last-of-type() sélectionne un élément en fonction de sa position en partant de la fin de la liste des éléments d'un type spécifique.

```
p:nth-last-of-type(1) {
  color: blue;
}
```

:nth-of-type()

:nth-of-type() sélectionne un élément en fonction de sa position dans la liste des éléments d'un type spécifique.

```
p:nth-of-type(3) {
  color: purple;
}
```

:only-child

:only-child sélectionne un élément qui est le seul enfant de son parent.

```
div:only-child {
  border: 1px solid black;
}
```

:only-of-type

:only-of-type sélectionne un élément qui est le seul élément de son type dans son parent.

```
div:only-of-type {
  background-color: yellow;
}
```

:optional

:optional sélectionne un élément de formulaire qui n'est pas requis.

```
input:optional {
  border: 1px dashed grey;
}
```

:out-of-range

:out-of-range sélectionne un élément de formulaire dont la valeur est en dehors de la plage autorisée.

```
input:out-of-range {
  border-color: red;
}
```

:past

:past sélectionne un élément qui représente une date ou une heure passée.

```
.time:past {
  color: grey;
}
```

:paused

:paused sélectionne un élément multimédia qui est en pause.

```

```

```
video:paused {
  border: 2px solid orange;
}
```

:picture-in-picture

:picture-in-picture sélectionne un élément vidéo qui est affiché dans une fenêtre picture-in-picture.

```
video:picture-in-picture {
  border: 2px solid green;
}
```

:placeholder-shown

:placeholder-shown sélectionne un élément de formulaire qui affiche un texte d'espace réservé.

```
input:placeholder-shown {
  background-color: lightgrey;
}
```

:playing

:playing sélectionne un élément multimédia qui est en cours de lecture.

```
video:playing {
  border: 3px solid blue;
}
```

:popover-open

:popover-open sélectionne un élément qui affiche une info-bulle.

```
div:popover-open {
  display: block;
}
```

:read-only

:read-only sélectionne un élément de formulaire qui est en lecture seule.

```

```

```
input:read-only {
  background-color: lightgrey;
}
```

:read-write

:read-write sélectionne un élément de formulaire qui est modifiable.

```
input:read-write {
  background-color: white;
}
```

:required

:required sélectionne un élément de formulaire qui est requis.

```
input:required {
  border: 2px solid red;
}
```

:right

:right sélectionne un élément qui est aligné à droite.

```
div:right {
  float: right;
}
```

:root

:root sélectionne l'élément racine du document.

```
:root {
  --main-color: coral;
}
```

:scope

:scope sélectionne l'élément de référence d'un sélecteur.

```

```

```
:scope > p {
  color: blue;
}
```

:seeking

`:seeking` sélectionne un élément multimédia qui est en cours de recherche.

```
video:seeking {
  border: 2px solid red;
}
```

:stalled

`:stalled` sélectionne un élément multimédia qui est en pause à cause d'une erreur de chargement.

```
video:stalled {
  border: 3px solid grey;
}
```

:state()

`:state()` sélectionne un élément en fonction de son état.

```
element:state(active) {
  background-color: yellow;
}
```

:target

`:target` sélectionne un élément qui est la cible d'un lien.

```
:target {
  border: 2px solid green;
}
```

:target-within

`:target-within` sélectionne un élément parent qui contient la cible d'un lien.

```

```

```
:target-within {
  border: 3px solid blue;
}
```

:user-invalid

:user-invalid sélectionne un élément de formulaire dont la valeur est invalide en raison d'une interaction utilisateur.

```
input:user-invalid {
  border-color: red;
}
```

:user-valid

:user-valid sélectionne un élément de formulaire dont la valeur est valide en raison d'une interaction utilisateur.

```
input:user-valid {
  border-color: green;
}
```

:valid

:valid sélectionne un élément de formulaire dont la valeur est valide.

```
input:valid {
  border-color: green;
}
```

:visited

:visited sélectionne les liens qui ont été visités.

```
a:visited {
  color: purple;
}
```

