Introduction CSS.mp4

dclic.francophonie.org/pluginfile.php/6347/mod_resource/content/2/Introduction aux feuilles de style CSS.html#h5pbookid=2058063678&section=top&chapter=h5p-interactive-book-chapter-be8231ef-9a5b-4b79-b444-8de3a684b7ee

Introduction

À quoi sert CSS ?

Le HTML, bien qu'il soit la base fondamentale d'une page web, ne suffit pas à lui seul pour offrir aux développeurs web de réaliser un contenu visuel attrayant et ergonomique. Le HTML se concentre sur la structure et la sémantique du contenu d'une page. Il définit les éléments tels que les titres, les paragraphes, les images et les liens, mais il n'inclut pas les détails de l'apparence visuelle de ces éléments.

L'association du HTML et du CSS crée une synergie puissante qui transforme une simple collection de balises en une présentation visuelle spectaculaire, raffinée et conviviale. Ainsi, cette association du HTML et du CSS garantit non seulement que le contenu soit présenté de manière structurée, mais qu'il soit également :

- **Expressif** : Le CSS permet d'exprimer la personnalité et le style de la marque à travers la présentation visuelle.
- **Innovant** : Le CSS offre la possibilité d'expérimenter avec des designs innovants et créatifs.
- **Cohérent** : Le CSS permet de maintenir une cohérence visuelle à travers toutes les pages du site en appliquant des styles réutilisables.
- **Équilibré** : Le CSS peut aider à maintenir un équilibre visuel entre les éléments de la page.
- **Professionnel** : L'application appropriée du CSS donne un aspect professionnel et soigné à la présentation, renforçant la crédibilité du site.
- **Adaptatif** : Grâce aux techniques de mise en page réactive, le CSS rend la présentation adaptative à différents dispositifs et tailles d'écran.

Le CSS : langage de style

Le CSS (Cascading Style Sheets), en français "Feuilles de style en cascade", joue un rôle essentiel pour donner vie aux pages web écrites en HTML et offrir aux utilisateurs une apparence bien soignée,

professionnelle, mémorable et convaincante en appliquant des règles de style aux éléments HTML. En utilisant des sélecteurs et des propriétés, le CSS permet aux développeurs de spécifier des règles détaillées pour mettre en forme les différents éléments contenus dans leurs pages web.

CSS est concurrent à HTML ?

- OUI
- NON

1 / 9

Introduction

2 / 9

Syntaxe

Syntaxe du langage CSS

La syntaxe du CSS est la grammaire qui régit comment vous exprimez vos choix de mise en forme. Pas comme les langages de programmation, CSS n'a pas la capacité de prendre des décisions conditionnelles ou d'exécuter des instructions logiques. Il fournit, plutôt, des **règles** pour styliser et mettre en forme les éléments HTML.

La structure d'une règle CSS doit respecter la forme suivante :

Le Sélecteur : permet de sélectionner les éléments sur lesquels vous aller appliquer le style souhaité. Il peut être basé sur le nom de la balise, les classes d'éléments, les identiants d'éléments HTML (id), les

attributs et même des sélecteurs pseudo-classes. Les sélecteurs CSS peuvent être classés en cinq catégories :

1. **Sélecteurs Simples :** Ils sont utilisés pour sélectionner des éléments en fonction de leur nom, de leur identifiant ou de leur classe.
2. **Sélecteurs Combinateurs :** Ils sont utilisés pour sélectionner des éléments en fonction de leur relation avec d'autres éléments.
3. **Sélecteurs de Pseudo-Classe :** Ils sont utilisés pour sélectionner des éléments en fonction de leur état ou de leur interaction avec l'utilisateur.
4. **Sélecteurs de Pseudo-Éléments :** Ils permettent de sélectionner et de styliser des parties spécifiques d'un élément, comme le contenu avant ou après l'élément.
5. **Sélecteurs d'Attributs :** Ils permettent de sélectionner des éléments en fonction de la présence d'un attribut particulier ou de sa valeur.

