

CSS

APPRENDRE À CODER EN CSS

Marine ABADI / UTOPIOS / M2I Formation



Notions de bases du CSS



■ C'est quoi ?

- ➔ CSS signifie Cascading StyleSheet (feuille de style en cascade)
- ➔ Il permet de mettre en forme et d'ajouter des styles à nos pages HTML
- ➔ Actuellement, c'est la version 3 de CSS qui est utilisée.

La syntaxe en CSS

■ La syntaxe CSS est composée de 3 éléments :

- **Le sélecteur** : l'élément sur lequel on applique la mise en forme souhaitée
- **La propriété** : l'effet que l'on va vouloir donner (couleur, taille du texte, police de caractères, ...)
- **La valeur** de la propriété CSS (noir, 20px, ...)



Lier le HTML et le CSS

Ecrire le CSS dans le code HTML

→ Dans l'élément `style`, incluse dans l'élément `head`

```
1  head>
2    <title>Ecrire le CSS dans le HTML</title>
3    <meta charset= "utf-8">
4
5    <style>
6      p{
7        color: noir;
8        border: 2px solid rgb(255, 0, 106);
9      }
10   </style>
11   </head>
12
13  <body>
14    <h1>Un titre de niveau 1</h1>
15    <p>Un paragraphe</p>
16    <p>Un deuxième paragraphe</p>
17  </body>
```

Lier le HTML et le CSS

Ecrire le CSS dans le code HTML

→ Dans des attributs `style`

```
94
95  <body style="background-color: red;">
96  |  <h1>Un titre </h1>
97  |  <p style="color: blue; font-size: 16px;">Un paragraphe</p>
98  |  <p>Un deuxième paragraphe</p>
99  </body>
100
```

Un titre

Un paragraphe

Un deuxième paragraphe

Lier le HTML et le CSS

■ Ecrire le CSS dans un fichier .CSS

- ➔ Il nous faut un fichier `.html` et un fichier `.css`
- ➔ Dans le fichier `.html` on va utiliser un élément HTML `link`.
- ➔ On va placer l'élément `link` au sein de l'élément `head` de notre fichier HTML.
- ➔ C'est une balise orpheline à laquelle on ajoute un attribut `rel` et un attribut `href`

```
<!DOCTYPE html>
<html lang="en">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css.css">
    <title>Marine ABADI - CV</title>
  </head>
```

Commenter / Indenter

Pour une meilleure lisibilité

- On **commente** le code à l'extérieur des accolades
- On **commente** une déclaration CSS (pour la « désactiver »)
- On va aussi **indenter** le texte dans un souci de clarté.

```
/*Un commentaire CSS*/
p{
    color: noir;
    border: 2px solid #rgb(255, 0, 106);
}
/*Un commentaire CSS
*des commentaires*/

p{
    /*color: noir;*/
    border: 2px solid #rgb(255, 0, 106);
}
```

Les sélecteurs CSS

Sélecteurs CSS simple

■ Les sélecteurs d'éléments

- On les appelle aussi **sélecteurs simples** car il permettent de sélectionner directement un élément
- Par exemple, p, h1, h2, h3, a etc....
- Ils consomment moins de ressources, mais nous limitent dans nos possibilités.

```
/*Un commentaire CSS*/  
p{  
    color: noir;  
    border: 2px solid #rgb(255, 0, 106);  
}  
/*Un commentaire CSS  
*des commentaires*/
```

Sélecteurs CSS class & id

Comment les utiliser ?

```
marineABADI.html  ◊ test.html  # css.css
s > marineabadi > Desktop > ◊ test.html > html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="css.css">
  <title>Document</title>
</head>
<body>
  <h1 id="bleu"> Un titre </h1>
  <p class="para">Un paragraphe</p>
  <p>Un deuxième paragraphe</p>
</body>
</html>
```

```
MarineABADI.html  ◊ test.html  # css.css
Users > marineabadi > Desktop > # css.css > .para
1  /*Un commentaire CSS*/
2
3  #bleu
4  {
5    color: blue;
6  }
7  .para
8  [
9    color: blueviolet;
10 ]
```

Un titre

Un paragraphe

Un deuxième paragraphe

→ Les attributs **#id** ont la priorité sur les attributs **.class**

→ Un même élément HTML va pouvoir recevoir différents attributs **.class**

Sélecteurs CSS class & id

A quoi ça sert ?

