Memento CSS

- Maj : 23/02/26 -

```
div {color: red; background-color: yellow; } // attributs simples
h1,div,p {color: red; background-color: yellow; } // regroupement
* {background-color: yellow; } // selecteur universel
```

Classes // utilisation du point .

```
.evidence {color: red;}                    <p class="evidence"> blabla </p>
div.jaune {color: yellow;}          // classe "jaune" applicable seulement sur lélément div.jaune
<div classe="classe1 classe2">bla</div>  // attribut : plusieurs classes au même élément.
```

Mise en place dans le Doc HTML

```
<head> :
<style type="text/css" title="monStyle"> .classe1 { color: red; font-family: Arial; } </style>
<link rel="stylesheet" type="text/css" href="monStyleExterne.css">
</head>
<body>
<style type="text/css" title="monStyleBody"> body { color: gray; font-family: Arial; } </style>
...
```

En référence à un id : // utilisation du symbole #

```
div { color: black; }
#bleu { color: blue; }
```

```
<div id="bleu">Blabla texte en bleu</div>
<div>Bliblu Texte en noir</div>
```

Sur un sélecteur d'attribut :

```
h2[title] {background-color: yellow; }
img[ismap] {border-color: red;}
```

On applique le style sur les éléments choisis, à partir du moment où ils possèdent certains attributs.
Dans l'exemple, "title" et "ismap" correspondant respectivement aux balises <h2> et <img>
> si une balise <h2> est déclarée sans attribut "title", elle ne sera pas concernée.

Sélecteur de valeur d'attribut :

```
element [attribut="valeur"] { définition de style }
```

ex :

```
code[title="code javascript"] { color: blue }
td [title="nom"][align="center"] { font-size: 14px }
```

On peut aussi indiquer simplement un attribut , quelque soit sa valeur :

```
[disabled] { cursor : not-allowed ; }
```

Autre exemple plus sophistiqué :

```
td[id |="nom"] { font-style: italic; }          //Toutes les cellules de tableau dont l'attribut id commence par "nom".
```

Encore plus finement joué avec CSS3 :

```
element[attribut ^= "valeur"] { définition du style }        //attribut commence par "valeur "
element[attribut $= "valeur"] { définition du style }        //attribut se termine par "valeur"
element[attribut *= "valeur"] { définition du style }        //attribut contient "valeur"
```

Sélecteur d'attribut modulé :
Attribut dont la valeur est comprise dans une liste séparée par des espaces

```
<a href="http://www.lesite.com/" rel="author external nofollow">En savoir plus sur Moi</a>
```

```
a[rel ~= "author"]{
        color : hotpink ;
}
```

Ici, on cible tous les éléments de type **a** ayant un attribut **rel** contenant le mot **author**.
Si on mettait **a**[**rel** ="author"], on ne ciblerait que les éléments **a** qui ont un unique attribut **rel="author"**

Sélecteurs contextuels parents-descendants :

On sépare les sélecteurs parent et enfant **avec un espace** :
div p{color: red;} // ne concerneront que les paragraphes p enfants de div (descendants indirects)

Selecteurs parent-enfant directs :

```
div > span { color: red; }
```

Utilisation d'un **chevron** (>) - le style est appliqué uniquement si span est enfant direct de div

Selecteurs parent-enfants de Css3 :

element:nth-child(n) { style }	// element est le nième enfant de son parent en commençant par le début.
element: nth-last-child(n) { style }	// element est le nième enfant de son parent en commençant par la fin.
element: first-of-type { style }	// element est le premier enfant de ce type
element: last-child { style }	// dernier enfant de son parent

```
th:first-of-type {color: red; }        /* seul le premier th aura une police rouge */

ul > li:first-child { color : red ; }   /* seul le premier élément de la liste sera rouge */
```

```
.container div:nth-child(2) { background : blue ; }          /* le second enfant de .container s'il s'agit d'une div */

ul > li:nth-child(3n) {  background : blue ; }               /* enfants de ul multiple de 3 si ce sont des li */
```

