Priorité des sélecteurs CSS :

En CSS, les sélecteurs ont des niveaux de priorité qui déterminent quel style sera appliqué si plusieurs sélecteurs ciblent le même élément. Voici un guide détaillé de la priorité des sélecteurs CSS, avec des exemples et des explications sur l'utilisation de `!important` pour forcer un style.

Ordre de priorité des sélecteurs CSS

Les sélecteurs CSS sont priorisés en fonction de leur spécificité. La spécificité est calculée en fonction des types de sélecteurs utilisés dans la règle CSS. Voici l'ordre de spécificité, du plus faible au plus fort :

1. **Sélecteurs de type (ou tag)** : sélectionne les éléments par leur nom de balise (par ex., `div`, `p`, `h1`).
2. **Sélecteurs de classe** : sélectionne les éléments par leur classe (par ex., `.classe`, `.btn`).
3. **Sélecteurs d'identifiant (ID)** : sélectionne les éléments par leur identifiant (par ex., `#monId`).
4. **Pseudo-classes** : ajoutent un état ou un comportement spécifique à un élément (par ex., `:hover`, `:first-child`).
5. **Pseudo-éléments** : ciblent une partie spécifique d'un élément (par ex., `::before`, `::after`).
6. **Sélecteurs en ligne** : les styles appliqués directement dans les balises HTML via l'attribut `style` ont la priorité la plus élevée, en dehors des `!important`.

Calcul de la spécificité

La spécificité est calculée avec un système de score. En général :

- Un ID ajoute **100** points de spécificité.
- Une classe ajoute **10** points.
- Un sélecteur de type ajoute **1** point.
- Un sélecteur pseudo-classe ajoute **10** points, comme une classe.
- Un pseudo-élément ajoute **1** point.

La priorité est donc calculée en fonction du total de ces points.

Exemple de spécificité

Supposons que nous ayons les sélecteurs suivants et leur spécificité correspondante :

1. `div` : spécificité de **1** (balise)
2. `.classe` : spécificité de **10** (classe)
3. `#monId` : spécificité de **100** (ID)
4. `div .classe` : spécificité de **11** (balise + classe)
5. `div #monId` : spécificité de **101** (balise + ID)
6. `a:hover` : spécificité de **11** (balise + pseudo-classe)
7. `#monId:hover` : spécificité de **110** (ID + pseudo-classe)

L’élément avec la plus haute spécificité l’emporte en cas de conflit. Voici un exemple HTML et CSS pour illustrer cela :

```html
<!DOCTYPE html>
<html lang="en">
<head>
<style>
  div { color: blue; }          /* Spécificité : 1 */
  .classe { color: green; }      /* Spécificité : 10 */
  #monId { color: red; }         /* Spécificité : 100 */
  div .classe { color: yellow; } /* Spécificité : 11 */
  div#monId { color: purple; }   /* Spécificité : 101 */
</style>
</head>
<body>

<div id="monId" class="classe">Texte exemple</div>

</body>
</html>
```

Dans cet exemple :

- L’élément `div` a l’ID `monId` et la classe `classe`.
- Les couleurs potentielles sont définies par plusieurs sélecteurs.
- La couleur finale sera `purple` parce que `div#monId` a la spécificité la plus élevée (101).

Utilisation de `!important`

Le mot-clé `!important` permet de forcer une règle CSS à s’appliquer en dépit de la spécificité des sélecteurs. Ce mot-clé devrait être utilisé avec précaution, car il peut rendre le code difficile à maintenir.

Exemple avec `!important`

Voici comment `!important` change la priorité des sélecteurs :

```html
<!DOCTYPE html>
<html lang="en">
<head>
<style>
  div { color: blue !important; }          /* Spécificité de 1, mais avec !important */
  .classe { color: green; }                /* Spécificité de 10 */
  #monId { color: red; }                   /* Spécificité de 100 */
  div .classe { color: yellow; }           /* Spécificité de 11 */
  div#monId { color: purple !important; }  /* Spécificité de 101 avec !important */
</style>
</head>
<body>

<div id="monId" class="classe">Texte exemple</div>

</body>
</html>
```

Dans cet exemple :

- `div { color: blue !important; }` force le texte à être en bleu, même si d'autres règles de style sont plus spécifiques.
- Si un autre `!important` avec une spécificité plus élevée est appliqué, c'est celui-là qui l'emportera.