- Les attributs `class` et `id` sont spécifiquement utilisés pour attribuer un style et non pas pour préciser un fonctionnement en HTML
- On va les ajouter dans n'importe quelle balise ouvrante d'un élément HTML
- Nous allons déjà devoir renseigner une valeur pour chaque attribut `.class` et `#id`.
- Les valeurs ne doivent pas indiquer d'espace, de caractères spéciaux et toujours commencer par une lettre
- Chaque `id` doit être utilisé une fois, les `class` peuvent être attribuer à plusieurs endroits de la page

Sélecteurs CSS universel

■ Le sélecteur CSS universel / étoile

- Le sélecteur étoile (*) permet de sélectionner tous les éléments d'une page
- Utile pour définir une même police de caractère

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

Sélecteurs CSS

■ Les sélecteurs CSS séparés par une virgule

- La [virgule \(,\)](#) va permettre de lister des sélecteurs séparés par une virgule
- Le style va être appliqué ici au [h1](#) et au [p](#)
- Permet de gagner du temps

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

Sélecteurs CSS combinateurs

■ Les sélecteurs à la suite

- Le combinateur (espace) permet de sélectionner les nœuds qui sont des descendants d'un élément donné.
- Le style va être appliqué aux éléments `a` contenu dans les éléments `p`

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

Sélecteurs CSS combinateurs

Sélecteurs d'éléments fils

- Le combinateur '>' permet de sélectionner les nœuds qui sont des fils directs d'un élément donné.
- `ul > li` permettra de cibler tous les éléments `` qui sont directement situés sous un élément ``

```
6
7
8  <div id="container">
9    <ul>
10      <li> List Item
11        <ul>
12          <li> Child </li>
13        </ul>
14      </li>
15      <li> List Item </li>
16      <li> List Item </li>
17      <li> List Item </li>
18    </ul>
19  </div>
```

Sélecteurs CSS combinateurs

Sélecteurs voisin direct

- Le combinateur `+` permet de sélectionner les nœuds qui suivent immédiatement un élément donné.
- `div + p` permettra de cibler n'importe quel élément `<p>` qui suit immédiatement un élément `<div>`

```
2
3
4  div + p {
5      color: red;
6  }
```

Sélecteurs CSS combinateurs

■ Sélecteurs voisin

- Le combinateur `~` permet de sélectionner les nœuds qui suivent un élément et qui ont le même parent.
- Le caractère `~` va nous être plus permissif que le caractère `+`
- `p ~ span` permettra de cibler les éléments `` qui suivent (immédiatement ou non) un élément `<p>` et qui ont le même élément parent.

```
3  
4  p ~ span {  
5    color: red;  
6 }
```

Sélecteurs CSS combinateurs

Combinateurs de colonnes

- Le combinateur `||` sélectionne les nœuds qui appartiennent à une colonne.
- `col || td` permettra de cibler n'importe quel élément `<td>` sous la portée d'une colonne `<col>`.

```
2
3
4  col || td {
5      color: red;
6  }
```

Sélecteurs CSS pseudo class

■ La pseudo classe :link

- La pseudo-classe `:link` est utilisée pour cibler toutes les balises d'ancres qui ne sont pas encore cliquées.
- la pseudo-classe `:visited` permet, comme vous l'aurez deviné, d'appliquer une mise en forme spécifique aux seules balises d'ancres de la page qui *ont* été cliquées ou *visitées*.

*Pour info
Il manque le i sur
l'imprim ecran*

```
3
4  a:link {
5      |   color: ■red;
6      |
7  a:visited [
8      |   color: ■purple;
9      |
10     ]}
```

Sélecteurs CSS pseudo class

■ Sélecteur d'attribut

→ Dans l'exemple, le sélecteur ne sélectionnera que les balises d'ancres qui ont un attribut title. Cette mise en forme spécifique ne sera donc pas appliquée aux autres balises d'ancres.

```
3  
4  a[title] {  
5      color: green;  
6  }
```

Sélecteurs CSS pseudo class

■ Sélecteur d'attribut avec extrait de code (précis ou non)

→ Dans l'exemple, l'extrait de code appliquera une mise en forme à toutes les balises d'ancres qui contiennent un lien vers <http://utopios.solutions>

Toutes les autres balises d'ancres resteront inchangées.

