CSS3

Mettre en place le CSS

Grâce au HTML, nous avons pu rédiger le contenu de notre site mais il est brut. Le CSS vient compléter ce code pour mettre en forme tout cela et donner au contenu l'apparence que l'on souhaite.

Où écrit-on le CSS ?

Vous avez le choix car on peut écrire du code en langage CSS à trois endroits différents :

- **1ère méthode :** dans un fichier .css (*méthode la plus recommandée*) ;
- **2ème méthode :** dans l'en-tête **<head>** du fichier HTML ;
- **3ème méthode :** directement dans les balises du fichier HTML *via* un attribut style (*méthode la moins recommandée*).

1ère méthode :

Page HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Premiers tests du CSS</title>
  </head>
  <body>
    <h1>Mon super site</h1>
    <p>Bonjour et bienvenue sur mon site !</p>
  </body>
</html>
```

Feuille CSS

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

Vous noterez le contenu de la ligne 5, `<link rel="stylesheet" href="style.css"/>` : c'est elle qui indique que ce fichier HTML est associé à un fichier appelé style.css et chargé de la mise en forme.

2ème méthode :

Il existe une autre méthode pour utiliser du CSS dans ses fichiers HTML : cela consiste à insérer le code CSS directement dans une balise **<style>** à l'intérieur de l'en-tête **<head>**.

Page HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Premiers tests du CSS</title>
    <style>
        p { color: blue; }
    </style>
  </head>
  <body>
    <h1>Mon super site</h1>
    <p>Bonjour et bienvenue sur mon site !</p>
  </body>
</html>
```

3ème méthode :

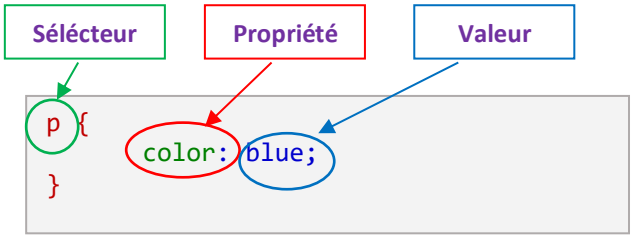
Dernière méthode, à manipuler avec précaution : vous pouvez ajouter un attribut style à n'importe quelle balise. Vous insérerez votre code CSS directement dans cet attribut :

Page HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="style.css" />
    <title>Premiers tests du CSS</title>
    <style>
        p { color: blue; }
    </style>
  </head>
  <body>
    <h1>Mon super site</h1>
    <p style="color: blue;">Bonjour et bienvenue sur mon site !</p>
    <p>Pour le moment, mon site est un peu vide. Patientez encore
un peu !</p>
  </body>
```

Cette fois, seul le texte du premier paragraphe (ligne 11), dont la balise contient le code CSS, sera coloré en bleu.

Syntaxe générale



Appliquer un style à plusieurs balises

```
p, h1 {
        color: blue;
}
```

Cela signifie : « *Je veux que le texte de mes **<h1>** et **<p>** soit écrit en bleu* ».

Des commentaires dans du CSS

Pour faire un commentaire, c'est facile ! Tapez /*, suivi de votre commentaire, puis */ pour terminer votre commentaire.

```
p {
        color: blue;
        /* Ceci est un commentaire */
}
```

Appliquer un style : class et id

Comment faire pour que certains paragraphes seulement soient écrits d'une manière différente ?
Pour résoudre le problème, on peut utiliser ces attributs spéciaux *qui fonctionnent sur toutes les balises* :

- L'attribut **class**
- L'attribut **id**

Les attributs class et id sont quasiment identiques. Mais généralement id est unique.

Code HTML

```
<h1 class="intro"> … </h1>
<p class="intro"> … </p>
<img id="photo" …/>
```

Code CSS

```
.intro { color: blue; }
#photo { width: 100%; }
```

La couleur bleue s'applique sur tous les balises qui ont comme class « intro ». et la largeur 100% s'applique sur l'image qui a comme id « photo ».

Les sélecteurs avancés

*** :** Sélectionne toutes les balises sans exception. On l'appelle le **sélecteur universel**.

```
* {
        color: blue;
}
```

Cela signifie : « *Je veux que tout le texte de ma page soit écrit en bleu* ».

element[attr] : Sélectionne toutes les balises **element** qui ont l'attribut **attr**.

```
p[title] {
        color: blue;
}
```

Cela signifie : « *Je veux que tous les paragraphes qui ont l'attribut title soient écrits en bleu* ».

element[attr="Valeur"] : Sélectionne toutes les balises **element** qui ont l'attribut **atrr** = **"Valeur"**

```
p[title="montexte"] {
        color: blue;
}
```

Cela signifie : « *Je veux que tous les paragraphes qui ont l'attribut title*=**"montexte"** *soient écrits en bleu* ».

element.maclasse : Sélectionne toutes les balises element qui ont comme classe « maclasse »

```
p.intro {
        color: blue;
}
```

Cela signifie : « *Je veux que tous les paragraphes qui sont de classe* **"intro"** *soient écrits en bleu* ».

a:link : Sélectionne tous les liens non visités
a:visited : Sélectionne tous les liens visités

```
a:link { color: blue; }
a:visited { color: green; }
```

Cela signifie : « *Je veux que tous les liens non visités soient écrits en bleu alors que les liens visités soient écrits en vert* ».

element:hover : Sélectionne l'élément au moment où le pointeur de la souris le survole
element:active : Sélectionne l'élément au moment où il est activé par un clic de la souris

```
h1:hover { color: blue; }
h1:active { color: green; }
```

Cela signifie : « *Je veux que tous les éléments h1 soient de couleur bleu lorsqu'isl sont survolés par la souris et soient de couleur vert lorsqu'ils sont cliqués par la souris* ».

Formatage du texte

La Taille `font-size`

Spécifie la taille du texte.

```
p {
        font-size: 16px;
}
```

Tous les paragraphes seront du taille 16 pixels.

La Police `font-family`

Spécifie la police du texte.Il est possible d'indiquer plusieurs polices séparées par des virgules (,). Si le premier Police est absent, le deuxième le remplacera, etc.

```
p {
        font-family: Arial, Impact;
}
```

Tous les paragraphes seront de Police Arial. Si l'Arial n'existe pas dans le navigateur, il sera remplacé par l'Impact.

Le Style `font-style`

Spécifie le style du texte. Les valeurs possibles sont : **italic** et **normal**.

