Université de Strasbourg

Conception de pages Web (HTML/CSS)

Introduction à CSS

Enzo Doyen
enzo.doyen@unistra.fr

2026 - LGA1DMT2

🕮 Plan

Récapitulatif du cours précédent

Jusqu'à présent, nous avons vu comment créer un site Web basique en utilisant HTML.

Nous avons également vu comment ajouter des éléments basiques : des titres, des paragraphes, des images, des liens, etc.

Toutefois, nos pages Web sont encore très basiques et peu esthétiques. On peut améliorer cela en utilisant le langage CSS.

I. Présentation de CSS

CSS, c'est quoi ?

- **C**ascading **S**tyle **S**heets (littéralement « feuille de style en cascade ») est un langage qui permet de définir l'apparence de contenus HTML.
- En particulier, il permet de définir un ensemble de règles de style qui seront appliquées à des éléments HTML spécifiques.

CSS, c'est quoi ?

CSS repose sur le principe de propriété/valeur (comme pour les attributs en HTML, p. ex. `<img src="mon_image.jpg">`).

On indique le **nom d'une propriété**, suivi d'un **deux-points**, puis du **nom d'une valeur**. Chaque ligne se termine par un **point-virgule**.

```
1  propriété1: valeur1;
2  propriété2: valeur2, "valeur3 à plusieurs mots";
```

CSS, c'est quoi ?

La liste complète des propriétés CSS est disponible sur MDN : https://developer.mozilla.org/fr/docs/Web/CSS/Reference/Properties

On peut par exemple utiliser la propriété `color` pour changer la couleur du contenu textuel d'un élément, et la propriété `font-family` pour changer la police du texte de cet élément :

```css
color: red; /* une seule valeur */
font-family: "Open Sans", sans-serif; /* 2 valeurs */
```

Intégration de CSS à du code HTML

On peut styliser nos balises directement à l'aide de l'attribut `style` en HTML :

```HTML
<p style="color: red;">Texte en rouge</p>
```

Intégration de CSS à du code HTML

On peut styliser nos balises directement à l'aide de l'attribut `style` en HTML :

```HTML
1  <p style="color: red;">Texte en rouge</p>
```

Utiliser l'attribut `style` n'est toutefois **pas recommandée** et on préfère plutôt l'une des deux méthodes suivantes :

- intégrer notre code CSS directement à notre fichier HTML dans une **balise** `<style>`, à l'intérieur de la balise `<head>` ;
- créer un fichier CSS externe et le lier à notre fichier HTML (**vivement recommandé, et méthode à préférer le reste du cours !**).

Intégration de CSS via la balise `style`

```HTML
<head>
  <style>
  p {
    color: red;
  }
  </style>
</head>
```

Écrire et intégrer des fichiers CSS

Comme pour les fichiers HTML, les fichiers CSS sont des fichiers texte avec une extension `.css`.

On peut ensuite les intégrer à notre code HTML comme ceci :

HTML

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

CSS : syntaxe de base

Pour chaque règle, on définit un **sélecteur** (ce à quoi la règle sera appliquée) et les **propriétés** à appliquer (avec leurs **valeurs**).

```CSS
sélecteur {
  propriété1: valeur1;
  propriété2: valeur2, "valeur3 à plusieurs mots";
  ...
}
```

CSS : syntaxe de base

Le **sélecteur** correspond à une balise HTML définie dans le document.

Par exemple, pour mettre **tous** les titres de niveau 1 (`h1`) en rouge, on peut utiliser la propriété `color` :

```css
h1 {
  color: red;
}
```

CSS : couleurs

Un certain nombre de couleurs sont prédéfinies en CSS, comme `red`, `blue`, `green`, etc.

Il est également possible de définir des couleurs à l'aide de leur code hexadécimal (par exemple, `#FF0000` pour le rouge).

```CSS
h1 {
  color: #FF0000;
}
```

CSS : couleurs

Vous pouvez utiliser le site https://htmlcolorcodes.com/ pour trouver le code hexadécimal d'une couleur, ou taper « color picker » sur Google.

CSS : syntaxe de base

Il est possible de **définir plusieurs sélecteurs pour une même règle** en les séparant par des **virgules**.

```CSS
h1, h2, h3 {
  color: #ffffff;
}
```

CSS : syntaxe de base

Toutes les **balises HTML de contenu** peuvent être utilisées comme sélecteurs, y compris `body`.

On peut notamment utiliser `body` comme sélecteur pour définir des styles qui s'appliqueront à l'ensemble de la page.

```css
body {
  background-color: #f0f0f0;
}
```

Élément span

Un élément HTML souvent utilisé en combinaison avec les styles CSS pour appliquer des effets particuliers à du **contenu textuel** est la balise `span`.