Précautions d'utilisation de `!important`

- **Limiter l'utilisation** : `!important` rend le code difficile à déboguer et à maintenir. Si vous en avez trop, cela peut être un signe qu'il y a un problème de structure.
- **Utiliser uniquement pour des exceptions** : Utilisez-le dans des cas exceptionnels où il n'y a pas d'autre option.
- **Réserver aux styles de dernière minute** : Par exemple, lorsque vous devez surcharger des styles provenant d'une bibliothèque tierce (comme Bootstrap) sans modifier directement le fichier CSS de la bibliothèque.

Résumé :

Type de sélecteur	Exemple	Spécificité	Priorité en cas de conflit
Balise(Type)	Div	1	Faible
Classe	.classe	10	Supérieur au sélecteur de balise
ID	#monId	100	Supérieur aux classes et balises
Pseudo-classe	:hover	10	Egal aux classes
Pseudo-élément	::before	1	Faible,équivalent à un sélecteur de balise
En ligne	Style= 'color :red ; '	N/A	Supérieur à tout sauf !important

En cas de doute ou de nécessité de surcharger un style, `!important` est un moyen de forcer une règle, mais il est recommandé de l'utiliser avec parcimonie pour éviter des conflits difficiles à gérer.

Rappel sur les pseudo-classes et les pseudo-éléments

Pseudo-classes

Les pseudo-classes ciblent un état particulier d'un élément. Elles sont précédées d'un seul deux points : .

Exemples courants de pseudo-classes :

Pseudo-classe	Exemple
:hover - - - lorsque l'utilisateur survole un élément .	a:hover { color: blue; }
:active - - - Lorsque l'élément est activé (par exemple, lorsqu'un lien est cliqué).	a:active { color: red; }
:focus - - - Lorsqu'un élément est focalisé, comme un champ de formulaire cliqué ou tabulé.	input:focus { border-color: green; }
:nth-child(n) - - -Cible le nième enfant d'un élément parent.	p:nth-child(2) { font-weight: bold; }
:first-child et **:last-child - - -** Cible respectivement le premier et le dernier enfant d'un élément parent.	p:first-child { color: orange; }
:not(selector) - - - Cible les éléments qui ne correspondent pas à un sélecteur particulier.	div:not(.active) { opacity: 0.5; }
:disabled, :checked, :required, etc. - - - Cible les états des éléments de formulaire.	input:disabled { background-color: grey; }

Pseudo-éléments :

Les pseudo-éléments permettent de **cibler une partie spécifique d'un élément** (comme le premier mot, ou un contenu ajouté). Ils sont précédés de deux-points ::.

Exemples courants de pseudo-éléments :

Pseudo-élément	Exemple
::before - - - Insère du contenu avant un élément (utile pour les icônes ou décorations).	p::before { content: "👉 "; color: blue; }
::after - - - Insère du contenu après un élément (souvent utilisé pour des décorations ou séparateurs).	p::after { content: " ✓"; color: green; }
::first-line - - - Cible uniquement la première ligne de texte d'un élément de bloc.	p::first-line { font-weight: bold; }
::first-letter - - - Cible la première lettre d'un élément de bloc	p::first-letter { font-size: 2em; color: red; }
::selection - - - Cible la portion de texte sélectionnée par l'utilisateur.	::selection { background-color: yellow; color: black; }

Rappel des differences :

Type	Syntaxe	Utilisation
Pseudo-classe	:pseudo-class	Cible un état de l'élément
Pseudo-élément	::pseudo-element	Cible une partie spécifique de l'élément