```
h1 {
        font-style: italic;
}
```

Tous les titres h1 seront de style italique.

L'épaisseur `font-weight`

Spécifie l'epaisseur du texte. Les valeurs possibles sont : **bold** (gras), **bolder** (plus gras), **lighter** (plus léger) et **normal**.

```
h1 { font-weight: bolder; }
h2 { font-weight: bold; }
```

Tous les titres h1 seront plus gras et tous les titres h2 seront gras.

Super-propriété de font `font`

Elle englobe toutes les propriétés du font dans une seule.
L'ordre des valeurs doit respecter l'une des rangements suivants :

- **font-style font-weight font-size font-family**
- **font-weight font-style font-size font-family**

Attention : font-family et font-size sont obligatoires. font-style et font-weight peuvent être omis.

```
h1 {
      font: bold italic 16px Impact;
}
```

Tous les titres h1 seront gras, italiques, du taille 16 pixels et du police Impact.

L'alignement `text-align`

Spécifie l'alignement horizontal du texte. Les valeurs possibles sont : **left**, **right**, **center** et **justify**.

```
h1 {
      text-align: center;
}
```

Tous les titres h1 seront alignés au centre.

L'ombre d'un texte `text-shadow`

Ajoute un ombre au texte. Il faut définr le décalage horizontal, le décalage vertical et la couleur de l'ombre dans l'un des ordres suivants :

- **couleur décalage-x décalage-y**
- **décalage-x décalage-y couleur**

Attention : le décalage peut être un nombre négatif.

```
h1 {
      text-shadow: gray 3px 5px;
}
```

Tous les titres h1 auront une ombre de couleur gris décalée horizontalement 3 pixels et décalée verticalement 5 pixels.

La transformation d'un texte `text-transform`

Transforme les caractères d'un texte en majuscule ou en minuscule. Les valeurs possibles sont : **uppercase** (tous les caractères en majuscules), **lowercase** (tous les caractères en minuscules) et **capitalize** (le premier caractère pour chaque mot en majuscule).

```
h1 { text-transform: capitalize; }
p { text-transform: lowercase; }
```

Tous les mots des titres h1 commenceront par des majuscules alors que les paragraphes seront en minuscules.

La couleur d'un texte `color`

Spécifie la couleur du texte. Il existe trois façons pour exprimer la couleur désirée :

- Mentionner directement le nom de la couleur. Les couleurs possibles sont: **white**, **silver**, **gray**, **black**, **red**, **maroon**, **lime**, **green**, **yellow**, **olive**, **blue**, **navy**, **fuchsia**, **purple**, **aqua** et **teal**.
- Utiliser la notation **hexadécimale** sous la forme **#RRGGBB**, c'est-à-dire pour chaque degré de rouge, de vert ou de bleu on associe deux chiffres héxadécimaux (un chiffre héxadécimal peut être : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f). Il est possible aussi d'utiliser un seul chiffre pour chaque dégré de couleur : **#RGB**.
- Utiliser la notation **rgb** sous la forme **rgb(R, G, B)**, c'est-à-dire pour chaque degré de rouge, de vert ou de bleu on associe un nombre décimal entre 0 et 255. Il est possible aussi d'utiliser la notation similaire **rgba** sous la forme **rgba(R, G, B, A)** où A est un nombre entre 0 et 1 qui represente le degré de transparence.

```
h2 { color: red; }
p { color: #55ff22; }
```

Tous titres h seront de couleur rouge alors que les paragraphes seront de couleur #55ff22.

Arrière-plan

Couleur d'arrière-plan `background-color`

Permet de définir la couleur de l'arrière-plan d'un élément donné.

```
nav {
        background-color: rgba(135,12,45,0.6);
}
```

La zone nav aura comme couleur d'arrière-plan la couleur rgba(135,12,45,0.6).

Image d'arrière-plan `background-image`

Permet de définir une image comme arrière-plan d'un élément donné.

```
#titre {
        background-image: url("media/img1.jpg");
}
```

La zone d'id titre aura comme arrière-plan l'image "media/img1.jpg".

Répétition d'arrière-plan `background-repeat`

Définit la façon dont l'image utilisée en arrière-plan est répétée. Une image d'arrière-plan pourra ainsi :

- **repeat-x** : répétée sur l'axe horizontal.
- **repeat-y** : répétée sur l'axe vertical.
- **repeat** : répétée sur les deux axes.
- **no-repeat** : ne pas répétée.

```
#titre {
        background-image: url("media/img1.jpg");
        background-repeat: repeat-x; }
```

La zone d'id titre aura comme arrière-plan l'image "media/img1.jpg" qui se répètera horizontalement.

Taille d'arrière-plan `background-size`

Définit la taille d'image d'arrière-plan pour l'élément. Les valeurs possibles sont :

- **cover** : l'image sera étirée sur toute l'espace.
- **largeur hauteur**: Les valeurs peuvent être en pixels (exemple 200px) ou en pourcentage (exemple 30%).
- **largeur** : en pixels ou en pourcentage. La hauteur sera automatique.

Attention : une largeur est en % désigne une largeur par rapport la largeur de la zone d'arrière-plan. Et une hauteur est en % désigne une hauteur par rapport la hauteur de la zone d'arrière-plan.

```
#titre { background-image: url("media/img1.jpg");
          background-size: 250px; }
aside { background-image: url("media/img2.jpg");
          background-size: 40% 200px; }
footer { background-image: url("media/img3.jpg");
          background-size: cover; }
```

"img1.jpg" en arrière-plan, sa largeur est 250 pixels et sa hauteur est automatique. "img2.jpg" en arrière-plan, sa largeur est 40% de la largeur de la zone aside et sa hauteur est 200 pixels. "img3.jpg" en arrière-plan, étirée sur toute la zone footer.

Super-propriété d'arrière-plan `background`

Combine toutes les propriétés d'arrière-plan dans l'ordre :

background-color background-image 0px/background-size background-repeat