L'avantage de cet élément est que, contrairement aux autres éléments textuels comme `b`, `i` ou `p`, aucun style par défaut n'est appliqué à cet élément, ce qui permet de la styliser comme bon nous semble dès le départ, sans avoir à supprimer les styles déjà appliqués par défaut.

Élément span

L'utilisation de l'élément span est par exemple utile pour appliquer des styles à du texte spécifique à l'intérieur d'un paragraphe :

```html
<p>Du texte avec <span>un fond vert</span></p>
```

```css
span {
  background: green;
}
```

II. Classes et identifiants

Ciblage spécifique

La solution proposée actuellement ne permet pas de cibler des éléments en particulier.

En effet, une règle appliquée à `p` s'appliquera à **TOUS** les éléments `p` sur la page. Il n'est pas possible de choisir spécifiquement quels éléments `p` nous voulons modifier.

Classes (`class`)

Comme solution, on peut définir des **classes** dans notre fichier HTML et les utiliser dans notre fichier CSS pour cibler un ou des éléments en particulier.

Pour ce faire, on ajoute un attribut `class` à l'élément HTML, suivi du nom de la classe entre guillemets.

Dans le CSS, on indique comme sélecteur **le nom de la classe précédé d'un point final**.

Classes : exemple

```HTML
<h1 class="special">Titre spécial</h1>
```

```CSS
.special {
  color: red;
}
```

Classes : exemple avec `span`

```HTML
<p>Du texte avec <span class="highlight">un fond vert, en grand</span></p>
```

```CSS
.highlight {
  background: green;
  font-size: large;
}
```

Identifiants (`id`)

Une alternative aux classes sont les **identifiants** (`id`).

Contrairement aux classes, un identifiant ne peut être associé qu'à **un seul élément HTML** : c'est comme une empreinte unique.

Pour définir un identifiant, on ajoute un attribut `id` à l'élément HTML, suivi du nom de l'identifiant entre guillemets.

Dans le CSS, on indique comme sélecteur **le nom de l'identifiant précédé d'un croisillon** (#).

Identifiants : exemple

```HTML
<h1 id="special">Titre spécial</h1>
```

```CSS
#special {
  color: red;
}
```

Identifiants : ancre

Les identifiants peuvent également être utilisés pour créer des ancres dans une page Web, c'est-à-dire des liens qui pointent vers une section spécifique de la page.

Pour cela, on ajoute un attribut `id` à l'élément HTML que l'on veut cibler, et on crée un lien vers cet élément en utilisant le symbole # suivi du nom de l'identifiant.

Identifiants : ancre

```HTML
<h1 id="section1">Section 1</h1>
<p>Contenu de la section 1</p>
<h1 id="section2">Section 2</h1>
<p>Contenu de la section 2</p>
```

```HTML
<a href="#section1">Aller à la section 1</a>
<a href="#section2">Aller à la section 2</a>
```

III. Combinateurs et cascade

Notion de cascade

Le terme « cascading » dans « Cascading Style Sheets » (CSS) signifie deux choses :

1. quand un élément se voit appliquer un style, tous les éléments **enfants héritent** de ce style ;

Éléments enfants et parents

Par exemple, dans ce code, `span` est l'élément **enfant** de `p` (ce dernier étant l'élément **parent** de `span`).

```HTML
<p>Du texte avec <span class="highlight">un fond vert, en grand</span></p>
```

<!DOCTYPE html>

Déclaration du type de document

<html>

Élément racine du document HTML

<head>

Contient les métadonnées/informations pour le document

<title>

Titre de la page

<body>

Contient le contenu visible de la page

<p>

Un paragraphe de texte

Notion de cascade : exemple

```css
body { /* TOUS les éléments de la page seront en bleu… */
  color: blue;
}
p { /* … sauf les paragraphes */
  color: red;
}
```

Notion de cascade

Le terme « cascading » dans « Cascading Style Sheets » (CSS) signifie deux choses :

1. quand un élément se voit appliquer un style, tous les éléments **enfants héritent** de ce style ;
2. l'application des règles se fait selon le **niveau de spécificité**.

Notion de cascade : exemple

```css
h1 { /* règle générale */
  color: red;
}

h1.special { /* règle spécifique (classe) */
  color: blue;
}
```

Notion de cascade : exemple

```css
h1.special { /* règle la plus spécifique */
  color: red;
}
.special { /* règle moins spécifique que h1.special */
  color: blue;
}
```

Notion de cascade

Il est possible d'utiliser des **sélecteurs descendants** pour cibler des éléments spécifiques.

Par exemple, pour cibler les textes en gras **à l'intérieur** d'un paragraphe, on peut utiliser le sélecteur `p strong`.

```css
p strong {
  color: blue;
}
```

Notion de cascade : combinateurs

L'exemple précédent utilise un **combinateur** (une espace) pour cibler les éléments descendants.

Il existe plusieurs types de combinateurs :

- l'espace (pour les descendants),
- `>` (pour les enfants directs),
- `+` (pour les éléments adjacents).

Combinateurs : exemple

Soit le HTML suivant :

```HTML
<div>
  <p class="ma-classe">
    <a href="#" class="ma-classe">Un lien</a>
    <img class="ma-classe"/>
  </p>
  <span class="ma-classe">Un élément span</span>
</div>
<h4 class="ma-classe">Un titre</h4>
```

Que cible `div .ma-classe` ? (espace = descendants)

```html
<div>
  <p class="ma-classe">
    <a href="#" class="ma-classe">Un lien</a>
    <img class="ma-classe"/>
  </p>
  <span class="ma-classe">Un élément span</span>
</div>
<h4 class="ma-classe">Un titre</h4>
```

Que cible `div .ma-classe` ? (espace = descendants)

HTML

```
1  <div>
2    <p class="ma-classe">
3      <a href="#" class="ma-classe">Un lien</a>
4      <img class="ma-classe"/>
5    </p>
6    <span class="ma-classe">Un élément span</span>
7  </div>
8  <h4 class="ma-classe">Un titre</h4>
```

Combinateurs : exemple

`div .ma-classe` s'appliquera à tous les **descendants** de `div` ayant la classe `ma-classe`, soit : `p`, `a`, `img` et `span`.

Que cible `div > .ma-classe` ? (signe sup. = enfants directs)

```html
<div>
  <p class="ma-classe">
    <a href="#" class="ma-classe">Un lien</a>
    <img class="ma-classe"/>
  </p>
  <span class="ma-classe">Un élément span</span>
</div>
<h4 class="ma-classe">Un titre</h4>
```

Que cible `div > .ma-classe` ? (signe sup. = enfants directs)

```HTML
<div>
  <p class="ma-classe">
    <a href="#" class="ma-classe">Un lien</a>
    <img class="ma-classe"/>
  </p>
  <span class="ma-classe">Un élément span</span>
</div>
<h4 class="ma-classe">Un titre</h4>
```

Combinateurs : exemple

`div .ma-classe` s'appliquera à tous les **descendants** de `div` ayant la classe `ma-classe`, soit : `p`, `a`, `img` et `span`.

`div > .ma-classe` s'appliquera uniquement aux éléments `p` et `span`, car ce sont les **seuls enfants directs** de `div`.

Que cible `div + .ma-classe` ? (signe plus = adjacents)

HTML

```html
<div>
  <p class="ma-classe">
    <a href="#" class="ma-classe">Un lien</a>
    <img class="ma-classe"/>
  </p>
  <span class="ma-classe">Un élément span</span>
</div>
<h4 class="ma-classe">Un titre</h4>
```

Que cible `div + .ma-classe` ? (signe plus = adjacents)

HTML

```
1 <div>
2   <p class="ma-classe">
3     <a href="#" class="ma-classe">Un lien</a>
4     <img class="ma-classe"/>
5   </p>
6   <span class="ma-classe">Un élément span</span>
7 </div>
8 <h4 class="ma-classe">Un titre</h4>
```

Combinateurs : exemple

`div .ma-classe` s'appliquera à tous les descendants de `div` ayant la classe `ma-classe`, soit : `p`, `a`, `img` et `span`.

`div > .ma-classe` s'appliquera uniquement aux éléments `p` et `span`, car ce sont les **seuls enfants directs** de `div`.

`div + .ma-classe` s'appliquera uniquement à l'élément `h4`, car c'est le **seul élément adjacent** à `div`.

Notion de cascade : priorité

La priorité des règles appliquées est déterminée par leur **spécificité**.

La hiérarchie de spécificité est la suivante :

1. style dans les balises HTML (le plus spécifique)
2. `id`,
3. `class`,
4. balises (p. ex. `p`, `h1`, etc.).

Notion de cascade : priorité et règle `!important`

Il est possible de définir la priorité d'une règle à son maximum en ajoutant `!important` à la fin de la valeur de la propriété.

```css
h1 {
  color: red !important;
}
```

L'utilisation de `!important` est toutefois **peu recommandée** et doit être réservée à des **cas exceptionnels**.

Pseudo-classes

Les **pseudo-classes** permettent de cibler des éléments HTML dans un état particulier. Elles sont spécifiques à certains types d'éléments.

Une pseudo-classe est ajoutée à un sélecteur en utilisant le symbole deux-points `:`, suivi du nom de la pseudo-classe.

Pseudo-classes : exemple

Avec la règle suivante, les liens deviennent rouges quand on passe la souris dessus :

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

Pseudo-classes

Il existe de nombreuses pseudo-classes,

comme `:hover`, `:active`, `:focus`, `:first-child`, etc., mais elles ne sont pas applicables à tous les éléments.

Liste des pseudo-classes sur MDN