GALIMATECH
L'EXPERIENCE DIGITALE

Apprendre le CSS

***

Plan du cours

- Introduction
- La syntaxe
- Comment utiliser CSS dans un fichier HTML
- Formatage du texte
- Les couleurs
- Les bordures
- Le Background
- La propriété display
- Les unités de mesures
- Les dimensions d’un élément
- Les Marges
- Le positionnement
- Le Float
- Le display grid
- Le flexbox
- Les variables

CSS est l'acronyme de Cascading Style Sheets. C’est un langage qui permet de manipuler l’affichage d’un document HTML. C’est un langage facile à appréhender et il constitue avec le HTML, l’un des langages de base de la programmation Web.

La syntax

L'utilisation du CSS se fait en deux étapes. D'abord on sélectionne le ou les éléments à éditer, puis on ajoute les propriétés css.

Pour sélectionner un élément nous allons utiliser les sélecteurs comme suite: selecteur{}.

Les sélecteurs en css sont soit le **nom de la balise**, l'attribut **class** d'une balise ou l'attribut **id** d'une balise. Le nom d'une balise HTML est le texte entre les signes < et >. La class et l'id sont respectivement, les valeurs des attributs class="" et id="" d'une balise.

L'attribut class peut prendre plusieurs valeurs comme suite: class="valeur1 valeur2 valeur3 ...". L'attribut id quant à lui, ne prend qu'une seule valeur qui doit être unique dans tout le document HTML. On ne doit pas avoir des éléments avec des attributs id identiques dans un même document HTML.

Après la sélection de l'élément nous pouvons utiliser les propriétés CSS pour le modifier. Pour définir une propriété nous utilisons la syntaxe suivante: propriete: valeur;

La syntaxe complète en CSS est donc: selecteur{propriete:valeur; propriete: valeur; propriete: valeur;}. Notez que chaque propriété se termine avec un ';'.

Comment utiliser CSS dans un fichier HTML

Il existe trois méthodes pour utiliser du CSS dans un document HTML:

- l'intégration dans la balise HTML
- Feuille de style interne
- Feuille de style externe

❖ L'intégration dans la balise (aussi appelée inline style) se fait avec l'attribut style. C'est une technique à éviter car le but premier du CSS est de dissocier le contenu de la mise en forme.

```
<p style="propriete: valeur; propriete:valeur ...">
```

❖ Nous pouvons écrire du css dans un document HTML en utilisant la balise <style> dans la balise <head> du document.

```
<style type="text/css">
 selecteur{
        propriete: valeur;
        propriete: valeur;
        }
</style>
```

Le css défini avec cette méthode ne va s'appliquer que sur le document ou il est présent.

❖ La feuille de style externe est la méthode préférée pour utiliser du CSS. Elle consiste à créer un fichier avec l'extension .css. Et à l'intégrer dans le document HTML comme suite:

```
<link rel="stylesheet" type="text/css"
href="chemin/vers/fichier.css"/>
```

Formatage du texte

Maintenant que nous pouvons intégrer du CSS dans un document HTML, nous pouvons commencer à faire nos premières transformations.

Le CSS offre enormements de propriétés pour formater du texte.

- **Color:** permet de définir la couleur du texte: `color: red`
- **font-size:** cette propriété permet de définir la taille du texte.
- **font-weight:** cette propriété permet de définir le gras du text. Elle peut prendre les valeurs bold, light, lighter ou des valeurs numérique suivant 100,200,300,400,500,600,700, 800 ou 900.
- **font-family:** va définir la police à utiliser. Elle prend plusieur valeurs séparées par des virgules et doit se terminer par une type de police generic. Il existe cinq type generics: **serif**, **sans-serif**, **monospace**,**fantasy** **cursive**.

  `font-family: Helvetica, Arial, sans-serif`

- **line-height:** permet de définir la hauteur des lignes. Elle peut prendre des pixels comme valeur, mais il est plus commun d'utiliser des valeurs entre 1 et 2: `font-size: 1.4;`
- **letter-spacing:** va permettre de manipuler les espaces entre les lettres. letter-spacing: 3px;
- **text-decoration:** cette propriété va appliquer plus transformation en fonction de sa valeur:
  - underline: va souligner l'élémen
  - overline: permet d'afficher une ligne au dessus du text
  - line-through: va barrer le texte
- **text-transform:** est souvent utilisé pour d'avantage de transformation. Elle prend les valeurs suivantes:
  - uppercase qui va mettre toute les lettres en majuscule.
  - lowercase va transformer toutes les lettres en minuscule.
  - capitalize va mettre la première lettre de chaque mot en majuscule.