```
#titre {
    background: url("media/img1.jpg") 0px/100px repeat;
}
```

La zone d'id titre aura comme arrière-plan l'image "media/img1.jpg" de largeur 100 pixels et hauteur auto, répété sur les deux axes x et y.

Attention : **background-position** et **background-size** sont inséparables dans la super-propriété background, si on veut mentionner l'un il faut mentionner l'autre avec la syntaxe : **background-position/background-size**. Puisque background-position est hors-programme, retenir seulement le fait de le remplacer par 0px.

Les bordures

Couleur de la bordure `border-color`

Spécifie la couleur de la bordure d'un élément.

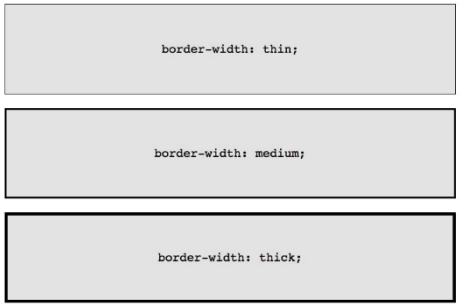
```
h2 {
        border-color: blue;
}
```

Le titre h2 aura une bordure bleue pour les 4 cotés.

Largeur de la bordure `border-width`

Spécifie la largeur de la bordure d'un élément.
On mentionne la largeur par l'une des façons suivantes :

- Valeur en pixels.
- **thin** : trait mince.
- **medium** : trait moyen.
- **thick** : trait épais.



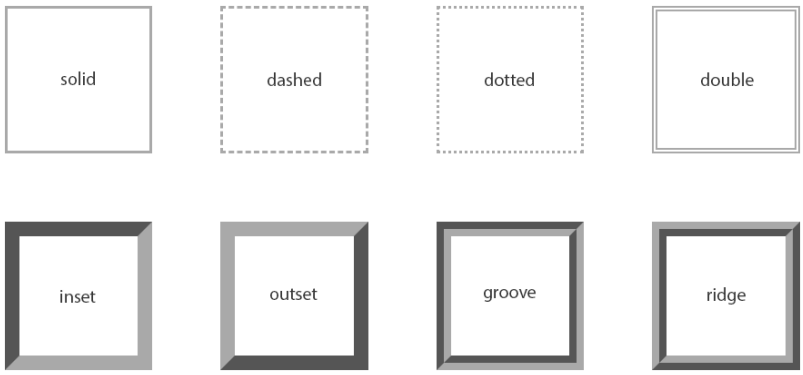

```
h2 {
        border-width: thin;
}
```

Le titre h2 aura une bordure mince pour les 4 cotés.

Style de la bordure `border-style`

Spécifie le style de la bordure d'un élément.
Les valeurs possibles pour cette propriété sont :

- **none** (pas de trait)
- **solid** (trait continu)
- **dotted** (trait en pointillés)
- **dashed** (trait discontinu)
- **double** (trait double)
- **groove** (trait en relief)
- **ridge** (autre trait en relief)
- **inset** (effet 3D enfoncé)
- **outset** (effet 3D sortant)




```
h2 {
        border-style: solid;
}
```

Le titre h2 aura une bordure trait continu pour les 4 cotés.

Super-propriété de la bordure `border`

Combine toutes les propriétés de la bordure dans l'ordre :
border-width border-style border-color ou **border-width border-color border-style**

```
#titre {
    border: medium dashed black;
}
```

La zone d'id titre aura comme bordure de 4 côtés un trait de largeur medium, en pointillé et noir.

En haut, à droite, à gauche, en bas…

Il est possible pour les propriétés ci-dessus (**border-color**, **border-width**, **border-style** et **border**) de mentionner une valeur différente pour chaque côté avec l'une des méthodes suivantes :

1ère méthode : utiliser les propriétés détaillées :

- Pour la couleur: **border-left-color**, **border-right-color**, **border-top-color**, **border-bottom-color**.
- Pour la largeur: **border-left-width**, **border-right-width**, **border-top-width**, **border-bottom-width**.
- Pour le style: **border-left-style**, **border-right-style**, **border-top-style**, **border-bottom-style**.
- Pour la super-propriété: **border-left**, **border-right**, **border-top**, **border-bottom**.

2ème méthode : affecter plusieurs valeurs à la propriété :

- Avec **deux valeurs**, la 1ère s'applique aux côtés **haut et bas** et la 2ème aux côtés **gauche et droite**.
- Avec **trois valeurs**, la 1ère s'applique au côté **haut**, la 2ème aux côtés **gauche et droite** puis la 3ème au côté **bas**.
- Avec **quatre valeurs**, la 1ère sur le côté **haut**, la 2ème sur le côté **droite**, la 3ème sur le côté **bas** et la 4ème sur le côté **gauche**.

```
h3 {  border-left-color: red;
      border-top-color: green; }
.t1 { border-color: blue gray red green; }
p {  border-color: blue yellow red; }
p {  border-width: 1px 2px; }
#par { border-left: 2px solid red;
       border-right: 1px dashed green; }
```

h3 : bordure rouge à gauche et vert en haut.
.t1 : bordure bleue en haut, gris à droite, rouge en bas et vert à gauche.
p : bordure bleue en haut, jaune à gauche et droite, rouge en bas.
p : bordure 1px en haut et en bas, 2px à gauche et à droite.
#par : bordure de 2px en trait continu rouge à gauche et de 1px en trait discontinu vert à droite.

Attention : la 2ème méthode n'est pas valable pour la super-propriété border. Avec la super-propriété **border** les 4 cotés ne peuvent qu'être similaires.

Bordure arrondie `border-radius`

Permet de définir des coins arrondis pour la bordure d'un élément.

Les valeurs possibles pour cette propriété :

- **Une valeur en pixels** : indique la mesure de rayon de la courbure.
- **Un pourcentage** : indique la mesure de rayon de la courbure exprimée en pourcentage par rapport à la taille de la boîte (élément dont on effectue son bordure arrondie).

```
#titre {
    border-radius: 20px;
}
```

La zone d'id titre aura une courbure de bordure pour les 4 coins d'un rayon 20 pixels.

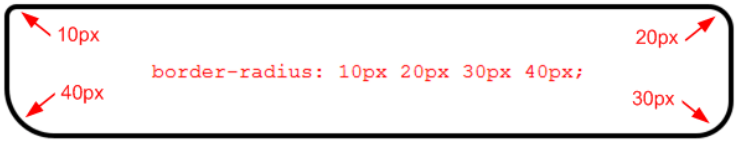
Il est possible de faire une courbure special pour chaque coin, il suffit d'utiliser l'une des méthodes suivantes :

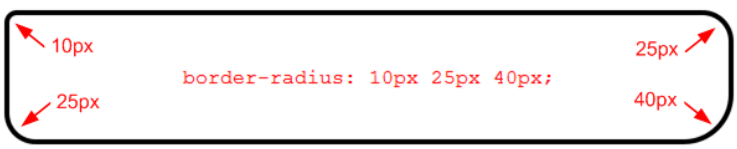
1ère méthode : utiliser les propriétés détaillées :

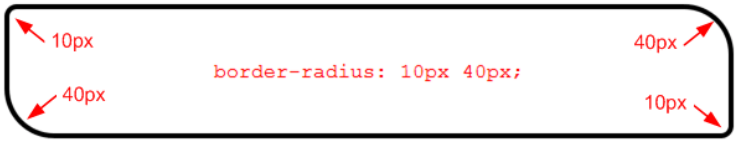
border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius

2ème méthode : affecter plusieurs valeurs à la propriété **border-radius** :

- Avec **deux valeurs** :
  - la 1ère s'applique sur les coins **haut-gauche** et **bas-droite.**
  - la 2ème s'applique sur les coins **bas-gauche** et **haut-droite**.
- Avec **trois valeurs :**
  - la 1ère s'applique sur le coin **haut-gauche.**
  - la 2ème s'applique sur les coins **bas-gauche** et **haut-droite**.
  - la 3ème s'applique sur le coin **bas-droite**.
- Avec **quatre valeurs** :
  - la 1ère s'applique sur le coin **haut-gauche.**
  - la 2ème s'applique sur le coin **haut-droite.**
  - la 3ème s'applique sur le coin **bas-droite**.
  - la 4ème s'applique sur le coin **bas-gauche.**







Modèle de boite

En CSS, tout élément est inclus dans une boîte ("*box*" en anglais). Comprendre le fonctionnement de ces boîtes est essentiel pour maîtriser la mise en page CSS ainsi que le positionnement des éléments d'une page HTML.

En CSS, il existe deux types de boîtes : les boîtes **en bloc** (*block boxes*) et les boîtes **en ligne** (*inline).*

Si une boîte est définie en **bloc**, elle suivra alors les règles suivantes :

- La boîte s'étend en largeur pour remplir totalement l'espace offert par son conteneur. Dans la plupart des cas, la boîte devient alors aussi large que son conteneur, occupant 100% de l'espace disponible.
- La boîte occupe sa propre nouvelle ligne et crée un retour à la ligne, faisant ainsi passer les éléments suivants à la ligne d'après.
- Presque toutes les balises (p, h1, img, …) sont de type bloc, mais on peut les changer ultérieurement en ligne.

Si une boîte est définie en **ligne**, alors :

- La boîte ne crée pas de retour à la ligne, les autres éléments se suivent donc en ligne.
- Exemple : les balises <a> et .

Ombre de boite `box-shadow`

Permet de définir un ombre à toute la boite d'un élément.

```
#titre {
    box-shadow: 20px 20px gray;
}
```

La zone d'id titre aura un ombre décalé horizontalement (offset-x) de 20px, décalé verticalement (offset-y) de 20 px et de couleur gris.

Dépassement de contenu `overflow`

En donnant des dimensions précises pour les blocs, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent. La propriété **overflow** sert à contrôler les dépassements.

Les valeurs acceptées par cette propriété sont :

- **visible** (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
- **hidden** : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra pas voir tout le texte.
- **scroll** : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte. C'est un peu comme un cadre à l'intérieur de la page.
- **auto** : c'est le mode « pilote automatique ». En gros, c'est le navigateur qui décide de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire).
- **clip** : presque même que **hidden**.

HTML

```
<p id='p1'>Je suis la première paragraphe qui va
dépasser tous les limites</p>
<p id='p1'>Je suis la deuxième paragraphe </p>
```

CSS

```
#p1{
  width:150px;
  height: 60px;
  overflow:visible;
}
```

Je suis la première paragraphe qui va dépasser tous les limites

Je suis la deuxième paragraphe

```
#p1{
  width:150px;
  height: 60px;
  overflow:hidden;
}
```

Je suis la première paragraphe qui va

Je suis la deuxième paragraphe

```
#p1{
  width:150px;
  height: 60px;
  overflow:scroll;
}
```

Je suis la

Je suis la

```
#p1{
  width:150px;
  height: 60px;
  overflow:auto;
}
```

Je suis la première

Je suis la deuxième paragraphe

Transparence `opacity`

La propriété **opacity** permet d'indiquer le niveau d'opacité (c'est l'inverse de la transparence).

- Avec une valeur de 1 (ou 100%), l'élément sera totalement opaque : c'est le comportement par défaut.
- Avec une valeur de 0 (ou 0%), l'élément sera totalement transparent (invisible).

HTML
```
<p>salut les amis</p>
```

CSS
```
p {
  background-color: black;
  color: white;
  opacity: 0.3;
}
```

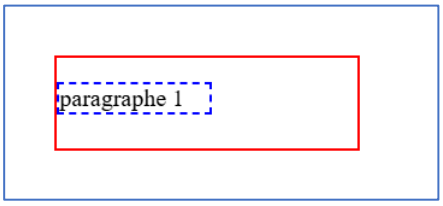
Les dimensions `width` et `height`

Pour commencer, intéressons-nous à la taille des blocs. Contrairement à une balise inline, un bloc a des dimensions précises. Il possède une largeur et une hauteur.

- **width** : c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).
- **height** : c'est la hauteur du bloc. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

HTML
```
<div>
   <p>paragraphe 1</p>
</div>
```

CSS
```
p {   width:50%;
      height: 30%;
      border:2px blue dashed; }
div { width:200px;
      height: 60px;
      border:2px red solid; }
```



Attention : le pourcentage indique le rapport **largeur de l'élément/largeur de son conteneur** (de même pour la hauteur). Par exemple une largeur de 60% indique que la largeur représente 60% de la largeur du l'élément conteneur.

Positionnement

Il existe plusieurs techniques permettant d'effectuer la mise en page de son site. Chacune a ses avantages et ses défauts.

Le positionnement avec display `display`

Sachant que la plupart des balises html peuvent être classées selon leurs types :

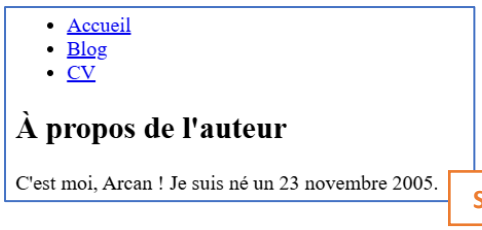
Type	Balise (liste non exhaustive)	Description
inline	**<a>, <cite>, <mark>, , <label>, <caption>, <legend>**	Eléments d'une ligne. Se placent les uns à côté des autres.
block	**<address>, <article>, <aside>, <body>, , <fieldset>, <form>, <footer>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <header>, <hr>, ,<main>, <nav>, , <p>, <script>, <section>, <style>, <table>, <tbody>, <tfoot>, <th>, <thead>, <tr>, **	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
inline-block	**<img/>, <input/>, <select>, <textarea>**	Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).

La propriété **display** permet de changer le comportement de balise (par exemple rendre une balise en block à inline).
Les valeurs possibles de la propriété display :

- **inline** : l'élément se comporte comme un élément **inline** ; il accepte d'être placé dans la même ligne avec des autres éléments. Et il devient non redimensionnable même s'il a les propriétés **width** et **height**.
- **block** : l'élément se comporte comme un élément **block**. Il occupe une ligne et n'accepte pas d'autres éléments à coté de lui. Et il devient redimensionnable avec les propriétés **width** et **height**.
- **inline-block** : l'élément se comporte comme un élément **inline** en acceptant d'être placé dans la même ligne avec des autres éléments. Mais aussi il se comporte comme un élément **block** en acceptant son redimensionnement à l'aide des propriétés **width** et **height**.

HTML
```
<nav>
  <ul>
    <li><a href="#">Accueil</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">CV</a></li>
  </ul>
</nav>
```

CSS
```
nav{
  display: inline-block;
  width:30%;
}
section {
  display: inline-block;
  width:69%;
```



SANS CSS

AVEC CSS

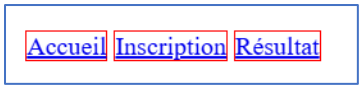
Par défaut, les balises **<nav>** et **<section>** sont de type **block** (donc elles viennent l'un au-dessous de l'autre). En CSS, on a modifié la nature de deux balises en **inline-block** (viennent l'un à coté de l'autre) à l'aide de la propriété **display**.
Ici, on n'a pas modifié la nature des balises à **inline** pour pouvoir les redimensionner (**inline** est non redimensionnable).