:volume-locked

:volume-locked sélectionne un élément multimédia dont le volume est verrouillé.

```
video:volume-locked {
  border: 2px solid black;
}
```

`:where()`

`:where()` sélectionne les éléments qui correspondent à un sélecteur spécifique.

```
:where(.important) {
  font-weight: bold;
}
```

Pseudo-éléments

Les pseudo-éléments en CSS sont utilisés pour styliser des parties spécifiques d'un élément qui ne peuvent pas être sélectionnées directement avec des sélecteurs normaux. Ils commencent tous par deux doubles deux-points `::`. Voici une liste de pseudo-éléments CSS que tu devrais connaître :

`::after`

Le pseudo-élément `::after` est utilisé pour insérer du contenu après le contenu d'un élément. Par exemple :

```
p::after {
  content: " - Fin de paragraphe";
}
```

Cela ajoutera le texte " - Fin de paragraphe" après chaque paragraphe.

`::backdrop`

Le pseudo-élément `::backdrop` est utilisé pour styliser la zone derrière une boîte de dialogue modale. Par exemple :

```
dialog::backdrop {
  background-color: rgba(0, 0, 0, 0.5);
}
```

Cela appliquera une couleur de fond semi-transparente noire à la zone derrière une boîte de dialogue modale.

`::before`

Le pseudo-élément `::before` est utilisé pour insérer du contenu avant le contenu d'un élément. Par exemple :

```
li::before {
  content: "- ";
}
```

Cela ajoutera un tiret devant chaque élément de liste.

`::cue`

Le pseudo-élément `::cue` est utilisé pour styliser les légendes d'une piste audio ou vidéo. Par exemple :

```
video::cue {
  color: white;
  background-color: black;
}
```

Cela appliquera une couleur de texte blanche et une couleur de fond noire aux légendes d'une vidéo.

`::cue-region`

Le pseudo-élément `::cue-region` est utilisé pour styliser une région spécifique d'une piste audio ou vidéo. Par exemple :

```
video::cue-region(title) {
  color: yellow;
  background-color: black;
}
```

Cela appliquera une couleur de texte jaune et une couleur de fond noire aux titres d'une vidéo.

`::file-selector-button`

Le pseudo-élément `::file-selector-button` est utilisé pour styliser le bouton d'un sélecteur de fichiers. Par exemple :

```
input[type="file"]::file-selector-button {
  border: 2px solid #007bff;
  color: #007bff;
  background-color: white;
}
```

Cela appliquera une bordure bleue, une couleur de texte bleue et une couleur de fond blanche au bouton d'un sélecteur de fichiers.

`::first-letter`

Le pseudo-élément `::first-letter` est utilisé pour styliser la première lettre d'un élément. Par exemple :

```
p::first-letter {
  font-size: 2em;
  color: red;
}
```

Cela appliquera une taille de police double et une couleur de texte rouge à la première lettre de chaque paragraphe.

`::first-line`

Le pseudo-élément `::first-line` est utilisé pour styliser la première ligne d'un élément. Par exemple :

```
p::first-line {
  font-weight: bold;
  color: blue;
}
```

Cela appliquera une graisse de police en gras et une couleur de texte bleue à la première ligne de chaque paragraphe.

`::grammar-error`

Le pseudo-élément `::grammar-error` est utilisé pour styliser les erreurs de grammaire dans un texte. Par exemple :

```
p::grammar-error {
  text-decoration: underline wavy red;
}
```

Cela appliquera une ligne ondulée rouge en dessous des erreurs de grammaire dans un paragraphe.

`::highlight()`

Le pseudo-élément `::highlight()` est utilisé pour styliser le texte sélectionné par l'utilisateur. Par exemple :

```
::selection {
  background-color: yellow;
  color: black;
}
```

Cela appliquera une couleur de fond jaune et une couleur de texte noire au texte sélectionné par l'utilisateur.