Formatage du texte: police personnalisée

La propriété **font-family** utilise par défaut les polices du système d'exploitation. Mais il peut aussi utiliser des polices customs. Il s'agit là de polices qui ne sont pas intégrées dans les systèmes d'exploitation, mais qui appartiennent à des services tiers.
En guise d'exemple, nous pouvons utiliser Google Font et nous allons importer la police Long Cang.
Pour cela nous pouvons soit importer la police dans le document HTML ou l'importer dans le CSS.
Pour importer une police dans le HTML nous allons ajouter une la balise **link** avec l'URLl de la police:
`<link href="https://fonts.googleapis.com/css?family=Long+Cang&display=swap" rel="stylesheet">`.
Cette méthode est la méthode recommandée.

Pour importer une police dans un fichier CSS il faut utiliser la propriété `@import url()`. Cette propriété doit être placée au tout début du fichier. `@import url('https://fonts.googleapis.com/css?family=Long+Cang&display=swap');`

Une fois la police importée, nous pouvons alors l'utiliser pour formater le texte.

```
font-family: 'Long Cang', sans-serif;
```

Les couleurs

Nous avons déjà eu un aperçu de l'utilisation des couleurs avec la propriété ‘color’ avec une valeur ‘red’ pour éditer du texte. Cette méthode est la plus facile mais certainement pas la plus exhaustive.

Le CSS offre d’autres méthodes pour définir la couleur d’un élément.

On peut utiliser l'écriture hexadécimale. Cette écriture utilise un # suivi de 6 caractères. Les deux premiers caractères représentent la couleur rouge. Les deux suivants la couleur verte. Et les deux derniers la bleue. Ces caractères peuvent prendre les valeurs numériques allant du 0 au 9. Ou les lettres allant de la lettre ‘a’ jusqu'à la lettre ‘f’

```
h1{color: #af8dc4;}
H2 {color: #ff0000}
```

Nous pouvons aussi utiliser la méthode décomposée. Elle se compose de l’attribut rgb() avec trois valeurs numériques allant de 0 à 255. Ces valeurs représentent les couleurs rouge vert et bleu.

```
p{color: rgb(255,0,143);}
a{color: rgb(9,32,120);}
```

Une variante de la fonction rgb est la fonction rgba. Celle-ci va ajouter un quatrième paramètre qui va définir l'opacité de la couleur. Ce paramètre est compris entre 0 et 1. La fonction rgba() est le plus souvent utilisée pour définir un arrière-plan.

```
div{background: rgba(2,43,21,0.94);}
```

Les bordures

Pour ajouter une bordure à un élément nous allons utiliser la propriété bordée.

border: 1px solid #aa0021;
Le premier paramètre définit la largeur de la bordure.
Le second paramètre définit le style de la bordure. Il peut être égale à **solid**, **dotted**, **dashed**, **hidden**, **double**, **groove**, **ridge**, **inset** et **outset** . Le lien suivant donne une présentation sur l'effet de chaque option: https://developer.mozilla.org/fr/docs/Web/CSS/border-style.
Le dernier paramètre est la couleur de la bordure.

Les paramètres de la propriété 'border' sont un raccourci de trois propriétés:
border-width qui définit la largeur des bordures
border-style: qui definit le style
border-color: qui définit la couleur.
La propriété border définit les bordures sur les quatre côtés, mais il est aussi possible d'utiliser des propriétés qui ne vont cibler qu'un côté de l'élément

border-top: pour la bordure du haut
border-bottom: pour la bordure du bas
border-left: pour la bordure à gauche
border-right: pour la bordure à droite

Nous pouvons aussi préciser l'arrondie des angles des bordures avec la propriété **border-radius.** Cette propriété peut prendre une à quatre valeurs

- border-radius: 12px définit l'arrondi sur les quatre cotés à 12
- border-radius: 3px 4px ; définit les arrondies en haut à gauche et en bas à droite à 3px et les arrondies en haut à droite et en bas à gauche à 4px
- border-radius: 3px 6px 4px; définit l'arrondie du haut à gauche à 3px; l'arrondit du bas à droite à 4px et les deux restants à 6px.
- border-radius: 2px 3px 4px 5px; définit la bordure en haut à gauche à 2px, la bordure en haut à droite à 3px, la bordure en bas à droite à 4px et la bordure en bas à gauche à 5px.

La propriété background

La propriété background permet de définir l'arrière-plan d'un élément.
background: red; va définir un fond rouge à l'élément
La propriété background est un raccourci de plein d'autres attributs qui peuvent être utilisés pour manipuler le fond d'un élément.

- **background-color** permet de définir la couleur de fond.
- **background-image** permet d'utiliser une image en bg. Elle s'utilise avec la fonction url() comme suite: url('chemin/vers/limage.jpg').
- **bg-repeat** gère la répétition de l'image en arrière-plan. Elle peut prendre les valeurs suivantes:
  - **repeat-x** pour une répétition en horizontale
  - **repeat-y** pour une répétition en vertical,
  - **repeat** est la valeur par défaut. Elle va répéter l'image partout,
  - **no-repeat** va désactiver la répétition.]
- **backgroundg-position** détermine la position de l'image en arrière-plan. Elle peut prendre les valeurs **top**, **left**, **right**, **bottom** ou **center**. On peut aussi utiliser des valeurs en pourcentage. Elle peut prendre deux valeurs pour préciser la position en x et en y.
- **background-size** va définir la taille de l'image en arrière-plan en fonction de son conteneur. Elle prend les valeurs:
  - **contain**: l'image sera redimensionnée pour pouvoir tenir dans le conteneur.
  - **cover**: l'image va prendre une taille qui va recouvrir la taille du conteneur
  - On peut aussi utiliser des pourcentages pour définir les dimensions de l'image. Dans ce cas les valeurs en pourcentages seront relatives au conteneur.
- **background-attachment**: définit le comportement du background lors du scroll. Il peut être en **scroll** qui est la valeur par défaut; l'arrière-plan va donc défiler avec le contenu. Ou peut aussi prendre la valeur **fixed** qui va figer l'image lors du scroll.

La propriété display

La propriété **display** permet de définir le mode d'affichage que doit adopter un élément. Elle précise sa position par rapport au contenu et gère aussi le comportement de ses enfants.
Les modes les plus utilisés sont **none**, **block**, **inline**, **inline-block** et **flex**:

- un element en display none ne sera plus visible sur la page web.
- un élément en display block prend toute la largeur de la page. Deux éléments avec ce mode ne peuvent pas être affichés côte à côte.
- Un élément en inline ne va prendre que l'espace nécessaire pour l'affichage de son contenu. Élément en inline ne peut pas être dimensionné.
- inline-block va faire la combinaison du mode block et du mode inline. C'est-à-dire, qu'il va permettre à un élément d'être dimensionné et va pouvoir s'afficher à côté d'un autre élément si l'espace restant le permet.
- Le display flex est un mode plus poussé que nous verrons plus tard.

La propriete diplay peut prendre d'autres valeurs que nous pouvons retrouver sur cette page https://developer.mozilla.org/fr/docs/Web/CSS/display.

les unités de mesures

Jusque-là nous n'avons utilisé que le pixel comme unité de mesure. Mais le CSS offre plus d'options pour définir les dimensions des éléments.

- **px** est équivalent à la taille du pixel d'un écran.
- **Le %** définit des valeurs en pourcentage pour un élément. Les pourcentages appliqués sont relatifs au parent de l'élément.
- **vw** cette unité va dépendre de la largeur du viewport. Le viewport est l'espace visible sans scroller.
- **vh** va dépendre de la hauteur du viewport.
- **vmin** va dépendre de la valeur la plus petite entre la largeur et la hauteur du viewport.
- **vmax** va déprendre de la valeur la plus grande entre la largeur et la hauteur du viewport.
- **em** cette unité dépend de la taille de la police qui est appliquée sur l'élément. C'est une unité qui peut prendre des valeurs décimales.
- **rem** est comme le **em** sauf qu'elle dépend de la police appliquée sur les éléments racines tels que body ou html. Cette unité permet d'avoir une référence commune pour définir les tailles et dimensions des éléments.

Plus de détails sont disponibles sur cette page: https://www.w3.org/Style/Examples/007/units.fr.html

Les dimensions d’un element

Pour définir les dimensions d'un élément avec du CSS nous allons utiliser les propriétés width et height.

width: va préciser la largeur d'un élément.
height: va définir la hauteur de l'élément.

Nous pouvons utiliser n'importe lesquelles des unités de mesures vues précédemment pour définir la valeur de ces propriétés.
Nous pouvons aussi utiliser les préfixes **max-** et **min-** devant chacune de ces propriétés pour définir respectivement la valeur maximale ou minimale.
Ainsi max-width va définir la largeur maximale que peut prendre un élément et min-width sa largeur minimale.
max-height va définir la hauteur maximale que peut prendre un element et min-height sa hauteur minimale.

Il faut noter que si un élément a un mode de display en inline il ne pourra pas être dimensionné. Les propriétés présentées ici n’auront alors aucun effet.

La propriété position

La propriété position permet de définir comment un élément doit être positionné dans le navigateur ou vis à vis de son parent. Elle prend les valeurs suivantes:

- **static**: c'est la valeur par défaut. Les éléments dans cet état sont positionnés en fonction du contenu.
- **absolute**: cet élément va s'afficher de manière absolue par rapport au navigateur ou par rapport à son parent. Il va alors avoir un comportement flottant. Il faudra ensuite utiliser les propriétés **top**, **right**, **bottom** et **left** pour positionner l'élément par rapport au haut, à droite, en bas ou à gauche de la fenêtre.
- **fixed** va figer l'élément. Elle se réfère à la fenêtre et pas au parent de l'élément.
- **sticky**: est une combinaison des positions **fixed** et **static**. L'élément avec cette propriété va avoir une position générique dans un premier temps. Puis au moment du scroll elle va se comporter comme un élément en position fixed.

relative: se comporte comme le static mais permet en plus l'utilisation des propriétés top, right, bottom, left. Cette propriété est le plus souvent utilisée sur le parent des éléments avec une position absolute. Un élément en position absolute se réfère par défaut à la fenêtre. Pour le forcer à se référer à son élément parent, nous allons utiliser la position **relative** sur ce dernier.

L'utilisation de la propriété position s'accompagne souvent de la propriété **z-index**. **z-index** permet de définir comment les éléments vont se superposer. Plus le z-index d'un élément est grand, plus il a de chance de se placer au-dessus des autres. Le z-index prend des valeurs numériques sans unité de mesure:

```
element{
        position: absolute;
        z-index: 10
}
```

Le Float

La propriété float permet de positionner un élément et force les autres éléments à se positionner au tour de l'élément en float.
Elle prend en valeur les paramètres suivants:

- left pour placer l'élément à gauche,
- right pour que l'élément soit à droite
- none pour désactiver le float.

Lorsqu'on ajoute la propriété float à un élément inline, celui-ci devient alors un élément block.

Travailler avec un float n'est pas aisé. Il est presque impossible de toujours savoir comment il va se comporter. Et vu que l'élément en float a un effet sur les éléments alentour, il arrive souvent que l'effet du float aille au-delà de l'espace souhaité.
Pour limiter l'effet d'un float, nous allons utiliser la propriété clear. Cette propriété prend quatre valeurs:

- left pour desactiver les float a gauche
- right pour desactiver les floats a droite
- both pour désactiver dans les deux sens
- none va supprimer les effets du clear.

Le display grid

Le display grid fait partie des paramètres de la propriété display. Il permet de positionner des éléments les uns à côté des autres comme dans une grille. Sa particularité est quand on met un élément en mode display: grid, on peut alors utiliser d'autres propriétés avec le mot clé grid, pour manipuler la position de ses enfants. Comme propriété grid nous avons:

- **grid-template-columns** qui va définir le nombre de colonnes et leurs tailles. On peut préciser plusieurs valeurs et chaque valeur sera une colonne ou utiliser la fonction repeat(nbr_colonne, taille_colonne).
- **grid-template-rows** définit les nombres de lignes. Elle définit ses valeurs de la même manière que la propriété grid-template-columns.
- **grid-column-gap:** va définir les espaces entre les éléments. En définissant cette valeur il faut prendre en compte le fait que la valeur de cet attribut va s'ajouter à la largeur des grilles et que les éléments peuvent alors déborder.
- **grid-auto-flow** permet de changer le comportement de remplissage des éléments. Par défaut le grid a un comportement de remplissage en horizontal. Cela veut dire qu'il va remplir les éléments ligne par ligne. On peut changer ce comportement avec la propriété grid-auto-flow. Elle prend les valeurs row et column. La valeur row est celle par défaut. La valeur column va entrer les éléments par colonne.

Toutes ces propriétés vont s'appliquer à l'élément parent. Mais il est aussi possible d'agir sur les éléments enfants.

- **grid-row-start** et **grid-row-end** vont définir la ligne de début du child et la ligne où il doit s'arrêter.
- **grid-column-start** et **grid-column-end** vont définir la colonne de début du child et la colonne de fin.

Les propriétés peuvent être très utiles pour définir le layout d'un site web. Pour en apprendre plus je vous invite à visiter le MDN: https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Grid_Layout

Le flexbox

Les flexbox permettent à partir d'un élément parent de manipuler les affichages des éléments enfants. Ils sont définis avec la propriété display

```
element {
  display: flex;
}
```

Une fois le mode flex applique au parent, on peut alors utiliser d'autres propriétés pour impacter l'affichage des éléments enfants.

flex-direction va définir la direction dans laquelle il va arranger les éléments. Il est égal à column ou row. Sa valeur column va arranger les éléments par colonne. Sa valeur row est la valeur par défaut, elle va arranger les éléments par ligne.

flex-wrap: va définir le comportement des enfants dans le cas ou il n'y a pas assez de place. Il prend les valeurs:

- nowrap: c'est la valeur par défaut. Elle va s'arranger pour que tous les éléments enfant tiennent sur une ligne.
- wrap: va forcer le retour à la ligne pour afficher les éléments restants
- wrap-reverse: force le retour à la ligne en inversant l'ordre des éléments.

justify-content: permet d'aligner horizontalement les éléments enfants. Elle prend les valeurs suivantes:

flex-start: va aligner tous les éléments vers la gauche

flex-end va aligner tous les éléments vers la droite

center: pour centrer les éléments

space-around: va ajouter un espace égal à la gauche et la droite de chaque élément

space-between: va ajouter un espace entre les éléments et supprimer les espaces au niveau des extrémités

align-items: va appliquer un alignement vertical des éléments. Par défaut elle a la valeur stretch qui va attirer les éléments pour qu'ils remplissent tout l'espace de l'élément parent. Elle peut aussi prendre la valeur **center** qui va garder la taille d'origine des éléments enfants en les centrant verticalement. Ou les valeurs flex-start et flex-end qui vont placer les éléments enfants respectivement au début et à la fin de l'élément parent.

Je vous invite à visiter ce lien https://flexbox.help/ pour mieux comprendre les propriétés flexbox

Le flexbox: flex items

En plus des propriétés flexbox que nous avons vues, qui s'appliquent sur l'élément parent, nous pouvons utiliser des propriétés qui elles vont s'utiliser sur les éléments enfants.

flex: cette propriété s'applique directement sur l'enfant.Elle permet de définir la proportion que peut prendre un élément.

```
element-enfant1{
        flex: 2;
}
```

Sur l'exemple ci-dessus, element-enfant1 aura une taille deux fois supérieure aux autres éléments enfants.

order: cette propriété aussi s'applique aux éléments enfant. Elle permet de manipuler l'ordre d'affichage.

element-enfant1{order: 1}

Cet exemple va placer l'élément element-enfant1 en dernière position.

Par défaut les éléments enfant ont une valeur order égale à 0. Plus un élément a une valeur order élevé, plus il sera proche de la dernière position.

align-self: cette propriété permet à un élément enfant d'avoir un alignement différent du groupe.

```
element-enfant1{ align-self: center}
```

Elle prend les valeurs:

flex-start: va placer l'élément sélectionné en début de liste
flex-end: va placer l'élément sélectionné en début de liste
center: va centrer l'élément sélectionné.
baseline: va aligner l'élément en fonction de la ligne de base.
stretch: va étirer l'élément pour prendre tout l'espace restant.

Nous pouvons visiter la page suivante https://the-echoplex.net/flexyboxes/ pour mieux nous familiariser avec ces propriétés.

Les variables

Les variables permettent de garder en mémoire une valeur que l'on pourra réutiliser à volonté. Elles vont permettre d'éviter le plus possible les répétitions des valeurs dans un fichier CSS.
Pour initialiser une valeur nous allons utiliser la syntaxe suivante:

```
--nom_variable: valeur;
```

Cette initialisation se fait dans un élément. Une norme veut que les variables soient définies dans un élément :root.

```
:root{
        --couleur: #deaa00;
        --hauteur: 80vh;
        border: 1px solid #001111;
}
```

Pour utiliser ces variables nous ferons

```
selector{
        background-color: var(--couleur);
        border: var(--border) ;
        Height: var(--hauteur);
}
```

Nous pouvons en apprendre plus sur cette page
https://developer.mozilla.org/fr/docs/Web/CSS/Using_CSS_custom_properties

Conclusion

Le HTML et CSS constituent la base de la presque totalités des site Web sur internet.

Nous pouvons donc dire que nous avons le bagage nécessaire pour concevoir un site.

Cependant la prochaine étape de notre formation, qui est le JavaScript, va nous permettre d'ajouter davantage de manipulations sur un site Web.

Liens utiles

GALIMATECH
L'EXPERIENCE DIGITALE

- https://developer.mozilla.org/en-US/docs/Web/CSS
- https://developer.mozilla.org/fr/docs/Web/CSS/display
- https://www.w3.org/Style/Examples/007/units.fr.html
- https://developer.mozilla.org/fr/docs/Web/CSS/CSS_Grid_Layout
- https://flexbox.help/
- https://the-echoplex.net/flexyboxes/
- https://developer.mozilla.org/fr/docs/Web/CSS/Using_CSS_custom_properties