```html
<a href="acc.html">Accueil</a>
<a href="ins.html">Inscription</a>
<a href="res.html">Résultat</a>
```

HTML

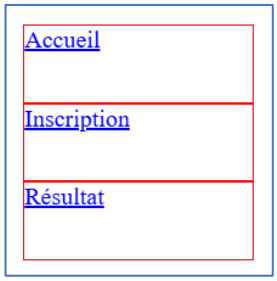
```css
a{ width:150px;
   height:50px;
   border:1px red solid; }
```

CSS




```css
a{display:block;
  width:150px;
  height:50px;
  border:1px red solid;}
```

CSS



Par défaut, la balise **<a>** est de type **inline** donc les éléments de cette balise viennent dans la même ligne et le redimensionnement est désactivé pour eux (**width** et **height** n'ont pas d'effet). Lorsqu'on modifie la nature de la balise en **block** les éléments de cette balise occupe chacun d'eux une ligne propre à lui. Et le redimensionnement (**width** et **height**) prend effet sur les éléments.

Le positionnement avec position

position avec left, top, right et bottom

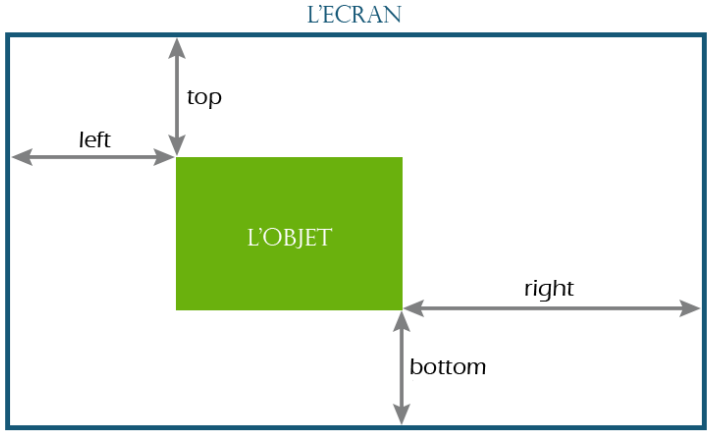
Il est possible de positionner les éléments de la page à l'aide de la propriété **position** accompagné d'une collection des propriétés de positionnement (**left**, **right**, **top** et **bottom**).

Les valeurs possibles de la propriété **position** :

- **static** : L'élément est positionné selon le flux normal du document. Les propriétés **top**, **right**, **bottom** et **left** n'ont aucun effet. Il s'agit de la valeur par défaut.
- **absolute** : L'espace initialement réservé à l'élément disparait. Et l'élément est positionné (par les valeurs de **top**, **right**, **bottom** et **left**) **par rapport à la page (ou par rapport le bloc conteneur s'il est positionné lui aussi)**.
- **relative** : L'espace initialement réservé à l'élément reste réservé à lui. Et l'élément est positionné (par les valeurs de **top**, **right**, **bottom** et **left**) **par rapport à sa position initial**.
- **fixed** : L'espace initialement réservé à l'élément disparait. Et l'élément est positionné (par les valeurs de **top**, **right**, **bottom** et **left**) **par rapport à la page**. **De plus, si la page est scrollée, l'élément fixé reste dans la même position.**
- **sticky :** se comporte comme **relative**. Lorsque la page est scrollée, il se comporte comme **fixed**.

On a besoin à côté de la propriété position, il est indispensable d'utiliser l'une ou plusieurs des propriétés suivantes :

- **left** : distance (en pixels ou en %) entre la borne gauche de l'élément et son **bloc conteneur**.
- **top** : distance (en pixels ou en %) entre la borne haute de l'élément et son **bloc conteneur**.
- **right** : distance (en pixels ou en %) entre la borne droite de l'élément et son **bloc conteneur**.
- **bottom** : distance (en pixels ou en %) entre la borne basse de l'élément et son **bloc conteneur.**



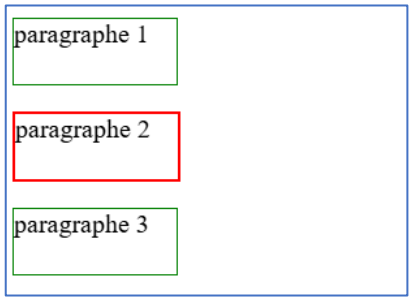
Attention :
En cas de **position relative**, les propriétés **left**, **top**, **right** et **bottom** donnent les mesures à partir de la position initiale de l'élément et non par rapport à **bloc conteneur**.
La distance **left** ou **right** en % désigne une mesure par rapport au largeur (**width**) du **bloc conteneur**.
La distance **top** ou **bottom** en % désigne une mesure par rapport à la hauteur (**height**) du **bloc conteneur**.

HTML