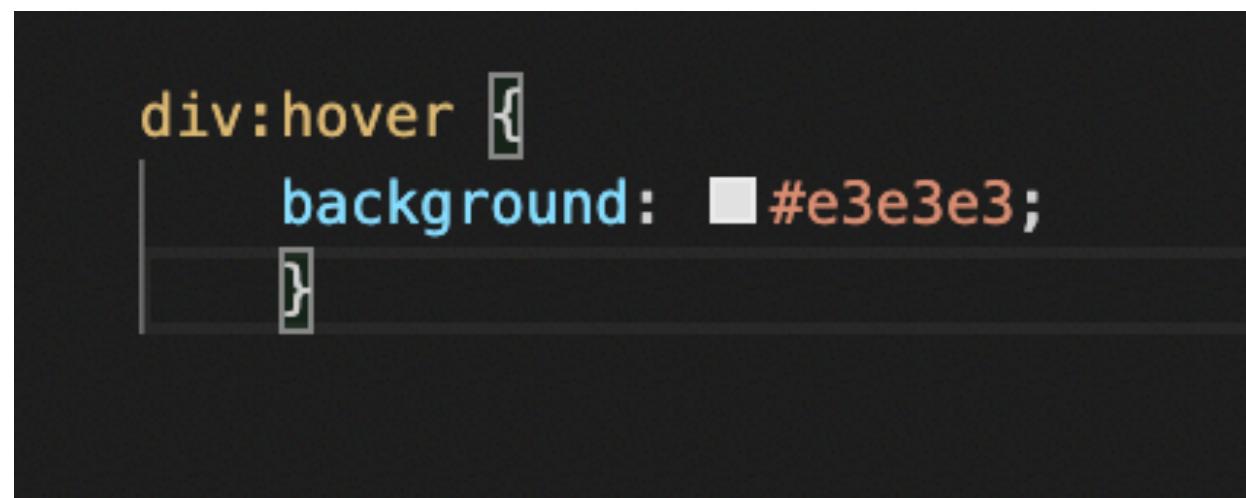
```
2  
3  
4  a[href="http://utopios.solutions"] {  
5      color: #37309e;  
6  }
```

```
2  
3  
4  a[href*="utopios"] {  
5      color: #37309e;  
6  }
```

Sélecteurs CSS pseudo class

■ La pseudo class :hover (action d'utilisation)

- La pseudo class :hover permettra d'appliquer une mise en forme spécifique quand un utilisateur survole un élément



```
div:hover {  
    background: #e3e3e3;  
}
```

A screenshot of a code editor showing a single line of CSS code. The code defines a selector 'div:hover' followed by a pair of curly braces {}, and inside, the 'background' property is set to the hex color '#e3e3e3'. The code is displayed in white text against a dark background.

Sélecteurs CSS pseudo class

■ La pseudo class :checked

- La pseudo class :checked ne ciblera un élément d'interface utilisateur que si il a été coché (*checked*) - comme un bouton radio ou une case à cocher.

```
input[type=radio]:checked {  
    border: 1px solid black;
```

Sélecteurs CSS pseudo class

■ La pseudo class :nth-child(n) ou nth-last-child(n)

→ On se sert de la pseudo class :nth-child(n) pour cibler un élément d'une liste

Ici, sera en rouge le 3ème élément des liste li

```
li:nth-child(3) {  
    color: red;  
}
```

Sélecteurs CSS pseudo class

■ La pseudo class :first-child ou :last-child

→ Cette pseudo-classe permet de cibler uniquement le premier enfant de l'élément parent.

Elle est utilisée notamment pour supprimer les bordures du premier et du dernier élément d'une liste.

```
ul li:first-child {  
    border-top: none;  
}  
  
ul li:last-child {  
    border-bottom: none;  
}
```

Cascade en CSS

■ Lequel a la priorité ?