La Déclaration : Les déclarations spécifient les styles que vous souhaitez appliquer aux éléments sélectionnés. Une déclaration CSS consiste en une propriété et sa valeur associée. Chaque déclaration est terminée par un point-virgule pour indiquer sa fin.

Le système de spécification CSS permet d'ajuster plusieurs propriétés de style en même temps, grâce à la possibilité d'inclure plusieurs déclarations au sein d'une seule règle.

La Propriété : Une propriété est une caractéristique visuelle spécifique que vous voulez la personnaliséé, telles que la couleur du texte, la taille, la police, la marge, le rembourrage, etc.

La Valeur de la Propriété : Située à droite de la propriété, après les deux points. Cette valeur détermine le style spécifique que vous voulez appliquer à la propriété en question.

De plus la syntaxe d'une règle CSS doit respecter ces exigences :

- l'ensemble des déclarations d'une même règle est placé entre des accolades ({}).
- séparer la propriété de sa valeur par les deux points (:) .
- séparer les différentes déclarations par un point-virgule (;) .

Dans les sections à venir de ce cours, nous explorerons en détail les éléments fondamentaux qui composent une règle CSS.

Quel élément d'une règle CSS est utilisé pour sélectionner les éléments HTML que vous souhaitez styliser ?

- Valeur de la propriété
- Sélecteur

-
-
  Propriété
-
-
  Déclaration
-

2 / 9

Syntaxe

3 / 9

Appliquer un style CSS

Où écrit-on le CSS ?

Il existe plusieurs manières pour intégrer le code CSS dans un code HTML. Chacune offre des avantages spécifiques en fonction de la complexité de votre projet et de votre préférence en matière de maintenance. L'utilisation judicieuse de ces méthodes vous permettra de maintenir un code organisé, cohérent et facile à gérer, tout en garantissant des styles visuellement attrayants et professionnels pour vos pages web. Voici un aperçu de ces différentes méthodes :

Feuille de style externe :

Lorsque vous voulez que plusieurs pages d'un même site web arborent une allure similaire, il est avisé d'adopter un ensemble de règles commun pour avoir une cohérence visuelle entre les différentes page constituant le site. Dans cette perspective, il est pratique d'écrire ces règles une seule fois dans une feuille de style partagée par l'ensemble des pages.

Créer un fichier CSS séparé avec une extension `.css` vous permet de centraliser tous vos styles dans un seul fichier. En utilisant la balise `<link>` dans la section `<head>` de votre HTML, vous reliez cette feuille de style externe à votre document.

Cette méthode est recommandée pour les projets de taille moyenne à grande, car elle maintient une séparation claire entre le code HTML et les styles. De plus, vous pouvez réutiliser les mêmes styles sur plusieurs pages. Cela signifie que vous n'avez pas besoin de dupliquer les règles de style pour chaque page, ce qui allège le code et facilite la mise à jour globale au lieu de modifier chaque page individuellement.

```
<!doctype html>

<html>

  <head>

    <meta charset="utf-8" />

    <title>Une expérience avec CSS</title>

    <!-- la feuille de style se trouve dans le répertoire courant --
>

    <link rel="stylesheet" href="styles.css" />

  </head>

  <body>

    <h1> Bonjour !</h1>

    <p> Ceci est un premier exemple CSS</p>

  </body>

</html>
```

Le fichier CSS devrait ressembler à cela :

```
h1 {

  color: blue;

  background-color: yellow;

  border: 1px solid black;

}


p {

  color: red;

}
```

Le lien hypertexte href de l'élément <link> doit diriger vers un fichier contenu dans votre système de fichiers.

Dans l'exemple précédent, le fichier CSS et le document HTML résident dans le même répertoire, toutefois, il est possible de les localiser ailleurs et d'ajuster le chemin d'accès, comme illustré ici :

```
<!-- Dans un sous-répertoire nommé styles dans le répertoire courant

-->

<link rel="stylesheet" href="styles/style.css" />


<!-- Dans un sous-répertoire nommé general, lui-même dans un sous-

répertoire nommé styles, dans le répertoire courant -->
```

```
<link rel="stylesheet" href="styles/general/style.css" />
```

```
<!-- Dans un sous-répertoire nommé styles, un niveau plus haut -->
```

```
<link rel="stylesheet" href="../styles/style.css" />
```

Balise `<style>` dans l'en-tête HTML :

Les règles CSS ont la possibilité d'être rédigées directement au sein de la section `<head>` du document HTML, en utilisant l'élément `<style>` . Cette méthode permet de définir des styles qui s'appliquent à une seule page. Elle est idéale pour les projets de petite envergure où les styles sont limités à une seule page.

L'exemple HTML ci-dessous met en lumière cette méthode :

```
<!doctype html>
```

```
<html>
```

```
  <head>
```

```
    <meta charset="utf-8" />
```

```
    <title>Mes expérimentations CSS</title>
```

```
    <style>
```

```
      h1 {
```

```
        color: blue;
```

```
        background-color: yellow;
```

```
        border: 1px solid black;
    }

    p {
        color: red;
    }
  </style>
</head>
<body>
  <h1>Hello World!</h1>
  <p>Ceci est mon premier exemple CSS</p>
</body>
</html>
```

Dans le cas de l'utilisation de styles internes dans un site web, le CSS doit être reproduit dans chaque page, ce qui implique que toute mise à jour des styles requiert une modification individuelle de chaque fichier. Dans ce cas là il est préférable d'opter pour une feuille de style externe.

Styles en ligne

Cette méthode applique les styles directement à des éléments HTML en utilisant l'attribut `style`. Cependant, cette méthode n'est généralement pas recommandée car elle peut rendre le code HTML plus difficile à lire et à maintenir, surtout pour les projets complexes.

```
<!doctype html>

<html>

  <head>

    <meta charset="utf-8" />

    <title>Mes expérimentations CSS</title>

  </head>

  <body>

    <h1 style="color: blue;background-color: yellow;border: 1px
solid black;">

      Bonjour !

    </h1>

    <p style="color:red;">Ceci est mon premier exemple CSS</p>

  </body>

</html>
```

Il est fortement déconseillé d'adopter cette méthode ! Le code résultant devient difficile à maintenir (les modifications ne se restreignent plus à chaque page, mais touchent chaque élément au sein des pages !). De plus, mêler le CSS avec le HTML complexifie la lecture du code. En séparant le contenu de la mise en forme, non seulement la lisibilité du code s'améliore, mais aussi la collaboration au sein de l'équipe est plus facile.

Appliquer un style CSS

4 / 9

Sélecteurs

Sélecteurs CSS

Toute règle CSS commence par un sélecteur. Un sélecteur est une expression qui indique au navigateur à quelle entité HTML s'applique la règle CSS correspondante. Le ou les éléments ciblés par le sélecteur sont le sujet de ce sélecteur.

```
h1{
    color: bleu;
    background-color: yellow;
}

p{
    color: red;
}
```

Les sélecteurs CSS peuvent être classés en cinq catégories :

1. **Sélecteurs Simples** : Ils sont utilisés pour sélectionner des éléments en fonction de leur nom, de leur identifiant ou de leur classe.
2. **Sélecteurs Combinateurs** : Ils sont utilisés pour sélectionner des éléments en fonction de leur relation avec d'autres éléments.

3. **Sélecteurs d'Attributs** : Ils permettent de sélectionner des éléments en fonction de la présence d'un attribut particulier ou de sa valeur.

4. **Sélecteurs de Pseudo-Classe** : Ils sont utilisés pour sélectionner des éléments en fonction de leur état ou de leur interaction avec l'utilisateur.

5. **Sélecteurs de Pseudo-Éléments** : Ils permettent de sélectionner et de styliser des parties spécifiques d'un élément, comme le contenu avant ou après l'élément.

Dans ce premier niveau de cours nous allons développer les trois premières catégories de sélecteurs CSS. Les deux autres catégories seront développer dans le niveau intermédiaire.