```
<div><p>paragraphe 1</p>
     <p id="pos">paragraphe 2</p>
     <p>paragraphe 3</p></div>
```

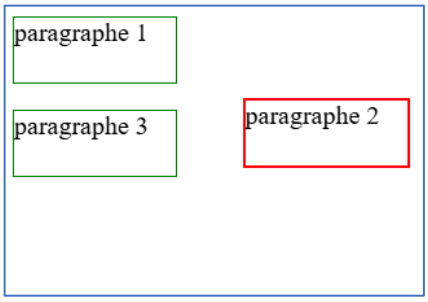
CSS

```
#pos { position:static;
       top:50px;
       left:150px; }
```



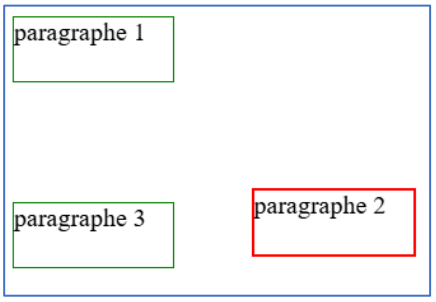
top et **left** n'ont pas d'effet car la **position** est **static**.

```
#pos { position:absolute;
       top:50px;
       left:150px; }
```



Le déplacement se fait par rapport à la page.
L'ancienne place est occupée.

```
#pos { position:relative;
       top:50px;
       left:150px; }
```

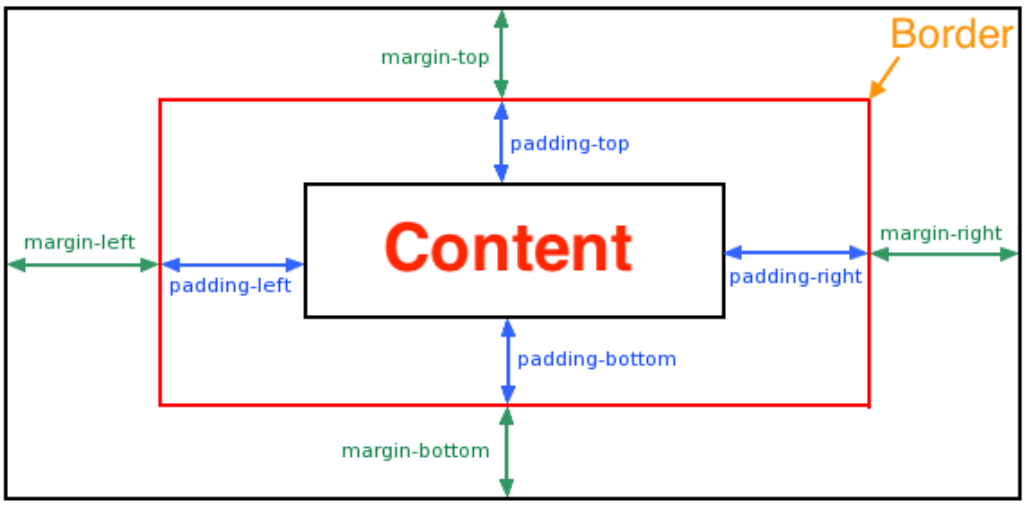


Le déplacement se fait par rapport à l'ancienne position.
L'ancienne place reste réservée.

Les marges `padding et margin`

Il faut savoir que tous les blocs possèdent des marges. Il existe *deux types de marges* :

- Les marges intérieures : **padding** et ses dérivés : **padding-top**, **padding-bottom**, **padding-left** et **padding-right**.
- Les marges extérieures : **margin** et ses dérivés : **margin-top**, **margin-bottom**, **margin-left** et **margin-right**.



Il est possible pour ces propriétés de mentionner une valeur différente pour chaque côté avec l'une des méthodes suivantes :

1ère méthode : utiliser les propriétés détaillées :

- Pour la marge intérieure: **padding-top**, **padding-bottom**, **padding-left** et **padding-right**.
- Pour la marge extérieure: **margin-top**, **margin-bottom**, **margin-left** et **margin-right**.

2ème méthode : affecter plusieurs valeurs à la propriété :

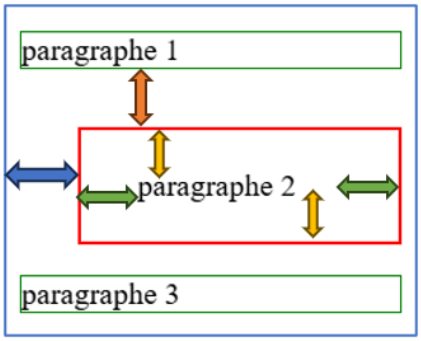
- Avec **deux valeurs**, la 1ère s'applique **en haut et en bas** et la 2ème **à gauche et à droite**.
- Avec **trois valeurs**, la 1ère s'applique **en haut**, la 2ème **à gauche et à droite** puis la 3ème **en bas**.
- Avec **quatre valeurs**, la 1ère **en haut**, la 2ème **à droite**, la 3ème **en bas** et la 4ème **à gauche**.

```
<div><p>paragraphe 1</p>
     <p id="pos">paragraphe 2</p>
     <p>paragraphe 3</p></div>
```

HTML

```
#pos{
  border:2px red solid;
  margin-top:30px;
  margin-left:30px;
  padding:20px 30px;
}
```

CSS



Pour centrer un bloc dans une page, il suffit de :
- donner une largeur (**width**) à ce bloc.
- mettre la marge extérieure à automatique : **margin : auto ;**

Filtre des images `filter`

Cette propriété permet de définir un filtre sur un élément. Généralement appliqué sur des images, le filtre peut s'appliquer aussi à des arrière-plans ou des bordures.

- **blur(r)** : applique un flou de rayon r.
- **grayscale( )** : convertit l'image en niveaux de gris. Un paramètre peut être présent pour indiquer le degré de conversion en gris.
- **invert( )** : inverse les couleurs de l'image. Un paramètre peut être présent pour indiquer le degré de l'inversion.

`img{ filter:blur(5px); }`	Appliquer un flou de rayon 5 pixels.
`img{ filter:grayscale(.7); }`	Rendre l'image en niveaux de gris à 70%.
`img{ filter:invert(90%); }`	Inverser les couleurs de l'image à 90%.

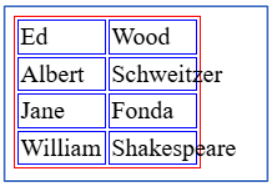
Propriétés des tableaux

Disposition de la table `table-layout`

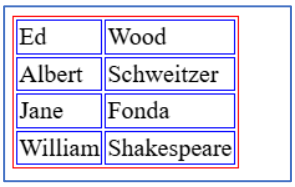
Définit la manière dont les cellules, les lignes et les colonnes sont disposées.
Les valeurs possibles sont :

- **auto** : Les largeurs du tableau et de ses cellules sont ajustées pour s'adapter au contenu. La plupart des navigateurs utilisent auto par défaut.
- **fixed** : Lorsque vous utilisez ce mot-clé, la largeur du tableau doit être spécifiée explicitement à l'aide de la propriété **width**.

```
table { table-layout: fixed;
        width: 120px;
        border: 1px solid red; }
```




```
table { table-layout: auto;
        width: 120px;
        border: 1px solid red; }
```

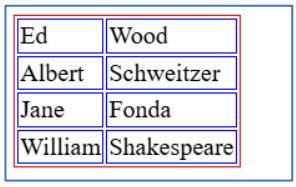


Bordure des cellules `border-collapse`

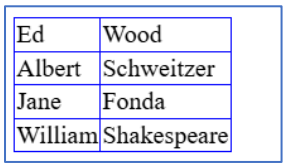
Définit la manière dont les bordures des cellules sont disposées.
Les valeurs possibles sont :

- **separate** : Les cellules adjacentes ont des bordures distinctes. C'est la valeur par défaut.
- **collapse** : Les cellules adjacentes ont des bordures partagées.

```
table { border-collapse: separate; }
```




```
table { border-collapse: collapse; }
```



Propriétés des listes

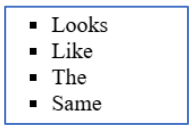
Type de marqueur `list-style-type`

Définit le type marqueur des éléments de liste.

- Les valeurs possibles pour les listes non ordonnées sont : **circle** (cercle) et **square** (carré).
- Les valeurs possibles pour les listes ordonnées sont : **upper-roman** (chiffres romains) et **lower-alpha** (lettres miniscules).

```
ol { list-style-type: lower-alpha; }
```


```
ul { list-style-type: square; }
```



Position de marqueur `list-style-position`

Définit la position des marqueurs des éléments de la liste.
Les valeurs possibles sont :

- **outside** : le marqueur de la liste est toujours dans un niveau séparé au texte. C'est la valeur par défaut.
- **inside** : lorsque le texte revient à la ligne il sera aligné sous le marqueur de la liste.

```
ul { list-style-position: outside; }
```

```
ul { list-style-position: inside; }
```

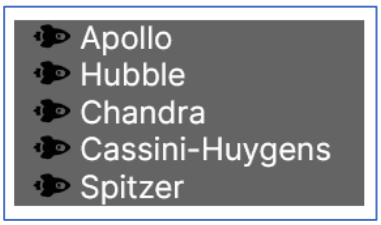
- Looks
- Like Like Like Like Like Like Like Like Like Like
- The
- Same

- Looks
- Like Like Like Like Like Like Like Like Like Like
- The
- Same

Image de marqueur `list-style-image`

Permet de définir une image comme marqueur de la liste non ordonnée.

```
ul { list-style-image: url("star-solid.gif"); }
```



Super-Propriété `list-style`

Combine les propriétés vues ci-dessous dans l'ordre :

list-style-type list-style-position list-style-image

```
ul { list-style: square inside url("star-solid.gif"); }
```

Si le type et l'image sont présent ensemble. L'image a la priorité d'apparaitre. Le type apparait si l'image est non chargée

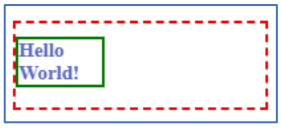
Les transformations `transform`

La propriété **transform** applique une transformation 2D à un élément. Cette propriété vous permet de faire pivoter, de mettre à l'échelle, de déplacer, d'incliner, etc. des éléments.

Translation `translate()`

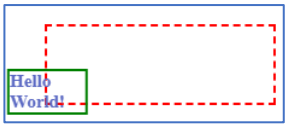
Permet d'effectuer une translation sur l'élément ciblé.

- **translate(A, B)** permet de décaler la position de l'élément de valeur **A** sur l'axe des **X** et de valeur **B** sur l'axe des **Y**.
- **translate(A)** permet de décaler la position de l'élément de valeur **A** sur l'axe des **X**.
- **A** et **B** peuvent être des valeurs en pixels ou en pourcentage. **A** en pourcentage désigne le décalage en **X** par rapport à la largeur de l'élément. Tandis que **B** en pourcentage désigne le décalage en **Y** par rapport à la hauteur de l'élément.
- Une valeur **positive** de **A** désigne un décalage à **gauche** tandis qu'une valeur **négative** désigne un décalage à **droite**.
- Une valeur **positive** de **B** désigne un décalage en **bas** tandis qu'une valeur **négative** désigne un décalage en **haut**.



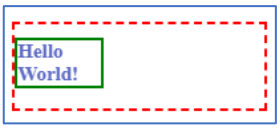

```
h1{ transform:translate(30px); }
```


```
h1{ transform:translate(-50%, 20px); }
```

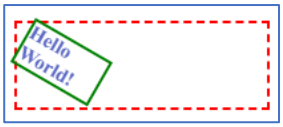


Rotation `rotate()`

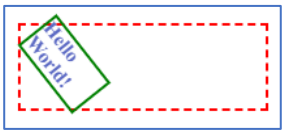
Permet d'effectuer une rotation d'un angle donné sur l'élément ciblé. L'angle peut être exprimé en degré ou en radian. L'angle de rotation peut être mesuré en degré (**deg**) ou en radian (**rad**).




```
h1{ transform:rotate(30deg); }
```




```
h1{ transform:rotate(.9rad); }
```



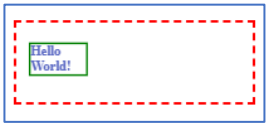
Echelle `scale()`

Permet d'effectuer une augmentation ou une diminution de taille sur l'élément ciblé.

- **scale(A, B)** permet de rendre la largeur **A** fois et la hauteur **B** fois.
- **scale(A)** permet de rendre la largeur **A** fois et la hauteur **A** fois.
- Une valeur **supérieure à 1** de **A** (respectivement **B**) engendre une **augmentation** de largeur (respectivement hauteur).
- Une valeur **inférieure à 1** de **A** (respectivement **B**) engendre une **diminution** de largeur (respectivement hauteur).


