

Semaine 09 - Pseudo-éléments CSS et propriétés CSS personnalisées

Pseudo-éléments CSS

Notions de base sur les pseudo-éléments

Un **pseudo-élément** est un mot-clé ajouté à un sélecteur qui permet de mettre en forme certaines parties de l'élément ciblé par la règle. La syntaxe est la suivante:

```
sélecteur::pseudo-élément {  
    propriété: valeur;  
}
```

Il est à noter que depuis CSS3, on utilise deux fois le caractère deux-points « :: » pour préfixer les pseudo-éléments (afin de pouvoir les différencier des pseudo-classes). Toutefois, la plupart des navigateurs prennent en charge les pseudo-éléments pour lesquels le préfixe n'est formé que d'un seul « : ».

Ainsi, un **pseudo-élément** CSS peut être utilisé pour **styler** les parties spécifiées d'un élément HTML comme tel. Comme dans cet exemple, il peut s'agir de styler la **première lettre** d'un paragraphe (*::first-letter*), ou encore sa **première ligne** (*::first-line*).

```
p::first-letter {  
    font-size: 3.5rem;  
    font-style: italic;  
    color: #02ae90;  
}  
  
p::first-line {  
    font-style: italic;  
    color: #02ae90;  
}
```

Le résultat est le suivant:

C' était le Lapin Blanc qui revenait en trotinant, et qui cherchait de tous côtés, d'un air inquiet, comme s'il avait perdu quelque chose ; Alice l'entendit qui marmottait : « La Duchesse ! La Duchesse ! Oh ! mes pauvres pattes ; oh ! ma robe et mes moustaches ! Elle me fera guillotiner aussi vrai que des furets sont des furets ! Où pourrais-je bien les avoir perdus ? » Alice devina tout de suite qu'il cherchait l'éventail et la paire de gants paille, et, comme elle avait bon cœur, elle se mit à les chercher aussi ; mais pas moyen de les trouver.

Ajout de contenu avec les pseudo-éléments

Les pseudo-éléments peuvent aussi être utilisés pour ajouter du contenu cosmétique **avant** (*::before*) ou **après** (*::after*) le contenu actuel d'un élément HTML. Pour cela, on doit utiliser et définir la **propriété CSS : content**. L'élément ajouté peut être du texte, un symbole graphique, un icône UTF8, une image, etc. La syntaxe de base sera alors la suivante:

```
sélecteur::before {  
    content: "contenu à ajouter avant le contenu actuel de l'élément HTML";  
}
```

Ou :

```
sélecteur::after {  
    content: "contenu à ajouter après du contenu actuel de l'élément HTML";  
}
```

Dans le code qui suit, on ajoute deux icônes UTF-8³ avant et après le contenu actuel d'un élément de type paragraphe:

```
p::before {  
    content: "\2771";  
}  
  
p::after {  
    content: "\2714";  
}
```

Le résultat est le suivant:

➤ Mettre un élément avant avec *::before* et après avec *::after* ✓

Pseudo-éléments et normalisation

Il est à noter que les pseudo-éléments ne sont pas compris avec le sélecteur CSS universel *. Ainsi, il faudra s'assurer de leur appliquer la même normalisation que les autres éléments HTML:

```
*, ::before, ::after {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}
```

³ Voir: <https://www.utf8icons.com/>

Ajout et stylisation de contenu vide

Il est aussi possible d'ajouter un contenu, comme une **boîte vide** en utilisant (*::before* ou *::after*) et de la formater. Pour cela, l'**élément HTML parent** doit avoir une **position CSS** définie autre que celle du flux régulier du document (*static*), habituellement: **position: relative**. Et, celle du pseudo-élément doit être en absolu: **position: absolute**, afin d'être en mesure d'ajuster sa position à celle de l'élément parent. Le **contenu vide** est exprimé comme suit:

```
sélecteurCSS::pseudo-élément ::before ou ::after {  
    content: "";  
}
```

En **combinaison avec la position ABSOLUTE du pseudo-élément**, on utilise habituellement une combinaison des propriétés: **top**, **right**, **bottom**, **left**, **width** et **height** pour définir la forme et l'emplacement final du pseudo-élément.

Par exemple, dans le code qui suit, on ajoute un rectangle en avant du contenu d'un élément de paragraphe:

```
p{  
    position: relative;  
    ....  
}  
  
p::before {  
    position: absolute;  
    content: "";  
    top: 0%;  
    bottom: 0%;  
    left: 0%;  
    width: 1.8rem;  
    border: 1px solid #ff7673;  
}
```

Le résultat est le suivant:



Il est à noter que les pseudo-éléments peuvent également faire l'objet de transitions ou d'animation CSS. Voir à ce sujet les exemples du cours de la semaine 9.

Stylisation d'éléments de formulaire

Malheureusement pour des raisons techniques et historiques, la majorité des éléments de formulaires ne s'allient pas très bien avec CSS et il est difficile de les styliser. Voir à ce sujet l'article suivant de Mozilla:

https://developer.mozilla.org/fr/docs/Learn/Forms/Styling_web_forms .

Nous allons examiner ici une astuce pour **styler des boutons radios**. Les principales étapes sont les suivantes :

- Les éléments **<input>** sont retirés de l'affichage avec : **display:none**;
- Les éléments **<label>** sont positionnés en CSS avec **position:relative**;
- Des **pseudo-éléments** sont déclarés pour les légendes avec **label::before**;
- L'aspect des pseudo-éléments est modifié lorsqu'un bouton-radio est coché avec **input:checked ~ label::before**.

Pour cet exemple le code HTML est le suivant:

```
<div class="les-choix">
  <div class="choix">
    <input type="radio" id="reponse1" name="reponse">
    <label for="reponse1">RÉPONSE 1</label>
  </div>
  <div class="choix">
    <input type="radio" id="reponse2" name="reponse">
    <label for="reponse2">RÉPONSE 2</label>
  </div>

  <div class="choix">
    <input type="radio" id="reponse3" name="reponse">
    <label for="reponse3">RÉPONSE 3</label>
  </div>
</div>
```

Le code CSS est le suivant:

```
/*Le <input> est retiré de l'affichage*/
.choix>input {
  display: none;
}

/*On gère la position de la légende avec la gestion d'un pseudo-élément*/
.choix>label {
  position: relative;
  margin-left: 2rem;
  cursor: pointer;
}

.choix>label::before {
  position: absolute;
  content: "";

  bottom: 0%;
  right: 100%;
  width: 2rem;
  height: 2rem;
  transform: translateX(-25%);


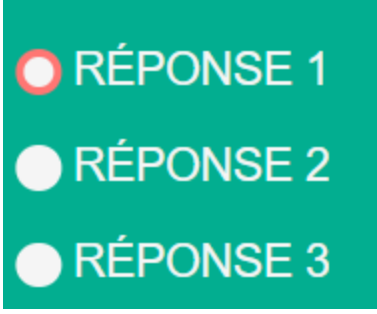
  border-radius: 50%;
}
```

```
background-color: whitesmoke;

transition: background-color 0.5s, border 0.5s;
}

/*Bouton radio sélectionné*/
.choix>input:checked~label::before {
    background-color: whitesmoke;
    border: 7px solid #ff7673;
}
```

Le résultat est le suivant:

Aspect des bouton radios	Aspect d'un bouton radio sélectionné
	

Quelques références utiles

- Mozilla (Septembre 2022). Pseudo-éléments. Repéré à : <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-elements>
- W3Schools (s.-d.). CSS Pseudo-elements. Repéré à : https://www.w3schools.com/css/css_pseudo_elements.asp

Propriétés CSS personnalisées et/ou variables CSS

Les propriétés CSS personnalisées (« *custom properties* » en anglais, aussi appelées *variables CSS*) sont des **entités** définies par les développeurs Web, contenant des **valeurs spécifiques utilisables à travers le document**.

Dans le texte qui suit, on prend le **choix éditorial** d'utiliser le vocable **variable CSS**.

Elles sont initialisées avec des propriétés personnalisées (par exemple `--couleur: black;`) et accessibles en utilisant la notation spécifique **var()** (par exemple `: color: var(--couleur);`).

Celles-ci sont particulièrement utiles pour **faciliter la maintenance du code**. Des sites et applications web complexes peuvent avoir des feuilles de style où de nombreuses valeurs sont répétées. Ainsi, la même couleur pourra être utilisée à des centaines d'endroits où il faudra la mettre à jour si besoin. Les propriétés personnalisées permettent de stocker une valeur à un endroit puis de réutiliser cette valeur (on factorise ainsi le code).

Le fait qu'elles puissent être mises à jour en JavaScript ouvre également des possibilités très intéressantes.

Syntaxe des variables CSS

Les noms des variables CSS sont **préfixés** par **deux tirets** : `--` (ex : `--nom-exemple`) et leur valeur peut être n'importe quelle **valeur CSS valide**. Par exemple;

```
--un-mot-cle: left;
--une-couleur: #0000ff;
--une-valeur-complexe: 3px 6px rgb(20, 32, 54);
```

Portée des variables CSS

La **portée** des variables CSS est celle des éléments sur lesquels elles sont déclarées. Ces propriétés personnalisées **contribuent à la cascade** i.e. qu'elles tiennent compte de l'héritage et de l'imbrication des éléments HTML. Les variables CSS peuvent donc avoir une **portée globale** ou **locale**.

Les **variables CSS globales** sont accessibles/utilisées dans tout le document, tandis que les **variables locales** ne peuvent être utilisées qu'à l'intérieur du sélecteur où elles sont déclarées ou par leurs nœuds enfants.

Cette note de cours traite plus particulièrement des variables CSS à portée globale.

Déclaration de variables CSS à portée globale

Pour créer une **variable CSS** avec une **portée globale**, il faut la déclarer avec la pseudo-classe **:root** qui permet de cibler la racine de l'arbre représentant le document. Pour un document HTML, **:root** ciblera donc l'élément `<html>` et aura le même comportement que le sélecteur `<html>` mais **sa spécificité sera plus forte**.

Dans l'exemple qui suit, nous déclarons **trois variables CSS globales** (`--premiere-couleur`, `--seconde-couleur` et `--espacement`):

```
:root {
  --premiere-couleur: #488cff;
  --deuxieme-couleur: #ffff8c;
  --espacement: 1rem;
}
```

IMPORTANT! Il est à noter que les noms donnés aux propriétés CSS personnalisées sont sensibles à la casse. Alors, attention aux majuscules et aux minuscules...

Utilisation de variables CSS

Pour récupérer la valeur déclarée d'une propriété CSS personnalisée, nous utilisons simplement la **fonction CSS var()** dans la feuille de style. La syntaxe est la suivante:

- styleCSS: **var(--propriété Personnalisée);**

En voici un exemple, pour l'entête et le pied de page d'une page HTML, qui récupère les valeurs des propriétés déclarées ci-haut:

```
header, footer {
  /*.....*/
  /*Les couleurs*/
  background-color: var(--premiere-couleur);
  color: var(--deuxieme-couleur);

  /*Le padding gauche*/
  padding-left: var(--espacement);
}
```

Utilisation de variables CSS avec valeur par défaut ou de secours

Il est possible de fixer des **valeurs par défaut** aux références données. En effet la fonction **var()** **accepte un paramètre supplémentaire** pour faire une autre référence (*fallback* en anglais) au cas où une variable CSS ne serait pas déjà déclarée ou définie. Par exemple:

```
color: var(--autre-couleur, #508e4c);
```

Dans l'exemple ci-haut, si la propriété **--autre-couleur** n'a pas été déclarée, c'est la valeur **#508e4c** qui sera utilisée par le navigateur.

Utilisation de variables CSS locales et animation

Les variables CSS peuvent fonctionner avec les animations CSS, afin de rendre les **règles @keyframes réutilisables** avec des valeurs différentes. Pour cela, nous devons définir les propriétés CSS personnalisées dans les sélecteurs qui ciblent les éléments que l'on souhaite animer, et y faire référence avec la fonction **var()** dans le bloc **@keyframes**. Par exemple, le code qui suit permet d'animer le rebond de 3 éléments avec **différentes hauteurs** en utilisant la même règle **@keyframes** :

```
@keyframes rebondir {
  from {
    transform: translateY(0vh);
  }
  to {
    transform: translateY(var(--hauteur-rebond));
  }
}
```

```
.ballon {
  animation: rebondir alternate infinite cubic-bezier(0.2, 0.65, 0.6, 1);
}
.ballon.un {
  --hauteur-rebond: -5vh;
  animation-duration: 200ms;
}
.ballon.deux {
  --hauteur-rebond: -10vh;
  animation-duration: 300ms;
}
.ballon.trois {
  --hauteur-rebond: -15vh;
  animation-duration: 400ms;
}
```

Modification d'une variable CSS avec JavaScript

Les variables CSS ont accès au DOM, ce qui signifie que l'on peut les modifier avec JavaScript de la même façon que les propriétés standards. Par exemple dans le code qui suit, on récupère le sélecteur correspondant à la pseudo-classe **:root** et on modifie la valeur de style pour la propriété **--premiere-couleur** déclarée:

```
//Récupérer le sélecteur :root
let selecteurRoot = document.querySelector(":root");
//Modifier la valeur de la propriété
selecteurRoot.style.setProperty("--premiere-couleur", "#2b3c72");
```

À explorer...

Cette note de cours ne fait que vous initier aux variables CSS. Il y a en effet de **NOMBREUSES POSSIBILITÉS** pour l'usage de celles-ci, tant sur le plan de l'**animation** que sur leur **manipulation avec JavaScript**... À vous d'en explorer les multiples possibilités...

Quelques références utiles

- EnvatoTuts (Mai 2017). How to Use CSS Variables for Animation. Repéré à : <https://webdesign.tutsplus.com/tutorials/how-to-use-css-variables-for-animation-cms-28868>
- Mozilla (Octobre 2022). :root - CSS Feuilles de styles en cascade. Repéré à : <https://developer.mozilla.org/fr/docs/Web/CSS/:root>
- Mozilla (Février 2023). Variables CSS. Repéré à : https://developer.mozilla.org/fr/docs/Web/CSS/Using_CSS_custom_properties
- W3Shool(s.-d.). CSS Variables - The var() Function. Repéré à : https://www.w3schools.com/css/css3_variables.asp