Feuille de style dans un fichier externe :

Dans l'entête du doc HTML :

```
<link rel="stylesheet" type="text/css" href="./css/monStyle.css" media="screen" title="super style">
```

Dans le fichier monStyle.css :

```
/* texte de présentation */
div {color: red;}
#bleu {text-align: right; }
...
```

FLEXBOX

Attribut flex-direction

- **row**: est la valeur par défaut, c'est elle qui distribue les éléments enfant sur l'axe horizontal dans l'ordre de flux (par défaut, de gauche à droite). Si vous appliquez l'attribut HTML **dir:rtl** alors l'alignement sera effectuée de droite à gauche.
- **column**: dans ce cas la distribution des éléments enfant sera effectuée sur l'axe vertical de haut en bas.
- **row-reverse**: la distribution sera faite suivant l'axe horizontal mais dans l'ordre inversé.
- **column-reverse**: la distribution sera faite suivant l'axe vertical mais dans l'ordre inversé (du bas vers le haut).

Attribut justify-content

- **flex-start**: (valeur par défaut) les éléments enfant seront alignés suivant l'axe principal dans l'ordre normal du flux au début du conteneur.
- **flex-end**: les éléments enfant seront alignés à la fin du conteneur.
- **space-between**: les éléments seront répartis sur tout l'axe principale, si leur largeur combinée est inférieure à celle du conteneur. Un espacement sera placé entre deux enfants voisins d'une manière régulière.
- **space-around**: les éléments seront répartis sur tout l'axe principal. Si leur largeur combinée est inférieure à celle du conteneur alors un espacement sera placé autour de tous les enfants.
- **center**: les éléments seront centrés dans le conteneur.

si : `flex-direction: row;` **justify-content** agit sur l'axe Horizontal **align-items** agit sur l'axe vertical

si: `flex-direction: column;` alors : c'est l'inverse.

Attribut align-content

The CSS **`align-content`** property sets the distribution of space between and around content items along a flexbox**'s cross-axis** or a grid**'s block axis**.

```
align-content: center; /* Pack items around the center */
align-content: start; /* Pack items from the start */
align-content: end; /* Pack items from the end */
align-content: flex-start; /* Pack flex items from the start */
align-content: flex-end; /* Pack flex items from the end */
```

Attribut align-items

The CSS **`align-items`** property sets the value on all direct children as a group.
In Flexbox, it controls the alignment of items on the Cross Axis.

```
align-items: center; /* Pack items around the center */
align-items: start; /* Pack items from the start */
align-items: end; /* Pack items from the end */
align-items: flex-start; /* Pack flex items from the start */
align-items: flex-end; /* Pack flex items from the end */
align-items: self-start; /* Pack flex items from the start */
align-items: self-end; /* Pack flex items from the end */
```

≡ Éléments remplacés Object-fit,...

Éléments remplacés

En CSS, un **élément remplacé** est un élément dont la représentation est en dehors du champ de CSS.
► Elements remplacés caractéristiques : <**img**> <**video**> <**object**> <**iframe**>

La propriété CSS **`object-fit`** définit la façon dont le contenu d'un élément remplacé doit s'adapter à son conteneur en utilisant sa largeur et sa hauteur.

```
object-fit: contain; // le contenu est converti pour garder son aspect (ratio) tout en s'ajustant à l' élément conteneur
object-fit: cover; // contenu redimensionné pour garder le ratio (aspect), tout en remplissant le conteneur
object-fit: fill; // remplissage totatl du conteneur
object-fit: none; // contenu pas redimensionné
object-fit: scale-down; // The content is sized as if none were selected - would result in a smaller concrete object size.
```

Propriété Background-size :

```
/* Valeurs avec un mot-clé */
background-size: cover;
background-size: contain;

/* Une seule valeur                          */
/* La valeur désigne la largeur de l'image. */
/* La hauteur vaut 'auto'                    */
background-size: 50%;
background-size: 3.2em;
background-size: 12px;
background-size: auto;

/* Deux valeurs                             */
/* Première valeur : la largeur de l'image */
/* Seconde valeur : la hauteur de l'image  */
background-size: 50% auto;
background-size: 3em 25%;
background-size: auto 6px;
background-size: auto auto;

/* Valeurs pour plusieurs arrière-plans */
background-size: auto, auto; /* À ne pas confondre avec `auto auto` */
background-size: 50%, 25%, 25%;
background-size: 6px, auto, contain;
```