::marker

Le pseudo-élément `::marker` est utilisé pour styliser les puces d'une liste. Par exemple :

```
li::marker {
  color: red;
  font-size: 1.5em;
}
```

Cela appliquera une couleur de texte rouge et une taille de police 1,5 fois plus grande aux puces d'une liste.

::part()

Le pseudo-élément `::part()` est utilisé pour styliser une partie spécifique d'un composant Web. Par exemple :

```
my-component::part(header) {
  background-color: lightgray;
  padding: 10px;
}
```

Cela appliquera une couleur de fond gris clair et un rembourrage de 10 pixels à l'en-tête d'un composant Web personnalisé.

::placeholder

Le pseudo-élément `::placeholder` est utilisé pour styliser le texte d'espace réservé d'un champ de formulaire. Par exemple :

```
input::placeholder {
  color: gray;
  font-style: italic;
}
```

Cela appliquera une couleur de texte grise et une police en italique au texte d'espace réservé d'un champ de formulaire.

`::selection`

Le pseudo-élément `::selection` est utilisé pour styliser le texte sélectionné par l'utilisateur. Par exemple :

```
::selection {
  background-color: yellow;
  color: black;
}
```

Cela appliquera une couleur de fond jaune et une couleur de texte noire au texte sélectionné par l'utilisateur.

`::slotted()`

Le pseudo-élément `::slotted()` est utilisé pour styliser le contenu d'un élément inséré dans un slot d'un composant Web. Par exemple :

```
my-component::slotted(p) {
  color: blue;
}
```

Cela appliquera une couleur de texte bleue aux paragraphes insérés dans un slot d'un composant Web personnalisé.

`::spelling-error`

Le pseudo-élément `::spelling-error` est utilisé pour styliser les erreurs d'orthographe dans un texte. Par exemple :

```
p::spelling-error {
  text-decoration: underline wavy red;
}
```

Cela appliquera une ligne ondulée rouge en dessous des erreurs d'orthographe dans un paragraphe.

`::target-text`

Le pseudo-élément `::target-text` est utilisé pour styliser le texte d'un élément cible. Par exemple :

```
```

```
a::target-text {
  color: red;
}
```

Cela appliquera une couleur de texte rouge au texte d'un élément cible d'un lien.

Voici des exemples pour les pseudo-éléments manquants :

`::view-transition`

Le pseudo-élément `::view-transition` est utilisé pour styliser la transition entre deux vues. C'est une fonctionnalité expérimentale et n'est pas encore prise en charge par tous les navigateurs.

```
::view-transition {
  transition: opacity 0.5s ease-in-out;
}
```

`::view-transition-group`

Le pseudo-élément `::view-transition-group` est utilisé pour styliser un groupe de transitions de vue. C'est une fonctionnalité expérimentale et n'est pas encore prise en charge par tous les navigateurs.

```
::view-transition-group {
  display: flex;
  flex-direction: column;
}
```

`::view-transition-image-pair`

Le pseudo-élément `::view-transition-image-pair` est utilisé pour styliser une paire d'images pendant une transition de vue. C'est une fonctionnalité expérimentale et n'est pas encore prise en charge par tous les navigateurs.

```
::view-transition-image-pair {
  border: 1px solid black;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
}
```

`::view-transition-new`

Le pseudo-élément `::view-transition-new` est utilisé pour styliser la nouvelle vue pendant une transition de vue. C'est une fonctionnalité expérimentale et n'est pas encore prise en charge par tous les navigateurs.

```
::view-transition-new {
  opacity: 0;
  animation: fade-in 0.5s forwards;
}
```

```
@keyframes fade-in {
  to {
    opacity: 1;
  }
}
```

`::view-transition-old`

Le pseudo-élément `::view-transition-old` est utilisé pour styliser l'ancienne vue pendant une transition de vue. C'est une fonctionnalité expérimentale et n'est pas encore prise en charge par tous les navigateurs.

```
::view-transition-old {
  opacity: 1;
  animation: fade-out 0.5s forwards;
}
```

```
@keyframes fade-out {
  to {
    opacity: 0;
  }
}
```

At-rules

Les At-rules en CSS sont des instructions spéciales utilisées pour définir des règles ou des paramètres qui s'appliquent à l'ensemble du document ou à une partie spécifique du document.