```
h1{ transform:scale(2,1.5); }
```

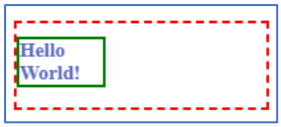

```
h1{ transform:scale(.7); }
```



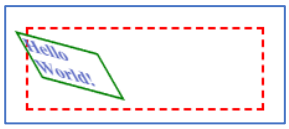
Inclinaison `skew()`

Permet d'incliner un élément le long des axes X et Y selon les angles donnés.

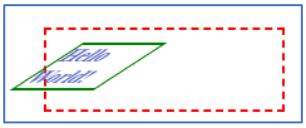
- **skew(A, B)** permet d'incliner l'élément d'un angle **A** pour l'axe des X et d'un angle **B** pour l'axe des Y.
- **skew(A)** permet d'incliner l'élément d'un angle **A** pour l'axe des X (l'inclinaison sur l'axe des Y est nulle).




```
h1{ transform:skew(30deg,15deg); }
```




```
h1{ transform:skew(-1rad); }
```



Les transitions `transition`

Les transitions permettent de modifier les valeurs des propriétés CSS d'un élément en douceur, sur une durée donnée lorsqu'il est cliqué ou survolé par la souris.

Pour créer une transition, il est possible d'utiliser les propriétés **transition-property**, **transition-duration** et **transition-delay** ou tout simplement utiliser la super-propriété **transition**.

- **transition-property** : permet de définir les propriétés qui vont subir la transition.
- **transition-duration** : permet de définir la durée de la transition (en secondes).
- **transition-delay** : permet de définir la durée à attendre avant le début de la transition (en secondes).
- **transition** : super-propriété qui remplace les trois propriétés ci-dessus dans l'ordre :

transition-property transition-duration transition-delay

```
h1{
  border:2px green solid;
  font-size:12px;
  width:50px;
  color:#5C6AC4;
  transition-property:color transform;
  transition-duration:3s;
  transition-delay:2s;
}
h1:hover{
  color:blue;
  transform:rotate(120deg);
}
```

=

```
h1{
  border:2px green solid;
  font-size:12px;
  width:50px;
  color:#5C6AC4;
  transition:color 3s 2s, transform 3s 2s;
}
h1:hover{
  color:blue;
  transform:rotate(120deg);
}
```

Les deux transitions ci-dessus font la même tâche : lors du survole de la souris sur l'élément **<h1>**, ce dernier attend 2 secondes ensuite son couleur change doucement de la couleur #5C6AC4 à la couleur red et il subit une rotation de 120 degrés. Ces changements durent 3 secondes.

Attention :

Si on veut effectuer la transition lorsque l'élément est **cliqué** par la souris, il suffit de remplacer **hover** par **active**.

Si la valeur de **transition-delay** est omis, la transition démarre directement sans durée d'attente.

Une transition est parfaite si elle se lance doucement ensuite elle revient à l'état initial doucement aussi.

Il est possible de transcrire la transition dans la partie **hover** (ou **active**), mais dans ce cas la transition ne sera pas parfaite : elle commencera doucement mais le retour à l'état initial se fera brutalement.

Avec la super-propriété transition, puisque chaque propriété a sa propre durée de transition et sa propre durée d'attente. Donc les propriétés changent indépendamment les uns des autres. Par exemple :

```
transition:color 2s 1s, transform 4s 2s;
```

Les animations `animation`

Une animation permet à un élément de changer progressivement d'un style à un autre. L'animation n'est pas liée à un évènement de la souris comme la transition et elle est plus sophistiquée.

Pour créer une animation, il est possible d'utiliser les propriétés **animation-name**, **animation-duration**, **animation-delay**, **animation-iteration-count** et **animation-direction** ou tout simplement utiliser la super-propriété **animation**.

- **animation-name** : définit le nom de l'animation.
- **animation-duration** : définit la durée de l'animation.
- **animation-delay** : définit la durée d'attente avec le début de l'animation.
- **animation-iteration-count** : définit le nombre de répétition de l'animation(par exemple 5). La valeur **infinite** pour cette propriété désigne que le nombre de répétition est infini.
- **animation-direction** : définit si une animation doit être jouée en avant, en arrière ou en cycles alternés.
  - **normal :** L'animation est jouée normalement (vers l'avant). C'est la valeur par défaut.
  - **reverse :** L'animation est jouée en sens inverse (à l'envers).
  - **alternate :** L'animation est d'abord jouée en avant, puis en arrière.
- **animation** : super-propriété qui remplace les cinq propriétés ci-dessus dans l'ordre :
  animation-name animation-duration animation-delay animation-iteration-count animation-direction

La règle **@keyframes** permet de définir les différentes étapes de l'animation. Chaque étape est un ensemble des propriétés prenants des valeurs spécifiques :

- Le plus simple est de mentionner deux étapes seulement ; l'une pour le début (**from**) et l'autre pour la fin (**to**).
- Le plus sophistiqué est de découper l'animation sur plusieurs étapes chacune d'elle est définit par un pourcentage : ainsi on partira de **0%** pour arriver à **100%** et au milieu en ajoutera tant d'étapes qu'on veut.

```css
h1{
  border:2px green solid;
  font-size:12px;
  width:50px;
  color:#5C6AC4;
  animation-name:example;
  animation-duration:3s;
  animation-delay:2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

=

```css
h1{
  border:2px green solid;
  font-size:12px;
  width:50px;
  color:#5C6AC4;
  animation:example 3s 2s infinite alternate;
}
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

Les deux animations ci-dessus font la même tâche : un changement de l'arrière-plan de la couleur rouge à la couleur jaune se fait doucement pendant 3 secondes après une attente de 2 secondes. L'animation se fait doucement dans les deux sens (**alternate**). L'animation se répète infiniment (**infinite**).

Attention :
il est possible de remplacer **from** par **0%** et remplacer **to** par **100%**. Et ajouter au milieu d'autres étapes. Aussi, il est possible d'avoir plusieurs propriétés à changer dans chaque étape.

```css
@keyframes example {
        0%   {background-color:red; left:0px; top:0px;}
        25%  {background-color:yellow; left:200px; top:0px;}
        50%  {background-color:blue; left:200px; top:200px;}
        75%  {background-color:green; left:0px; top:200px;}
        100% {background-color:red; left:0px; top:0px;}
}
```