`contain` : Un mot-clé qui redimensionne l'image afin qu'elle soit la plus grande possible et que l'image conserve ses proportions

`cover` : Un mot-clé dont le comportement est opposé à celui de `contain`. L'image est redimensionnée pour être aussi grande que possible et pour conserver ses proportions

`auto` : un mot-clé qui redimensionne l'image d'arrière-plan afin que ses proportions soient conservées.

`<length>`
Une valeur de type `<length>` qui redimensionne l'image afin que celle-ci occupe la longueur indiquée dans la dimension concernée

`<percentage>`
Une valeur de type `<percentage>` qui redimensionne l'image d'arrière-plan proportionnellement à la taille de la zone dédiée à l'arrière-plan, définie via `background-origin`.

ANIMATIONS

Exemple de rotation après un délai de 10s (*source : openAI chatGPT*) :

```css
/* Define the CSS class */
.rotate-element {
  /* Set the initial rotation angle to 0 */
  transform: rotate(0deg);
  /* Set the transition duration to 1 second for smooth animation */
  transition: transform 1s;
}

/* Use keyframes to define the rotation animation */
@keyframes rotateAnimation {
  /* Set the rotation angle from 0 to 360 degrees */
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

/* Apply the animation to the element with the class after a 10-second delay */
.rotate-element {
  /* Set the animation properties */
  animation-name: rotateAnimation;
  animation-duration: 2s; /* Set the duration of the animation to 2 seconds */
  animation-delay: 10s;   /* Set the delay of the animation to 10 seconds */
  animation-iteration-count: 1; /* Run the animation only once */
  animation-fill-mode: forwards; /* Keep the element at the end state after the
animation */
}
```

Plusieurs animations pour un même élément du DOM :

Ici, on fait appel à deux animations : **whitepulse** et **rotateAnimation**, en les séparant par un virgule.
Les paramètres sont eux aussi séparés par une virgule, le premier étant affecté à la première animation, et ainsi de suite.

```css
#idDiv > img {

            opacity: 1; background-color: white;border: 0px solid white;
            border-radius: 50%;width: 22vw;
            padding: 15px; margin: 15px;

            animation: whitePulse 9s ease-out, rotateAnimation 2s ease-out;
            animation-duration: 9s, 1.5s;
            animation-iteration-count: infinite, 1;
            margin-left: 20%;
            /* margin-left: 20%; */
```

```
}

@keyframes rotateAnimation {
{
   -webkit-transform: rotate(0);
           transform: rotate(0);
           background: #bbbbbb;
       margin-left: 0%;
 }
 {
   -webkit-transform: rotate(30deg);
           transform: rotate(30deg);
           background: #eeeeee;
 }
 {
   -webkit-transform: rotate(-30deg);
           transform: rotate(-30deg);
           background: #eeeeee;
 }
% {
   -webkit-transform: rotate(0deg);
           transform: rotate(0deg);
           background: #eeeeee;
       margin-left: 20%;
 }

@keyframes whitePulse {
        from { background-color: transparent; -webkit-box-shadow: 0 0 9px #333; }
    25% { background-color: transparent; -webkit-box-shadow: 0 0 18px #aaa; }
        to { background-color: transparent; -webkit-box-shadow: 0 0 9px #fff; }
    50% { background-color: #333333; -webkit-box-shadow: 0 0 88px gold;}
        to { background-color: transparent; -webkit-box-shadow: 0 0 9px #ffffff; }
    0% { background-color: transparent; -webkit-box-shadow: 0 0 18px #a25c8f;}
  }
```