→ La mot clef `!important` sert à forcer l'application d'une règle CSS.

→ Le degré de précision du sélecteur

1. → L'application d'un attribut `style` dans un élément
2. → `#id`
3. → `.class`
4. → sélecteur d'élément ou pseudo-élément

→ L'ordre d'écriture des règles. C'est la dernière déclaration qui prime sur les déclarations précédentes.

Mise en forme



Mise en forme du texte

Font family

- La propriété CSS font-family
- On met plusieurs polices pour s'assurer que tous les navigateurs puissent les lire
- Si un nom de police qui contient des espaces, il faut mettre des guillemets ou apostrophes.
- On utilise aussi les familles génériques : serif, sans-serif, monospace, cursive, fantasy (web safe font)
- Par ex. Times New Roman et Georgia correspondent à la famille Serif

```
        ,  
        '@font-face' {  
            font-family: "Open Sans";  
            src: url("/fonts/OpenSans-Regular-webfont.woff2") format("woff2"),  
                 url("/fonts/OpenSans-Regular-webfont.woff") format("woff");  
        }  
    }  
}
```

Mise en forme & police

■ Google Font

- ➔ Google Fonts est un service d'hébergement de polices libres de droit
- ➔ On va créer un lien en HTML vers la police Google désirée puis mentionner la police en valeur de notre propriété font-family en CSS
- ➔ La police ne dépend pas du navigateur
- ➔ On va pré-charger le jeu complet de caractères de la police via Google dans le navigateur
- ➔ <https://fonts.googleapis.com/>

Mise en forme & police

■ Les autres propriétés du texte

- ➔ La propriété `font-size` : modifier la taille de la police
- ➔ La propriété `font-weight` : modifier l'épaisseur de la police
- ➔ La propriété `font-style` : modifier l'inclinaison de la police.

Mise en forme & police

■ La propriété font-size

→ Les valeurs mots clés

- **xx-small** : la moitié de celle définie dans le navigateur
- **x-small** : 60% de celle définie dans le navigateur
- **small** : 80% de celle définie dans le navigateur
- **medium** : égale à celle définie dans le navigateur
- **large** : 10% plus grande que celle définie dans le navigateur
- **x-large** : 50% plus grande que celle définie dans le navigateur
- **xx-large** : deux fois plus grande que celle définie dans le navigateur.

- **smaller** : sera plus petite que celle de son élément parent
- **larger** : sera plus grande que celle de son élément parent

Mise en forme & police

■ La propriété font-size

→ Les valeurs longueur

- en pixels `px` (attention mal-voyants)
- en `em` : dynamiques et relatives à la valeur de l'élément parent
- en `rem` : relatives à la taille définie pour la propriété de l'élément racine
- en `%` : proportionnelles à la valeur renseignée pour la propriété dans l'élément parent

Mise en forme & police

■ La propriété font-weight

- ➔ **normal** : valeur par défaut qui correspond à un poids de police « normal »
- ➔ **lighter** qui va définir une police d'écriture plus fine que pour la valeur normal
- ➔ **bold** qui va définir une police d'écriture plus épaisse que pour la valeur normal
- ➔ **bolder** qui va définir une police d'écriture très épaisse

Mise en forme & police

■ La propriété font-style

- ➔ **normal** : valeur par défaut, les caractères seront droits
- ➔ **italic** : la police va s'afficher en italique
- ➔ **oblique** : la police va être tordue pour être rendue de façon oblique.

Mise en forme du texte

■ Les propriétés

- ➔ **text-align** : gérer l'alignement du texte : `left, center, right, justify`
- ➔ **text-transform** : gérer la casse du texte (majuscules ou en minuscules) : `none, lowercase, uppercase, capitalize`
- ➔ **text-decoration** : ajouter des éléments de décoration :
 - `underline, overline, line-through`
 - `Solid, double, dotted, dashed, wavy`

- ➔ **text-indent** : définir l'indentation d'un texte par rapport à l'élément parent : `px, em, %,`
- ➔ **text-shadow** : ajouter des ombres autour d'un texte ex : `text-shadow: 0px 0px 5px red;`