Groupement de sélecteurs

Quand un groupe de déclarations CSS s'applique à plusieurs éléments distincts, on peut combiner les sélecteurs individuels en une liste. Par exemple, si j'ai le même CSS pour un **h1** et pour une classe **.special**, je pourrais écrire deux règles :

```
h1 {
  color: blue;
}

.special {
  color: blue;
}
```

ou bien une seule règle en combinant les sélecteurs, séparés par une virgule. L'espace est valide avant ou après la virgule.

```
h1,
.special {
  color: blue;
```

```
}
```

Quand on regroupe ainsi des sélecteurs, si l'un des sélecteurs est invalide la règle toute entière sera ignorée.

Dans l'exemple suivant, la règle avec le sélecteur de classe invalide sera ignorée, alors que le `h1` sera mis en forme comme prévu.

```
h1 {

  color: blue;

}
```

```
..special {

  color: blue;

}
```

En combinant les sélecteurs, la règle est considérée invalide et donc ignorée : ni `h1` ni les éléments de classe `.special` ne seront mis en forme.

```
h1,

..special {

  color: blue;

}
```

Spécificité

Dans de nombreux cas, deux sélecteurs différents peuvent cibler le même élément HTML. Considérons la feuille de style ci-dessous où j'ai une règle avec un sélecteur p qui colore les paragraphes en bleu, puis une règle qui colore en rouge les éléments dans la classe special.

```
.special {

  color: red;

}

p {

  color: blue;

}
```

Disons que dans notre document HTML, nous avons un paragraphe avec un attribut class valant special. Les deux règles pourraient s'appliquer. Selon vous, quelle sera la couleur du paragraphe ?

```
<p class="special">De quelle couleur suis-je?</p>
```

Le langage CSS a des règles pour déterminer quelle mise en forme appliquer en cas de collision de sélecteurs — elles sont appelées cascade et spécificité. Dans le bloc de code ci-dessous, nous avons défini deux règles pour le sélecteur p, mais le paragraphe finit par être coloré en bleu. En effet, la déclaration qui l'a défini en bleu apparaît plus tard dans la feuille de style et les styles ultérieurs remplacent les précédents. C'est la cascade en action.

Vous pouvez tester et modifier en ligne ce code en utilisant codepen.

4 / 9

Sélecteurs

5 / 9

Sélecteurs Simples

Sélecteur de type

Le sélecteur de type vise un élément HTML (une balise) dans votre document ; on l'appelle aussi sélecteur de balise ou d'élément.
Dans l'exemple ci-dessous, les sélecteurs span, em et strong sont utilisés.

```
em {

  font-style: italic;

  color: red;

}



strong {

  font-weight: bold;

  color: yellow;

}



span {

  background-color: yellow;

  color: blue;

}
```

Cela permet d'appliquer une mise en forme à chaque occurrence de < span> , < em> et < strong> .

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Sélecteurs de classe

Identifiable par un point (.), le sélecteur de classe choisit tous les éléments du document auxquels cette classe a été attribuée.
Dans l'illustration ci-après, nous avons définit une classe nommée .highlight et nous l'avons associé à divers emplacements du document.
Tous les éléments assortis de cette classe bénéficient d'une mise en évidence particulière.

```
.highlight {

    background-color: yellow;

}

<h1 class="highlight">Class selectors</h1>

<span class="highlight">L’ Agence Universitaire de la

Francophonie :</span>

<p>Association mondiale d’ établissements d'enseignement supérieur

et de recherche

francophones, <strong>l'AUF</strong> regroupe plus de 1000

établissements

universitaires sur tous les continents dans environ 120 pays.</p>
```

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Cibler des classes d'un élément donné

La création d'un sélecteur est possible pour cibler des éléments d'un type spécifique qui font partie d'une classe particulière.
Dans l'exemple qui suit, la classe 'highlight' crée une mise en évidence.
Cependant, cette mise en évidence diffère selon qu'elle s'applique à un élément ou à un titre <h1>.
Cette distinction est établie en utilisant un sélecteur de type pour l'élément ciblé, combiné avec la classe, sans aucun espace intermédiaire.

```
span.highlight {

    background-color: yellow;

}
```

```
h1.highlight {

    background-color: pink;

}
```

```
<h1 class="highlight">Class selectors</h1>
```

```
<span class="highlight">L' Agence Universitaire de la

Francophonie :</span>
```

```
<p>Association mondiale d’ établissements d'enseignement supérieur
et de recherche
francophones, <strong>l'AUF</strong> regroupe plus de 1000
établissements
universitaires sur tous les continents dans environ 120 pays.</p>
```

```
<h2 class="highlight"> Formation D-CLIC</h2>
```

Cette méthode limite la réutilisabilité du CSS : la règle n'est valable que pour ces éléments spécifiques, ce qui signifie qu'il serait nécessaire de créer un nouveau sélecteur afin d'étendre la portée de la règle à d'autres éléments.

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Sélecteurs d’ID

Un sélecteur d'ID débute avec le symbole # au lieu du point, cependant, il est utilisé en principe de la même manière qu'un sélecteur de classe.
Toutefois, il est important de noter qu'une ID ne peut être utilisée qu'une seule fois dans un document.

Ce sélecteur vise l'élément associé à l'ID en question ; il est possible de préfixer l'ID avec un sélecteur de type pour restreindre la cible uniquement à une correspondance précise entre l'élément et l'ID.

Jetons un œil à ces deux applications dans l'exemple ci-dessous :

```
#one {
    background-color: yellow;
}
```

```
h1#heading {

    color: rebeccapurple;

}

<h1 id="heading">Sélecteur d'ID</h1>


<p> Un sélecteur d'ID débute avec le symbole # au lieu du point,

cependant,

il est utilisé en principe de la même manière qu'un sélecteur de

classe.

Toutefois, il est important de noter qu'une ID ne peut être utilisée

qu'une seule fois dans un document.

</p>


<p id="one"> Ce sélecteur vise l'élément associé à

l'<strong>ID</strong> en question ; il est possible de préfixer

l'ID avec un sélecteur de type pour restreindre la cible uniquement

à une correspondance

précise entre l'élément et l'ID.</p>
```

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Sélecteur universel

Le sélecteur universel, symbolisé par un astérisque (*), englobe l'ensemble du document.
Dans l'exemple qui suit, le sélecteur universel est employé pour éliminer les marges de tous les éléments et colorer le texte en vert..

En conséquence, plutôt que de conserver l'espacement par défaut inséré par le navigateur entre les en-têtes et les paragraphes, tout est positionné en continuité.
Cette absence de distinction entre les paragraphes peut altérer la lisibilité.

```
* {

    margin: 0;

    color: green;

}
```

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Cibler un élément appartenant à plus d'une classe

Il est envisageable d'assigner multiples classes à un même élément, ce qui permet de les sélectionner individuellement.
Alternativement, vous pouvez cibler l'élément seulement lorsque toutes les classes sont présentes dans le sélecteur.
Cette approche peut se révéler pratique lorsque vous développez des éléments pouvant être combinés de diverses manières sur votre site.

L'exemple ci-dessous illustre trois balises < div> , chacune renfermant une note distincte.
Si une boîte est associée à la classe 'notebox', elle présente une bordure grise.
Si en plus elle appartient aux classes 'warning' ou 'danger', la couleur de la bordure est modifiée.

```
.notebox {

  border: 4px solid #666;

  padding: .5em;

}
```

```
.notebox.warning {
  border-color: orange;
  font-weight: bold;
}

.notebox.danger {
  border-color: red;
  font-weight: bold;
}
```

```
<div class="notebox">
    Il s'agit d'une note d'information.
</div>

<div class="notebox warning">
    Cette note indique un avertissement.
</div>
```

```
<div class="notebox danger">

    Cette note indique un danger !

</div>
```

```
<div class="danger">

    Cette note ne sera pas stylisée - elle doit également avoir la

classe notebox

</div>
```

On communique au navigateur la combinaison spécifique de classes en reliant les sélecteurs de classe consécutifs sans laisser de vide entre eux.

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Quel sélecteur CSS cible un élément HTML en utilisant son nom de balise ?

- Sélecteur d'ID
- Sélecteur d'élément
- Sélecteur de classe
- Sélecteur de type

Sélecteurs Simples

6 / 9

Sélecteurs d'attribut

Sélecteur de présence et de valeur

Ces sélecteurs vous donnent la capacité de choisir un élément en fonction de la présence d'un attribut spécifique (tel que href) ou en se fondant sur différentes correspondances avec la valeur d'un attribut donné.

Dans l'exemple ci-dessous, vous pouvez observer ces sélecteurs en action :

- **li[class]** cible tous les éléments <li> possédant un attribut class. De cette manière, tous les éléments de la liste sont visés, à l'exception du premier.
- **li[class="a"]** cible spécifiquement les <li> qui font partie de la classe 'a' et uniquement de celle-ci. Un élément <li> appartenant à la classe 'a' mais aussi à une autre classe ne sera pas pris en compte. Ce sélecteur cible donc le deuxième élément de la liste.
- **li[class~="a"]** cible chaque élément <li> dont l'attribut class contient 'a' dans sa liste de valeurs (séparées par des espaces). Les éléments deux et trois de la liste sont ainsi sélectionnés."

```
li[class] {

    font-size: 200%;

}
```

```
li[class="a"] {

    background-color: yellow;

}


li[class~="a"] {

    color: red;

}

<h1>Attribute presence and value selectors</h1>

<ul>

    <li>Point 1</li>

    <li class="a">Point 2</li>

    <li class="a b">Point 3</li>

    <li class="ab">Point 4</li>

</ul>
```

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Sélecteurs ciblant une sous-chaîne

Ces sélecteurs fournissent une approche plus précise pour cibler des sous-ensembles spécifiques au sein des valeurs des attributs. Par exemple, supposons que vous ayez défini les classes 'box-warning' et 'box-error', et que vous ayez l'intention de cibler les classes dont le nom commence par "box-". Dans ce cas, le sélecteur d'attribut [class^="box-"] vous offre précisément cette possibilité.

L'exemple qui suit illustre ces sélecteurs en action :

- **li[class^="a"]** vise toute valeur d'attribut qui débute par 'a'. Par conséquent, ce sélecteur cible les deux premiers éléments de la liste.
- **li[class$="a"]** s'applique à toute valeur d'attribut qui se termine par 'a'. Ainsi, ce sélecteur cible les éléments un et trois de la liste.
- **li[class*="a"]** correspond à toute valeur d'attribut contenant 'a' quelque part. En conséquence, ce sélecteur cible tous les éléments de la liste."

```
li[class^="a"] {
    font-size: 200%;
}

li[class$="a"] {
    background-color: yellow;
}

li[class*="a"] {
```

```
    color: red;
}

<h1>Attribute substring matching selectors</h1>

<ul>
    <li class="a">Point 1</li>
    <li class="ab">Point 2</li>
    <li class="bca">Point 3</li>
    <li class="bcabc">Point 4</li>
</ul>
```

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Sensibilité à la casse

Pour traiter les valeurs des attributs de manière insensible à la casse (majuscules ou minuscules), ajoutez le modificateur 'i' avant la parenthèse de fermeture. Ce modificateur indique au navigateur de reconnaître les caractères ASCII indépendamment de leur casse (a = A). En l'absence de ce modificateur, les valeurs seront interprétées en fonction de la sensibilité à la casse de la langue du document, le HTML étant sensible à la casse.

Dans l'exemple ci-dessous, le premier sélecteur cible les valeurs commençant par 'a', seule la première entrée de la liste est concernée, les deux suivantes commencent par 'A' majuscule. Le deuxième sélecteur est utilisé avec le modificateur insensible à la casse, ciblant ainsi tous les éléments de la liste.

Quel sélecteur cible un élément ayant un attribut "data-active" ?