GRID LAYOUT

Exemple

========== **HTML PART**

```
<!DOCTYPE html>
<html lang="fr">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" type="text/css" href="css-diy.css">
   <title>Document test</title>
</head>
<body style="background-color: gray; font-size: 3em;">
  <div class="grille">
      <div class="cellheaderG">
         headerG
      </div>
      <div class="cellheaderD">
         headerD
      </div>
      <div class="cellule-2-1">gauche</div>
      <div class="cellule-2-2">centre</div>
      <div class="cellule-2-3">droite</div>
      <div class="cellfooter">cellfooter</div>
  </div>
</body>
</html>
```

========== **CSS PART**

```
.grille {
   width: 90vw;
   height: 90vh;
   margin: auto;
   display: grid;
   grid-template-rows: 1fr 60% 1fr;
   grid-template-columns: 1fr 70% 1fr;
   grid-template-areas:
    "hautGauche haut haut"
    "gauche centre droite"
    "bas bas bas";
}

.cellule-2-1 { grid-area: gauche; background-color: brown; }
.cellule-2-2 { grid-area: centre; background-color: tomato;}
.cellule-2-3 { grid-area: droite; background-color: darkgoldenrod;}

.cellfooter { grid-area: bas; background: yellow; }
.cellheaderG { grid-area: hautGauche; background-color: bisque; }
.cellheaderD { grid-area: haut; background-color: blueviolet;}
```

Propriété "grid-auto-flow"

```
<!DOCTYPE html>
<html>
<head>
  <style>
.item3 { grid-column: auto / span 2; }

.grid-container {
  display: grid;
  grid-template-columns: auto auto 25%;
  grid-template-rows: auto auto;
  grid-gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}
.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
.orange {
  background-color: orange;
}
  </style>
</head>
<body>

<h1>The grid-auto-flow Property</h1>
<p>The <em>grid-auto-flow</em> property controls how auto-placed items are inserted in the grid.</p>
<p>This grid has three columns and two rows.</p>

<p>"item3" spans two columns, where will the other items be inserted?.</p>
<h2>grid-auto-flow: row</h2>
<div class="grid-container" style="grid-auto-flow: row;">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
</div>
<h2>grid-auto-flow: column</h2>
<div class="grid-container orange" style="grid-auto-flow: column;">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
</div>
<h2>grid-auto-flow: row dense</h2>
<p>The "dense" value fills holes in the grid:</p>
<div class="grid-container" style="grid-auto-flow: row dense;">
  <div class="item1">1</div>
  <div class="item2">2</div>
  <div class="item3">3</div>
  <div class="item4">4</div>
</div>

</body>
</html>
```

The grid-auto-flow Property

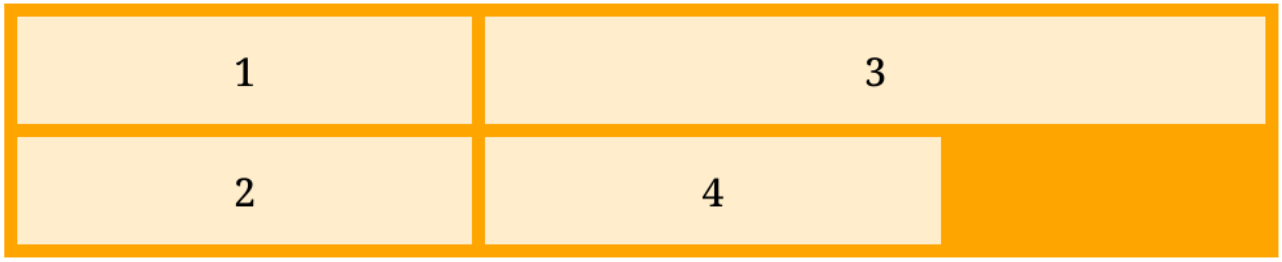
The *grid-auto-flow* property controls how auto-placed items are inserted in the grid.

This grid has three columns and two rows.

"item3" spans two columns, where will the other items be inserted?.

grid-auto-flow: row

grid-auto-flow: column



grid-auto-flow: row dense

The "dense" value fills holes in the grid: