

# HTML / CSS

## Balises vues dans les exemples

La plus part des balises fonctionnent par couple, une ouvrant le bloc affecté par la balise, l'autre le refermant, certaines fonctionnent en solo.

### Balises importantes :

Pour faire un lien

`<a href="adresse cible" target="blank_">` Chaîne de caractères réactive(une fois cliquée elle vous emmènera vers le lien) `</a>`

L'ajout du paramètre permet de faire que le lien ne soit pas ouvert dans l'onglet actif mais dans un nouveau.

`<h1>`, `<h2>`, `<h3>` Sont des balises de titres et sous titres, plus le nombre est grand plus le titre sera petit.

```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/d4qRJTjvqvs" frameborder="0"
allow="accelerometer; autoplay; encrypted-media; gyroscope;
picture-in-picture" allowfullscreen></iframe>
```

Ça c'est une balise permettant l'intégration d'une video youtube, mais les iframe permettent en fait l'intégration de tout et n'importe quoi : généralement ce sont des pages ou des morceaux de pages venant d'un autre site, les paramètres permettent de configurer l'intégration.

`<p>`bloc affecté par la balise paragraphe `</p>`

`<br/>` balise solo qui provoque un retour à la ligne

`` Balise solo qui servira à inclure une image, ici plus particulièrement l'image s'appelant machin.jpg se trouvant dans le répertoire pic, qui sera affichée avec une hauteur et une largeur valant 50% des dimensions réelles de l'image.

`<!--`balise solo commentaire visible pour le programmeur ignorée par le navigateur`>`

`<center>` Permet de centrer horizontalement sur la ou les lignes le texte du bloc.

`<strong>`, `<big>`, `<em>` Permettent respectivement d'écrire un texte en gras, en grand, en italique

Pour faire une liste, on utilisera les balises `<ul>` et `<li>`, la balise `<ul>` pour délimiter la globalité de la liste et une balise `<li>` pour chaque item. Par exemple pour faire une liste de liens internes je peux écrire :

```
<ul>
  <li><a href="#musiques">leurs musiques</a></li>
  <li><a href="#infos">leurs vidéos</a></li>
</ul>
```

(Crédit : Clément Cauquil)

### Structure de base

C'est la structure/norme que l'on retrouvera dans la plus part des fichiers HTML, elle est nécessaire pour que la page puisse être homologuée (on peut toutefois se passer de certaines balises sans que ça empêche le navigateur de l'interpréter correctement)

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

bloc contenant les paramètres de la page

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

norme des caractères utilisés dans la page

```
<meta name="viewport" content="width=device-width">
```

```
<title> titre de la page </title>
```

(visible sur l'onglet du navigateur)

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

indique où l'on pourra trouver la feuille de style

```
</head>
<body>
  <script src="script.js"></script>
  C'est là qu'on mettra l'essentiel de notre page
</body>
</html>
```

### **Pièges de l'HTML**

Il ne reconnaît pas les retours à la ligne du code, vous serez obligé d'utiliser la balise

Il ne reconnaît pas non plus les espaces répétés :

Si vous écrivez « a                      a » dans votre code HTML il le traduira par « a a », pour accumuler les espaces il faudra utiliser &nbsp; pour chacun d'entre eux.

si c'est pour faire une indentation de début de paragraphe vous pouvez aussi utiliser une balise et mettre comme style : indent-text: 5em ; ce qui veut dire décalage de la largeur de 5 caractères.

Il ne reconnaît pas les caractères < et > , il croit que vous voulez écrire des balises, donc pour pouvoir les matérialiser sur votre pages web il faudra utiliser leur code HTML : &lt; (lesser than) et &gt; (greater than)

pour  $\leq$  et pour  $\geq$  ça sera &le; (lesser or equal) et &ge; (greater or equal)

## CSS (bases)

### Effet de style :

On peut localement ajouter dans une balise (par exemple la balise neutre `<div>`) le paramètre `style` suivi d'une liste d'indications se terminant par des points virgules et globalement encadrée par des guillemets, par exemple :  
`<p id="infos" style="text-indent: 15px;">`

(en plus d'avoir un paramètre `style` on a rajouté un paramètre d'identification (unique) `id="infos"`)

Quand on veut créer une norme de présentation pour un document, par exemple faire que tous les paragraphes aient la même forme, que les titres soient d'une couleur particulière etc on utilisera une feuille de style CSS, où l'on consignera tous ces paramètres. D'ailleurs cette feuille de style pourra être réemployée par la suite pour les pages à venir ce qui permettra d'avoir une présentation unique pour tout un site, et le jour où l'on veut amender notre présentation il suffira d'agir sur la feuille pour que tout le site soit changé.