QCM sur les pseudo-classes et pseudo-éléments

1. Quelle pseudo-classe cible un lien déjà visité ?	A) :visited B) :link C) :hover D) :active
2. Quelle pseudo-classe est utilisée pour styliser un élément lorsque l'utilisateur passe la souris dessus ?	A) :hover B) :focus C) :active D) :visited
3. Quelle pseudo-classe cible un élément activé, par exemple lors du clic ?	A) :hover B) :focus C) :active D) :checked
4. Quelle pseudo-classe stylise uniquement le premier élément enfant d'un parent ?	A) :nth-child(1) B) :first-child C) :first D) :only-child
5. Quelle pseudo-classe cible le champ focalisé (cliqué ou tabulé) d'un formulaire ?	A) :hover B) :focus C) :checked D) :active
6. Quelle est la syntaxe correcte pour styliser uniquement les paragraphes pairs dans un conteneur ?	A) p:nth-child(odd) B) p:nth-child(even) C) p:nth-child(2n+1) D) p:nth-last-child(even)
7. Quelle pseudo-classe stylise le seul élément enfant d'un parent ?	A) :only B) :last-child C) :only-child D) :unique-child
8. Quelle est la pseudo-classe correcte pour cibler le dernier enfant d'un type donné ?	A) :last-type B) :nth-last-child(1) C) :last-of-type D) :last-child
9. Quelle pseudo-classe sélectionne les cases à cocher cochées dans un formulaire ?	A) :checked B) :enabled C) :active D) :required
10. Quelle pseudo-classe cible un élément qui ne correspond pas à un sélecteur donné ?	A) :without B) :not C) :none D) :no
11. Quelle est la différence entre :nth-child(n) et	A) Aucune différence

:nth-of-type(n) ?	B) :nth-child cible tous les éléments enfants C) :nth-of-type cible les éléments d'un même type D) :nth-of-type cible uniquement le premier élément
12. Quelle pseudo-classe cible les éléments requis dans les formulaires ?	A) :required B) :optional C) :enabled D) :valid
13. Quelle pseudo-classe cible un élément en état invalide ?	A) :invalid B) :disabled C) :error D) :not-valid
14. Quelle pseudo-classe stylise le premier mot d'un élément de texte ?	A) ::first-word B) ::first-letter C) ::first-line D) ::word
15. Quelle est la syntaxe correcte pour insérer du contenu avant un élément en CSS ?	A) :before B) ::before C) :start D) ::first
16. Quelle pseudo-classe cible le champ désactivé d'un formulaire ?	A) :unchecked B) :disabled C) :readonly D) :invalid
17. Quelle pseudo-classe cible l'élément sélectionné par l'utilisateur, par exemple du texte ?	A) :selected B) ::selection C) :chosen D) ::highlight
18. Quelle pseudo-classe stylise un lien non encore visité ?	A) :hover B) :link C) :active D) :target
19. Quelle est la différence entre `::before` et `::after` ?	A) ::before ajoute du contenu après l'élément B) ::after ajoute du contenu avant l'élément C) ::before ajoute du contenu avant l'élément et ::after ajoute après D) Aucune différence
20. Comment sélectionner uniquement les premiers éléments dans une liste de paragraphes ?	A) p:first-child B) p:nth-child(1) C) p:nth-of-type(1) D) Toutes les réponses sont correctes
21. Quel pseudo-élément stylise la première lettre d'un élément de texte ?	A) ::first-letter B) ::first-word C) ::first-line D) :first-child
22. Quel pseudo-élément peut être utilisé pour styliser la première ligne d'un paragraphe ?	A) ::line B) :first-line C) ::first-line D) ::first-letter

23. Quelle pseudo-classe est utilisée pour cibler l'élément actuellement ciblé dans l'URL ?	A) :active B) :target C) :focus D) :hover
24. Comment appliquer un style à un champ de formulaire non requis ?	A) :required B) :optional C) :empty D) :disabled
25. Quelle pseudo-classe sélectionne tous les éléments parents qui contiennent au moins un élément enfant ?	A) :not-empty B) :has C) :not(:empty) D) :filled
26. Comment appliquer un style à l'élément qui est l'avant-dernier enfant d'un type donné ?	A) :nth-last-child(2) B) :nth-last-of-type(2) C) :nth-child(2) D) :nth-of-type(2)
27. Quelle pseudo-classe sélectionne un champ de formulaire valide ?	A) :checked B) :valid C) :enabled D) :complete
28. Quelle pseudo-classe sélectionne un élément qui n'a pas d'enfants ?	A) :none B) :empty C) :null D) :blank
29. Quelle pseudo-classe applique un style aux liens qui contiennent une ancre correspondant au fragment de l'URL ?	A) :focus B) :link C) :target D) :visited
30. Comment sélectionner le troisième élément d'un groupe d'éléments identiques ?	A) :nth-child(3) B) :nth-of-type(3) C) :last-of-type(3) D) A et B sont corrects