`@charset`

`@charset` est utilisée pour définir l'encodage de caractères du document CSS. Elle doit être placée en premier dans le fichier CSS.

```
@charset "UTF-8";
```

`@color-profile`

`@color-profile` est utilisée pour définir le profil de couleur du document CSS. Elle n'est pas prise en charge par tous les navigateurs.

```
@color-profile url("color-profile.icc");
```

@container

`@container` est utilisée pour définir un conteneur pour les styles de mise en page. Elle est utilisée avec la propriété `container`.

```
@container (min-width: 500px) {
  .element {
    width: 50%;
  }
}
```

@counter-style

`@counter-style` est utilisée pour définir un style de compteur personnalisé. Elle est utilisée avec la propriété `list-style-type`.

```
@counter-style custom {
  system: cyclic;
  symbols: "▪" "○" "▫";
}

li {
  list-style-type: custom;
}
```

@font-face

`@font-face` est utilisée pour définir une police personnalisée et l'utiliser dans le document CSS.

```
@font-face {
  font-family: "Ma police";
  src: url("ma-police.woff2") format("woff2"), url("ma-police.woff")
format("woff");
  font-weight: normal;
  font-style: normal;
}
```

```
body {
  font-family: "Ma police", sans-serif;
}
```

@font-feature-values

`@font-feature-values` est utilisée pour définir des valeurs personnalisées pour les fonctionnalités de police.

```
@font-feature-values "ss01" {
  "on" {
    -webkit-font-feature-settings: "ss01" 1;
    font-feature-settings: "ss01" 1;
  }
  "off" {
    -webkit-font-feature-settings: "ss01" 0;
    font-feature-settings: "ss01" 0;
  }
}
```

@font-palette-values

`@font-palette-values` est utilisée pour définir des palettes de couleurs personnalisées pour les polices.

```
@font-palette-values --my-palette {
  base-palette: dark;
  override-colors: #ff0000 #00ff00 #0000ff;
}

body {
  font-palette: --my-palette;
}
```

@import

`@import` est utilisée pour importer une feuille de style externe dans le document CSS.

```
@import url("style.css");
```

@keyframes

`@keyframes` est utilisée pour définir une animation CSS.

```
@keyframes mon-animation {
  0% {
    transform: scale(1);
  }
  50% {
    transform: scale(2);
  }
  100% {
    transform: scale(1);
  }
}

.element {
  animation: mon-animation 2s infinite;
}
```

@layer

`@layer` est utilisée pour définir des couches de styles CSS. Elle est utilisée avec la propriété `@layer`.

```
@layer base {
  body {
    font-size: 16px;
  }
}

@layer theme {
  body {
    color: #333;
  }
}
```

@media

`@media` est utilisée pour appliquer des styles en fonction de la taille de l'écran ou d'autres caractéristiques du dispositif.

```
@media screen and (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

@namespace

`@namespace` est utilisée pour définir un espace de noms pour les sélecteurs et les propriétés.

```
@namespace url(http://www.example.com/);

|* {
  color: red;
}
```

@page

`@page` est utilisée pour définir des styles pour l'impression.

```
@page {
  size: A4;
  margin: 2cm;
}
```

@property

`@property` est utilisée pour définir des propriétés CSS personnalisées.

```
@property --ma-propriete {
  syntax: "<length>";
  initial-value: 0;
  inherits: false;
}

.element {
  --ma-propriete: 10px;
}
```

@scope

`@scope` est utilisée pour définir une portée pour les styles. Elle n'est pas prise en charge par tous les navigateurs.

```
@scope (selector) {
  /* styles */
}
```

`@starting-style`

`@starting-style` est utilisée pour définir un style par défaut pour un élément. Elle n'est pas prise en charge par tous les navigateurs.

```
@starting-style {
  h1 {
    color: red;
  }
}
```

`@supports`

`@supports` est utilisée pour vérifier si une propriété CSS est prise en charge par le navigateur.

```
@supports (display: grid) {
  .element {
    display: grid;
  }
}
```

Fonctions mathématiques

`abs()`

La fonction `abs()` renvoie la valeur absolue d'un nombre.

```
width: abs(-10px); /* renvoie 10px */
```

`acos()`

La fonction `acos()` renvoie l'arc cosinus d'un nombre en radians.

```
transform: rotate(acos(0.5)); /* renvoie 1.0472 rad */
```

`asin()`

La fonction `asin()` renvoie l'arc sinus d'un nombre en radians.

```
transform: rotate(asin(0.5)); /* renvoie 0.5236 rad */
```

`atan()`

La fonction `atan()` renvoie l'arc tangente d'un nombre en radians.

```
transform: rotate(atan(1)); /* renvoie 0.7854 rad */
```

`atan2()`

La fonction `atan2()` renvoie l'angle en radians entre l'axe des x positifs et le point donné par les coordonnées (y, x).

```
transform: rotate(atan2(1, 1)); /* renvoie 0.7854 rad */
```

`cos()`

La fonction `cos()` renvoie le cosinus d'un angle.

```
transform: rotate(cos(0.7854rad)); /* renvoie 0.7071 */
```

`sin()`

La fonction `sin()` renvoie le sinus d'un angle.

```
transform: rotate(sin(0.7854rad)); /* renvoie 0.7071 */
```

`tan()`

La fonction `tan()` renvoie la tangente d'un angle.

```
transform: rotate(tan(0.7854rad)); /* renvoie 1 */
```

Fonctions de manipulation de chaînes

`attr()`

La fonction `attr()` récupère la valeur d'un attribut HTML.

```
content: attr(title); /* affiche la valeur de l'attribut title */
```

`counter()`

La fonction `counter()` permet de manipuler des compteurs CSS.

```
content: counter(section) "." counter(subsection); /* affiche un compteur
personnalisé */
```

`counters()`

La fonction `counters()` combine plusieurs compteurs.

```
content: counters(chapter, ".") " "; /* affiche un compteur avec un séparateur */
```

Fonctions de calcul

`calc()`

La fonction `calc()` effectue des calculs dans les propriétés CSS.

```
width: calc(100% - 50px); /* soustrait 50px de 100% */
```

`clamp()`

La fonction `clamp()` définit une valeur avec des limites.

```
font-size: clamp(
  1rem,
  2vw,
  1.5rem
); /* définit une taille avec des valeurs minimales et maximales */
```

`max()`

La fonction `max()` utilise la plus grande valeur entre plusieurs arguments.

```
width: max(300px, 50%); /* utilise la plus grande valeur */
```

`min()`

La fonction `min()` utilise la plus petite valeur entre plusieurs arguments.

```
width: min(300px, 50%); /* utilise la plus petite valeur */
```

`minmax()`

La fonction `minmax()` définit une valeur minimale et maximale dans un contexte de grille.

```
grid-template-columns: repeat(
  auto-fit,
  minmax(200px, 1fr)
); /* définit des colonnes avec des tailles minimales et maximales */
```

Autres fonctions

`env()`

La fonction `env()` récupère la valeur d'une variable de l'environnement utilisateur.

```
padding-bottom: env(
  safe-area-inset-bottom
); /* utilise la valeur de la zone sécurisée */
```

`exp()`

La fonction `exp()` renvoie la valeur de `e` à la puissance d'un nombre donné.

```
transform: scale(exp(1)); /* renvoie 2.7183 */
```

`fit-content()`

La fonction `fit-content()` définit la taille d'un élément en fonction de son contenu.

```
width: fit-content(
  300px
); /* ajuste la largeur à son contenu avec une limite de 300px */
```

`hypot()`

La fonction `hypot()` renvoie la longueur de l'hypoténuse d'un triangle droit.

```
transform: scale(hypot(3, 4)); /* renvoie 5 */
```

`log()`

La fonction `log()` renvoie le logarithme naturel d'un nombre.

```
transform: scale(log(1)); /* renvoie 0 */
```

`mod()`

La fonction `mod()` renvoie le reste d'une division.

```
transform: rotate(mod(5, 2) * 45deg); /* renvoie 45 degrés */
```

`path()`

La fonction `path()` définit un chemin SVG pour être utilisé dans les animations CSS.

```
motion-path: path("M10 10 H 90 V 90 H 10 Z"); /* définit un chemin SVG */
```

`pow()`

La fonction `pow()` renvoie un nombre élevé à la puissance d'un autre.

```
transform: scale(pow(2, 3)); /* renvoie 8 */
```

ray()

La fonction `ray()` crée une direction sous forme d'angle.

```
background-image: conic-gradient(
  at ray(45deg)
); /* crée un dégradé directionnel */
```

rem()

La fonction `rem()` renvoie une valeur relative à la taille de la police racine du document.

```
font-size: 2rem; /* définit une taille de police égale à 2 fois la taille racine
*/
```

repeat()

La fonction `repeat()` répète une image ou un modèle défini.

```
background: repeat(url("image.jpg")); /* répète une image en arrière-plan */
```

round()

La fonction `round()` arrondit un nombre à l'entier le plus proche.

```
width: round(33.7px); /* renvoie 34px */
```

sign()

La fonction `sign()` renvoie le signe d'un nombre.

```
transform: scale(sign(-3)); /* renvoie -1 */
```

sqrt()

La fonction `sqrt()` renvoie la racine carrée d'un nombre.

```
transform: scale(sqrt(9)); /* renvoie 3 */
```

`url()`

La fonction `url()` inclut une ressource externe dans une propriété CSS.

```
background-image: url("image.jpg"); /* inclut une image en arrière-plan */
```

`var()`

La fonction `var()` récupère la valeur d'une variable CSS personnalisée.

```
color: var(--main-color); /* utilise une variable CSS */
```

Les types

Ce cours couvre les différents types en CSS, en expliquant chaque type et en fournissant des exemples concrets pour faciliter la compréhension. Chaque type est présenté avec ses valeurs possibles et son utilité dans la conception des pages web.

`<absolute-size>`

Les valeurs de type `<absolute-size>` définissent la taille de la police de manière absolue. Ce sont des tailles fixes, indépendantes de la taille du conteneur parent.

Valeurs :

- `xx-small`
- `x-small`
- `small`
- `medium` (valeur par défaut)
- `large`
- `x-large`
- `xx-large`

Exemple :

```
p {
  font-size: large;
}
```

`<alpha-value>`

Les valeurs `<alpha-value>` déterminent la transparence d'une couleur, allant de 0 (complètement transparent) à 1 (complètement opaque).

Valeurs possibles :

- Entre 0 et 1.

Exemple :

```
div {
  background-color: rgba(255, 0, 0, 0.5); /* Rouge à 50% d'opacité */
}
```

`<angle>`

Les valeurs `<angle>` définissent les angles. Elles sont utilisées dans des transformations (comme `rotate`), des dégradés, etc.

Unités disponibles :

- `deg` (degrés)
- `rad` (radians)
- `grad` (gradians)
- `turn` (tours)

Exemple :

```
div {
  transform: rotate(45deg);
}
```

`<basic-shape>`

Ce type est utilisé pour découper une forme basique dans un élément à l'aide de la propriété `clip-path`.

Formes possibles :

- `inset()`
- `circle()`
- `ellipse()`

- `polygon()`

Exemple :

```
div {
  clip-path: circle(50%);
}
```

`<blend-mode>`

Les valeurs `<blend-mode>` définissent la manière dont les couleurs de deux éléments se mélangent.

Valeurs disponibles :

- `normal`
- `multiply`
- `screen`
- `overlay`
- `darken`
- `lighten`
- `color-dodge`, etc.

Exemple :

```
div {
  background-color: rgba(255, 0, 0, 0.5);
  mix-blend-mode: screen;
}
```

`<box-edge>`

Les valeurs `<box-edge>` définissent les bords à partir desquels les dimensions sont calculées.

Valeurs :

- `padding-box`
- `border-box`
- `content-box`

Exemple :

```
div {
  box-sizing: border-box;
```

```
}
```

`<calc-sum>`

Les valeurs `<calc-sum>` permettent de réaliser des calculs simples dans des propriétés CSS comme `width`, `height`, etc.

Exemple :

```
div {
  width: calc(100% - 50px);
}
```

`<color>`

Les valeurs `<color>` représentent les couleurs en CSS. Plusieurs formats sont possibles.

Formats disponibles :

- Nom de couleur (`red`, `blue`, `green`)
- Hexadécimal (`#ff0000`)
- RGB (`rgb(255, 0, 0)`)
- HSL (`hsl(0, 100%, 50%)`)

Exemple :

```
div {
  background-color: #00ff00;
}
```

`<custom-ident>`

Ces valeurs sont utilisées pour définir des identifiants personnalisés, notamment avec les variables CSS.

Exemple :

```
:root {
  --main-color: #3498db;
}

div {
```

```
  background-color: var(--main-color);
}
```

<dimension>

Les dimensions en CSS sont exprimées en unités de longueur telles que `px`, `em`, `rem`, `%`, etc.

Exemple :

```
div {
  width: 100px;
  height: 50%;
}
```

<display-box>

Définit la manière dont l'élément est affiché.

Valeurs disponibles :

- `block`
- `inline`
- `flex`
- `grid`
- `inline-block`, etc.

Exemple :

```
div {
  display: flex;
}
```

<easing-function>

Ces fonctions définissent la manière dont une animation évolue au cours du temps.

Valeurs disponibles :

- `ease`
- `linear`
- `ease-in`
- `ease-out`

- `ease-in-out`
- `cubic-bezier()`

Exemple :

```
div {
  transition: all 0.5s ease-in-out;
}
```

`<filter-function>`

Les fonctions de filtre sont utilisées pour appliquer des effets visuels sur un élément, comme le flou ou la saturation.

Valeurs disponibles :

- `blur()`
- `brightness()`
- `contrast()`
- `grayscale()`
- `hue-rotate()`, etc.

Exemple :

```
div {
  filter: blur(5px);
}
```

`<flex>`

Les valeurs de type `<flex>` sont utilisées dans les éléments `flex` pour définir leur flexibilité.

Exemple :

```
div {
  display: flex;
  flex: 1 1 auto;
}
```

`<gradient>`

Les dégradés permettent de créer des transitions progressives entre plusieurs couleurs.

Types disponibles :

- `linear-gradient()`
- `radial-gradient()`
- `conic-gradient()`

Exemple :

```
div {
  background: linear-gradient(to right, red, blue);
}
```

`<length>`

Les unités de longueur définissent la taille des éléments, des marges, etc.

Valeurs possibles :

- `px`
- `em`
- `rem`
- `vw` / `vh` (pourcentages de la largeur ou hauteur de la fenêtre)

Exemple :

```
div {
  margin: 20px;
}
```

`<position>`

Les valeurs `<position>` définissent la manière dont un élément est positionné dans le flux du document.

Valeurs disponibles :

- `static`
- `relative`
- `absolute`
- `fixed`
- `sticky`

Exemple :

```css
div {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

`<string>`

Les chaînes de caractères en CSS sont utilisées notamment dans les pseudo-éléments ou pour des attributs de contenu.

Exemple :

```css
div::before {
  content: "Hello, World!";
}
```

`<time>`

Les durées permettent de spécifier la durée des animations ou des transitions.

Unités disponibles :

- `s` (secondes)
- `ms` (millisecondes)

Exemple :

```css
div {
  transition: all 2s;
}
```