Dans la feuille de style si on a envie que tous les titres h1 soient écrits en vert avec un police de taille 62pixelx et entourés en fuchsia d'un trait épais de deux pixels, alors on pourra écrire dans la feuille de style

```
h1
{
  border: 2px rgb(255, 0, 106) solid;
  color: rgb(38, 179, 10);
  font-size: 62px;
}
```

On peut aussi agir sur des balises ayant une classe ou une id donnée :

```
.head { background-color: #13214a;}
```

Permet d'agir sur toutes les balises de la classe « head » et de mettre leur contenu avec un fond de couleur dont le code hexadécimal est `#13214a`, c'est-à-dire `#13= 1*16+3=19` en rouge, `#21=2*16+1=33` en vert et `#4a= 4*16+10*1=74` en bleu.

Remarque : Si on a envie un bloc peut être multi-classes, en HTML il suffit d'espacer les noms de classes lors de la déclaration : `<div class="classe1 classe2"> bla bla </div>`

On peut aussi associer à un bloc unique certaines propriétés, le plus simple est alors de lui donner une **identité unique** genre `<div id="infos">` et puis dans le code CSS d'associer à cette identité des paramètres : `#infos { text-indent: 40px; color:gray;}` qui fera commencer le texte écrit en gris avec une indentation de 40 pixels.

## CSS avancé

(Analyse de la proposition d'Antoine Collard )

<https://repl.it/@jkergot/Antoine-Collard-cool-CSS-page#index.html>

```
8 | <link href="https://fonts.googleapis.com/css2?
  | family=Poppins:wght@300&display=swap" rel="stylesheet">
9 | </head>
10 | <body>
11 | <div class="header">
12 |   <a href="/">Accueil</a>
13 |   <a href="#forum">Forum</a>
14 |   <a href="#infos">Infos</a>
15 | </div>
16 | <div class="parallax p1"></div>
17 | <div id="forum" align="center" class="container">
18 |   <h3>Non</h3>
19 | </div>
20 | <div class="parallax"></div>
21 | <div id="infos" align="center" class="container">
22 |   <h3>Oui</h3>
23 | </div>
24 | <div class="parallax p2"></div>
25 | <div class="footer" align="center">
26 |   © Copyright personne | Tous droits réservés
27 | </div>
28 | </body>
29 | </html>
```

Ligne 8 on importe la police de caractère Poppins pour qu'elle soit utilisable quand on l'appellera dans le fichier CSS (ligne 1 de code sur la partie de droite).

Lignes 12, 13 et 14 ce sont des liens internes qui nous permettent de regarder la page à partir de la balise portant le nom proposé.

En dehors du parallaxe le code CSS suivant permet de voir comment jouer avec la couleur d'une image.

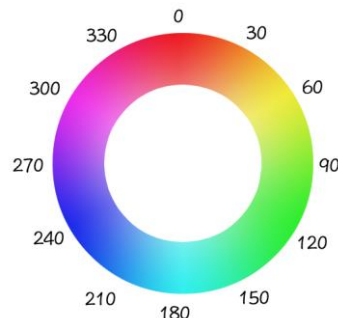
```
1 | *{font-family: 'Poppins', sans-serif; margin: 0;}
2 | html { scroll-behavior: smooth; }
3 | .header{
4 |   display: flex;
5 |   flex-direction: row;
6 |   justify-content: space-around;
7 |   background-color: #262626;
8 |   padding: 20px;}
9 | .header a{
10 |   text-decoration: none;
11 |   color: white;}
12 | .parallax{
13 |   background-image: url
14 |   ('https://c4.wallpaperflare.com/wallpaper/712/358/421/earth-space-es
15 |   pace-planet-wallpaper-preview.jpg');
16 |   height: 450px;
17 |   background-attachment: fixed;
18 |   background-position: center;
19 |   background-repeat: no-repeat;
20 |   background-size: cover;
21 | }
22 | .p1{ filter: hue-rotate(180deg); }
23 | .p2{ animation: teinte 4s infinite;}
24 | @keyframes teinte{
25 |   from{filter: hue-rotate(0);}
26 |   to{filter: hue-rotate(360deg);}
27 | }
28 | .container, .footer{
29 |   background-color: #262626;
30 |   padding: 20px 0;
31 |   width: 100%;
32 |   color: white;
33 | }
```