- [active]
- [data-active]
- .data-active
- [data-="active"]

6 / 9

Sélecteurs d'attribut

7 / 9

Style en fonction de la position

Style en fonction de la position

Nous étudions actuellement le style d'un élément qui s'adapte en fonction de sa position dans le document.

De nombreux sélecteurs permettent de réaliser ce type de comportement :

- Sélecteur d'espace (Espace descendant) :

Ce sélecteur cible tous les éléments descendants de l'élément précédent, peu importe leur niveau d'imbrication. Tous les éléments correspondants seront stylisés.

Exemple HTML :

```
<div>
```

```
    <p>Paragraphe 1</p>
```

```
    <div>
```

```
        <p>Paragraphe 2</p>
```

```
    </div>
```

```
</div>
```

Exemple CSS :

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

Résultat : Les deux éléments <p> seront stylisés en bleu.

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

- Sélecteur de > (Enfant direct) :

Ce sélecteur cible uniquement les éléments qui sont des enfants directs de l'élément précédent.

Exemple HTML :

```
<div>
```

```
    <p>Paragraphe 1</p>
```

```
    <div>
```

```
        <p>Paragraphe 2</p>
```

```
    </div>
```

```
</div>
```

Exemple CSS :

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

Résultat : Seul l'élément `<p>` dans le premier `<div>` sera stylisé en rouge.

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

- ### Sélecteur + (Frère immédiat) :

Ce sélecteur cible un élément qui suit immédiatement l'élément précédent et qui partage le même parent.

Exemple HTML :

```
<h2> Titre 1 </h2>
```

```
<p>Paragraphe 1</p>
```

```
<h2>Titre 2</h2>
```

```
<p>Paragraphe 2</p>
```

Exemple CSS :

```
h2 + p {
```

```
   font-weight: bold;
```

```
}
```

Résultat : Le deuxième paragraphe sera en gras car il suit immédiatement un élément `<h2>`.

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

- ### Sélecteur ~ (Frère général) :

Ce sélecteur cible tous les éléments qui partagent le même parent et qui suivent l'élément précédent.

Exemple HTML :

```
<h2>Titre 1</h2>
```

```
<p>Paragraphe 1</p>
```

```
<h2>Titre 2</h2>
```

```
<p>Paragraphe 2</p>
```

Exemple CSS :

```
h2 ~ p {
```

```
    color: green;
```

```
}
```

Résultat : Les deux paragraphes seront en vert car ils suivent des éléments `<h2>`.

Vous pouvez tester et modifier ce code en ligne en utilisant codepen.

Exercice:

L'exemple présent actuellement contient seulement deux règles. Pour enrichir cela, vous pouvez introduire une nouvelle règle qui applique la couleur rouge au balisage `<span>` uniquement lorsqu'il est contenu à l'intérieur d'un élément `<p>`.

Si tout se déroule comme prévu, après avoir enregistré le CSS et actualisé la page HTML dans votre navigateur, le texte enveloppé par un `<span>` dans le premier paragraphe devrait devenir rouge, tandis que le `<span>` dans le premier élément de la liste restera inchangé.

```
li em {
```

```
    color: red;
```

```
}
```

```
h1 + p {
```

```
    font-size: 200%;
}
<h1>Je suis une rubrique de niveau 1</h1>

<p>This is a paragraph of text. Dans le texte se trouvent un
<span>élément span</span> et un <a
href="http://example.com">lien</a>.</p>

<p>Il s'agit du deuxième paragraphe. Il contient un <em>élément</em>
mis en valeur.</p>

<ul>
   <li>Point <span>un</span></li>
   <li>Point deux</li>
   <li>Point <em>trois</em></li>
</ul>
```

Vous pouvez compléter ce code en utilisant codepen..

Style en fonction de la position

8 / 9

Propriétés CSS

Liste des principales propriétés

Propriété : color

- Explication : Définit la couleur du texte.
- Valeurs possibles : noms de couleur (red, blue), codes hexadécimaux (#FF0000), codes RGB (rgb(255, 0, 0)).
- Exemple : color: blue;

Propriété : font-size

- Explication : Contrôle la taille de la police.
- Valeurs possibles : pixels (12px), em (1.5em), pourcentage (150%).
- Exemple : font-size: 16px;

Propriété : text-align

- Explication : Aligne le texte dans un élément.
- Valeurs possibles : left, right, center, justify.
- Exemple : text-align: center;

Propriété : background-color

- Explication : Définit la couleur de l'arrière-plan.
- Valeurs possibles : mêmes valeurs que pour la propriété color.
- Exemple : background-color: #F0F0F0;

Propriété : border

- Explication : Crée une bordure autour de l'élément.

- Valeurs possibles : taille (1px), style (solid, dotted), couleur (valeurs de couleur).
- Exemple : border: 1px solid black;

Propriété : margin

- Explication : Espace entre l'élément et les éléments voisins.
- Valeurs possibles : pixels, pourcentage, auto.
- Exemple : margin: 10px;

Propriété : padding

- Explication : Espace entre le contenu de l'élément et sa bordure.
- Valeurs possibles : pixels, pourcentage.
- Exemple : padding: 20px;

Propriété : display

- Explication : Contrôle le comportement d'affichage de l'élément.
- Valeurs possibles : block, inline, inline-block, flex, grid, etc.
- Exemple : display: inline-block;

Propriété : position

- Explication : Contrôle le positionnement de l'élément.
- Valeurs possibles : static, relative, absolute, fixed.
- Exemple : position: absolute;

Propriété : width

- Explication : Définit la largeur de l'élément.
- Valeurs possibles : pixels, pourcentage, auto.
- Exemple : width: 300px;

Propriété : font-family

- Explication : Spécifie la police à utiliser pour le texte.
- Valeurs possibles : Noms de polices, familles génériques (serif, sans-serif, monospace, etc.).
- Exemple : font-family: "Arial", sans-serif;

Propriété : line-height

- Explication : Contrôle l'espacement vertical entre les lignes de texte.
- Valeurs possibles : Valeurs numériques, valeurs sans unités, pourcentages.

- Exemple : `line-height: 1.5;`

Propriété : `font-weight`

- Explication : Définit l'épaisseur de la police (gras ou normal).
- Valeurs possibles : `normal`, `bold`, `bolder`, `lighter`, valeurs numériques.
- Exemple : `font-weight: bold;`

Propriété : `text-decoration`

- Explication : Ajoute ou supprime la décoration du texte, comme les soulignements.
- Valeurs possibles : `none`, `underline`, `overline`, `line-through`, `initial`.
- Exemple : `text-decoration: underline;`

Propriété : `border-radius`

- Explication : Définit le rayon des coins arrondis d'un élément avec une bordure.
- Valeurs possibles : Valeurs numériques (px, em, %), valeurs multiples pour les coins individuels.
- Exemple : `border-radius: 10px;`

Propriété : `box-shadow`

- Explication : Ajoute une ombre à un élément.
- Valeurs possibles : Position de l'ombre, décalage horizontal, décalage vertical, flou, couleur.
- Exemple : `box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);`

Propriété : `text-transform`

- Explication : Modifie la casse du texte (majuscules, minuscules, etc.).
- Valeurs possibles : `uppercase`, `lowercase`, `capitalize`, `none`.
- Exemple : `text-transform: uppercase;`

Propriété : `position`

- Explication : Définit la méthode de positionnement d'un élément.
- Valeurs possibles : `static`, `relative`, `absolute`, `fixed`, `sticky`.
- Exemple : `position: absolute;`

Propriété : `z-index`

- Explication : Contrôle l'empilement d'éléments positionnés.
- Valeurs possibles : Valeurs numériques, `auto`.
- Exemple : `z-index: 1;`

Propriété : `opacity`

- Explication : Contrôle le degré de transparence d'un élément.
- Valeurs possibles : Valeurs entre 0 (complètement transparent) et 1 (pleinement opaque).
- Exemple : `opacity: 0.7;`

Une liste plus comlète est disponible dans ce lien.

Propriétés CSS