Ligne 21, pour la classe p2 on fait une animation nommée teinte qui dure 4 seconde et qui tournera à l'infini. Cette animation est définie entre les lignes 23 et 26 avec un point de départ et un point d'arrivée.

Lors de la définition de la classe p1 comme pour le point de départ et d'arrivée de l'animation on voit apparaître :

```
filter: hue-rotate(angle en degré);
```

ça correspond à un filtre qui transforme toute couleur de la roue chromatique en une autre faisant avec la première l'angle proposé en paramètre.



Par exemple dans p1 l'angle proposé est 180° donc le rouge est transformé en bleu ciel, le vert en violet etc.

En p2 on aura donc un tour de roue chromatique en 4s, qui sera répété encore et encore.

Pour en savoir plus sur les animations je vous invite à consulter la page : [https://www.w3schools.com/css/css3\\_animations.asp](https://www.w3schools.com/css/css3_animations.asp), qui est riche en exemples.

Autres points intéressants du code d'Antoine :

`scroll-behavior: smooth;` à la ligne 2 du code CSS permet d'avoir une transition fluide quand je vais passer d'un bloc à l'autre à l'aide de des liens de mon menu. Jolie animation pour voir la différence entre le mode smooth et le mode auto : <https://developer.mozilla.org/fr/docs/Web/CSS/scroll-behavior>

\* l'utilisation de l'étoile à la ligne 1, permet d'appliquer la propriété (choix de la police de caractère poppins) à tout le document.

Ligne 27 quand on veut associer un certains nombres de paramètres à des balises de classes différentes, on peut lister les classes avant de donner les propriétés.

Ligne 9 : permet de donner des caractéristiques pour un bloc qui cumule les appartenances, ici tout ce qui est à la fois à la fois dans une balise de classe header et dans une balise a aura les propriétés données entre les lignes 10 et 11.

### Bonus :

Pour afficher une vidéos dont la taille sera interactive : elle prendra 100% de la largeur de la page on peut mettre le lien proposé par youtube, virer les indicateurs de dimension lui associer la classe **vidéo** et mettre tout ça dans une balise de classe **container** du moment que dans le CSS on met :

```
.container {
    position: relative;
    width: 100%; height: 0;
    padding-bottom: 56.25%; }
.vidéo {
    position: absolute;
    top: 0; left: 0;
    width: 100%; height: 100%; }
```

sur la partie HTML ça donnerait :

```
<div class="container">
    <iframe src="adresseDeLaVideoYoutube" frameborder="0"
allowfullscreen class="video"></iframe>
</div>
```

## CSS avancé 2

(Analyse de la proposition d'Antoine Collard )

<https://repl.it/@jkergot/Antoine-Collard-Cool-CSS-page-2#index.html>

```
6 <title>Antoine Cool Css Page 2</title>
7 <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300&display=swap" rel="stylesheet">
8 <link href="style.css" rel="stylesheet" type="text/css" />
9 </head>
10 <body>
11 <div class="container" align="center">
12 <li class="btn">
13 <a class="a" href="">
14 | Bouton
15 </a>
16 </li>
17 </div>
18 </body>
19 </html>

1 *{margin: 0;font-family: 'Poppins', sans-serif;}
2 .container{
3   background-color: #111;
4   height: 100vh;
5   line-height:100vh;}
6 /*First Anim*/
7 .btn{width: fit-content;overflow: hidden; list-style-type:none;}
8 .btn .a{
9   padding: 5px;
10  font-size: 25px;
11  position: relative;
12  background: #ffdd32;
13  background: -webkit-linear-gradient(90deg , #ffdd32 0%,#ffdd32 50%,#ffdd32 50%,#ffa332 51%,#fc9732 :
14  background: linear-gradient(90deg , #ffdd32 0%,#ffdd32 50%,#ffdd32 50%,#ffa332 51%,#fc9732 100%);
15  -webkit-background-clip: text;
16  -webkit-text-fill-color: transparent;
17  animation: 10s bganim linear infinite;
18  transition: .3s ease;}
19 .btn .a::after, .btn .a::before{
20   content:'';
21   position: absolute;
22   width: 100%;
23   height: 2px;
24   background: #ffdd32;
25   transition: .5s
26 }
27 .btn .a::after{
28   top: 0;
29   left: 100%}
30 .btn .a::before{
31   bottom: 0;
32   left: -100%}
33 }
34 .btn .a:hover::before, .btn .a:hover::after{
35   left: 0;
36 }
37 @keyframes bganim {
38   100% { background-position: 100vw 0px; }
39 }
```

Commençons par le commencement

La classe container qui va contenir tout ce qui est visible sur l'écran a deux caractéristiques de hauteur utilisant l'unité vh (qui correspond à 1% de la hauteur de la fenêtre dans laquelle on travaille).

Ainsi le bloc prend toute la hauteur de la fenêtre et 1 ligne de texte espaces supérieurs et inférieurs inclus fera 100% de la hauteur de la fenêtre (ces valeurs seront à ajuster si on a envie de se servir de ces boutons pour faire une barre de menu.

Le bouton est compris du texte et de deux barres horizontales (initialement à droite du texte pour la supérieure et à gauche du texte pour l'inférieure) qui seront amenées à se déplacer en cas de survol.

Ligne 7 : avec on dit que la zone que l'on va regarder se limitera au texte et avec on s'assure bien que tout ce qui dépasse de cette zone ne soit pas affiché (c'est pour ça que les barres sont invisibles en cas de survol.

`list-style-type:none`; Sert à dire que bien que les boutons soient les éléments d'une liste on ne mettra pas un point (ou un autre marqueur) devant chaque item

A terme le bouton sera interactif, c'est un hyperlien c'est pour cela qu'il est dans une balise `<a>` mais comme on n'a pas de destination la partie href du code html (ligne 13 du code HTML) est laissée vide.

Dégradés de couleurs :

Ils sont définis avec la fonction

Qui est bien détaillée ici :

<https://developer.mozilla.org/fr/docs/Web/CSS/linear-gradient>

Si on avait envie de faire un dégradé du jaune à l'orange sur une page on pourrait écrire : `<body style="background: -webkit-linear-gradient(#ffdd32, #fc9732);">`

Et si on avait envie d'avoir une transition plus localisée et selon une ligne diagonale (à 45deg) :

```
<body style="background: -webkit-linear-gradient(45deg, #ffdd32 0%, #ffdd32 45%, #fc9732 65%, #fc9732 100%);">
```

Pendant les 45 premiers pourcents pas de changement idem pour les 45 derniers.

Avec la ligne `-webkit-background-clip: text;` on redéfinit le background comme étant le texte autrement dit le dégradé sera appliqué sur le texte et non l'arrière.

`-webkit-text-fill-color: transparent;` nous permet de ne pas avoir de couleur à priori sur le texte

`animation: 10s bganim linear infinite;` permet d'avoir une animation appelée `bganim` durant `10s` et s'effectuant à vitesse régulière (`linear`) et ne s'arrêtant jamais (`infinite`)

L'animation en question est `@keyframes bganim { 100% { background-position: 100vw 0px; }}` et elle permet de faire bouger le background autrement dit la zone sur laquelle on va appliquer le dégradé orange et jaune ;


Deuxième animation

On associe à tout objet cumulant les classes « a » et « btn » deux petits items un placé avant (before) et l'autre après (after)

N'étant pas identiques mais ayants des points en commun les items sont définis en deux temps, d'abord au travers de leurs propriétés communes lignes 19 à 26 (le trait est long comme le texte associé, jaune et haut de 2 pixels, l'animation à venir sera effectuée en une demi seconde) puis

de manière individuelles séparément : lignes 27 à 29 pour l'item placé avant l'objet (le trait sera en haut et décalé d'une largeur de texte sur la droite) et 30 à 33 pour l'item placé après (placement du trait en bas et décalé d'une largeur de texte sur la gauche).

L'animation est déclenchée par le survol ( :hover) et elle consiste en ramener le bord gauche du trait complètement à gauche du mot.

**Bonus** : si on a envie de faire suivre n'importe quel lien externe (autrement dit commençant par « http ») d'un symbole  on peut rajouter les lignes suivantes dans le code CSS :

```
a[href^="http"]::after {
    background-image: url(https://s3-us-west-2.amazonaws.com/s.cdpn.io/161359/open-in-new.svg);
    background-size: contain;
    content: "";
    display: inline-block;
    vertical-align: middle;
    width: 1em;
    height: 1em;
}
```

`content` est vide car on n'écrit rien

`inline-block` indique qu'on reste aligné (`inline`) avec l'objet de départ et qu'en plus on va pouvoir ajuster la taille de l'objet(`block`).

## Bases de JavaScript

Je vous invite à télécharger et à consulter le document <http://www.dimension-k.com/maths/ISN/ISN-javascript-cours-kergot.pdf> il couvre bien les bases.

Mais reprenons tout de même ce qui a été vu durant ces dernières séances :

### Variables

Contrairement à Python, les variables sont à définir, et l'endroit où on le fera et la manière dont on le fera aura d'importantes conséquences qui seront évoquées en temps utile, pour l'instant faisons simple : Elles sont introduites généralement au début du code avec `var` nom et on peut leur donner des valeurs plus tard ou faire une pierre deux coups : `var nom =17`

### Communication directe avec l'utilisateur

pour afficher un message : `alert("message")`  
pour poser une question : `variable = prompt("question")`

### Boucles for

Contrairement à python ces boucles ne se font pas par parcours d'une liste ou d'une chaîne de caractère, mais par la modification d'un compteur étape après étape.  
La syntaxe sera *for(var NomDeVariable= valeur d'initialisation ; limite de comptage ; règle d'évolution de la variable) { actions à faire }*  
Par exemple :  
`for (var i=0;i<50;i++){document.write(i+"<br/>"); }`  
Va nous écrire les 50+1 premiers entiers sur la page web active.  
`i++` est une manière courte d'écrire `i=i+1`

### Travail sur les listes :

Pour explorer une liste ou une chaîne de caractères (qui pourrait être vue comme une liste de caractère collés

les uns aux autres) on écrira le nom de la liste suivi du rang du caractère qui nous intéresse. Attention le premier rang est 0 donc `liste1[7]` nous donnera l'élément de rang 7 de `liste1` , autrement dit le 8<sup>ème</sup> élément (7+1=8).

Pour éviter d'appeler un élément non défini, il peut être intéressant de déterminer la longueur d'une liste/chaîne, on le fera avec la méthode `length` : `liste1.length` donne le nombre d'éléments dans `liste1`.



## Exemple

Conversion en écriture décimale d'un nombre proposé en écriture binaire.

Brouillon

$$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$$

$$11011_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 0 + 2 + 1 = 27$$

$$101010_2 = 1 \times 10^5 + 0 \times 10^4 + 1 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 0 \times 10^0 = 32 + 0 + 8 + 0 + 2 + 0 = 42$$

Tentative de théorisation

Variable1= « 10101 »

Variable1[0]= « 1 »

Variable1[4]= « 1 »

|           |   |   |   |   |   |  |  |  |  |
|-----------|---|---|---|---|---|--|--|--|--|
| Puissance | 0 | 1 | 2 | 3 | 4 |  |  |  |  |
| rang      | 4 | 3 | 2 | 1 | 0 |  |  |  |  |

Puissance + rang = longueur de chaîne - 1

⇔ puissance = longueur de chaîne - 1 - rang

Algorithme

Demander nombreBin le nombre en écriture binaire

nombreDecimal=0

Pour i allant de 0 jusqu'à longueur de la chaîne -1

    Si nombreBin[i]=« 1 » alors

        nombreDecimal=nombreDecimal+2<sup>longueur de chaîne -1 - i</sup>

afficher nombreDecimal

Programme

```
var nombreBin=prompt("donne le nombre binaire") ;
var nombreDecimal=0 ;
for (var i=0;i<nombreBin.length;i++) {
    if (nombreBin[i]=="1") {
        nombreDecimal=nombreDecimal+Math.pow(2,nombreBin.length-1-
i);
    }
}
alert("l'écriture décimale du nombre sera "+ nombreDecimal)
```