Mise en forme du texte

■ Interlignes et espacements

- ➔ `line-height` : hauteur entre les lignes en `px`, `em`
- ➔ `letter-spacing` : espace entre les lettres en `px`, `em`
- ➔ `word-spacing` : espace entre les mots en `em`

Les couleurs

■ Les différentes possibilités

- ➔ Les valeurs de type « nom de couleur »
 - Au départ 16 couleurs normalisées
 - Maintenant il y en a 140
- ➔ Les valeurs de type [RGB « Red Green Blue »](#)
- ➔ Les valeurs hexadécimales de type [#FF8800](#)
- ➔ On peut aussi rendre un élément HTML plus ou moins transparent grâce à la propriété [opacity](#) ex : [opacity: 0.6](#)

Les types d'affichage



Type d'affichage d'un élément

■ Type block (div) / Type inline (span)

→ Le type d'affichage d'un élément va toujours être défini en CSS par la propriété display.

- `display : block` : affichage sous forme d'un bloc ;
- `display : inline` : affichage en ligne
- `display : inline-block` : un inline va prendre des caractéristiques d'un bloc
- `display : none` : l'élément n'est pas affiché.

Eléments de type inline

Comment ils se comportent ?

- Un élément de **type inline** n'occupe que la largeur nécessaire à l'affichage de son contenu
- Les éléments de **type inline** vont se placer en ligne
- Un élément de **type inline** peut contenir d'autres éléments de type inline mais ne devrait pas contenir d'éléments de type block.
- Ils concernent : em, strong, span, a , button, input, label, textarea, img etc.

Eléments de type block

Comment ils se comportent ?

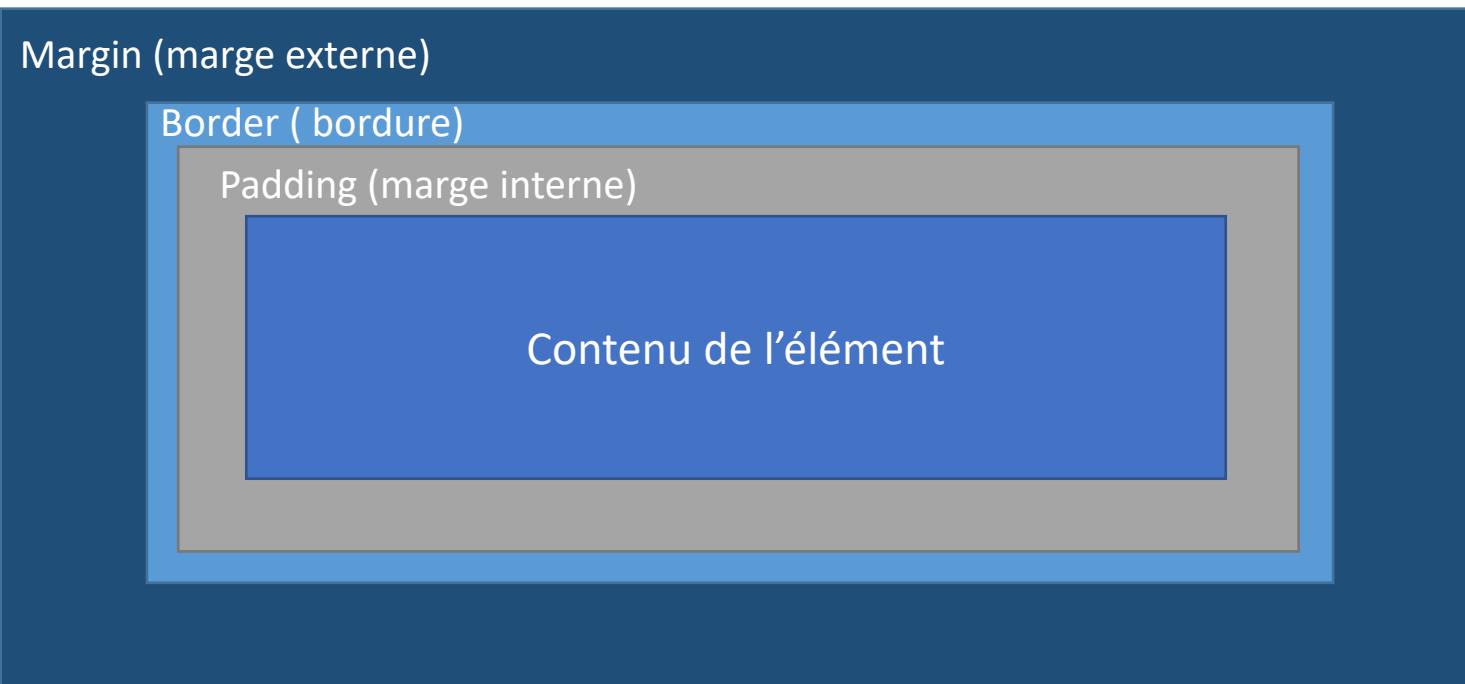
- Un élément de **type block** va prendre la largeur disponible
- Un élément de **type block** va toujours « aller à la ligne », occuper une nouvelle ligne dans une page et ne jamais se positionner à côté d'un autre élément par défaut
- Un élément de **type block** peut contenir d'autres éléments de type block ou de type inline
 - Ils concernant l'élément div, p, h1, h2, ..., article, aside, header, footer, nav et section
 - Les listes ul, ol...
 - Les éléments figure et figcaption...
 - Les éléments video, etc.

Le modèle de boîte

Le modèle de boîte

C'est quoi ?

- Tout élément HTML peut être représenté par un empilement de différentes boîtes rectangulaires



- Les propriétés `width` et `height` : définir la largeur et la hauteur de la boîte « contenu »
- La propriété `padding` : définir la taille des marges internes
- La propriété `border` : définir des bordures pour notre élément
- La propriété `margin` : définir la taille des marges externes

Le modèle de boîte

■ Les propriétés

- ➔ La propriété `display` : définir un type d'affichage pour un élément
- ➔ La propriété `position` : positionner nos éléments de différentes façons dans une page
- ➔ La propriété `float` : faire « flotter » des éléments HTML dans la page

Le modèle de boîte

■ Propriétés width et height

- ➔ Les propriétés `width` et `height` vont pouvoir accepter plusieurs types de valeurs : `px`, `em`, `%`
- ➔ On va éviter de mixer les valeurs
- ➔ Si l'élément dépasse de son parent on peut utiliser la propriété `overflow : hidden`
- ➔ Si le contenu dépasse de l'élément, on peut utiliser la propriété `overflow: scroll`
- ➔ Attention à bien prendre en compte les marges, padding etc...

Le modèle de boîte

■ Propriété padding

- ➔ On peut ajouter des **marges internes** à un élément grâce à la propriété CSS **padding**.
- ➔ L'ajout de marges internes ou padding va augmenter la taille totale de l'élément
- ➔ La propriété **padding** est la version raccourcie de **padding-top**, **padding-left**, **padding-bottom** et **padding-right**
- ➔ Les valeurs vont être en exprimés en **px**, **em**, **%**
- ➔ Attention pas de négatif

Le modèle de boîte

■ Propriété border

- ➔ L'espace pris par la bordure (`border`) va se trouver entre la marge intérieure et la marge extérieure.
- ➔ La propriété `border` est la version raccourcie de
 - `border-width` (`thin`, `medium`, `thick`) en px
 - `border-style` (`solid`, `double`, `dotted`, `dashed`)
 - `border-color`
- ➔ On peut aussi utiliser `border-radius`

Le modèle de boîte

■ Propriété margin

- ➔ On va pouvoir ajouter des marges externes à un élément grâce à la propriété CSS margin
- ➔ C'est la propriété raccourcie de margin-top, margin-left, margin-bottom et margin-right
- ➔ On peut utiliser comme valeur : px, em, % ou auto
- ➔ On peut parfois mettre des valeurs négatives
- ➔ La propriété margin va souvent être utilisée pour centrer horizontalement, on utilise des marges auto des deux côtés, attention : ça ne marchera que pour des éléments de type block

Position et affichage des éléments

Position et affichage

■ Propriété display

→ la propriété CSS **display** affectait la façon dont un élément s'affiche dans une page et comment il va être emboité

Les valeurs possibles :

- block
- inline
- inline-block (un block qui se comporte en inline)
- flex

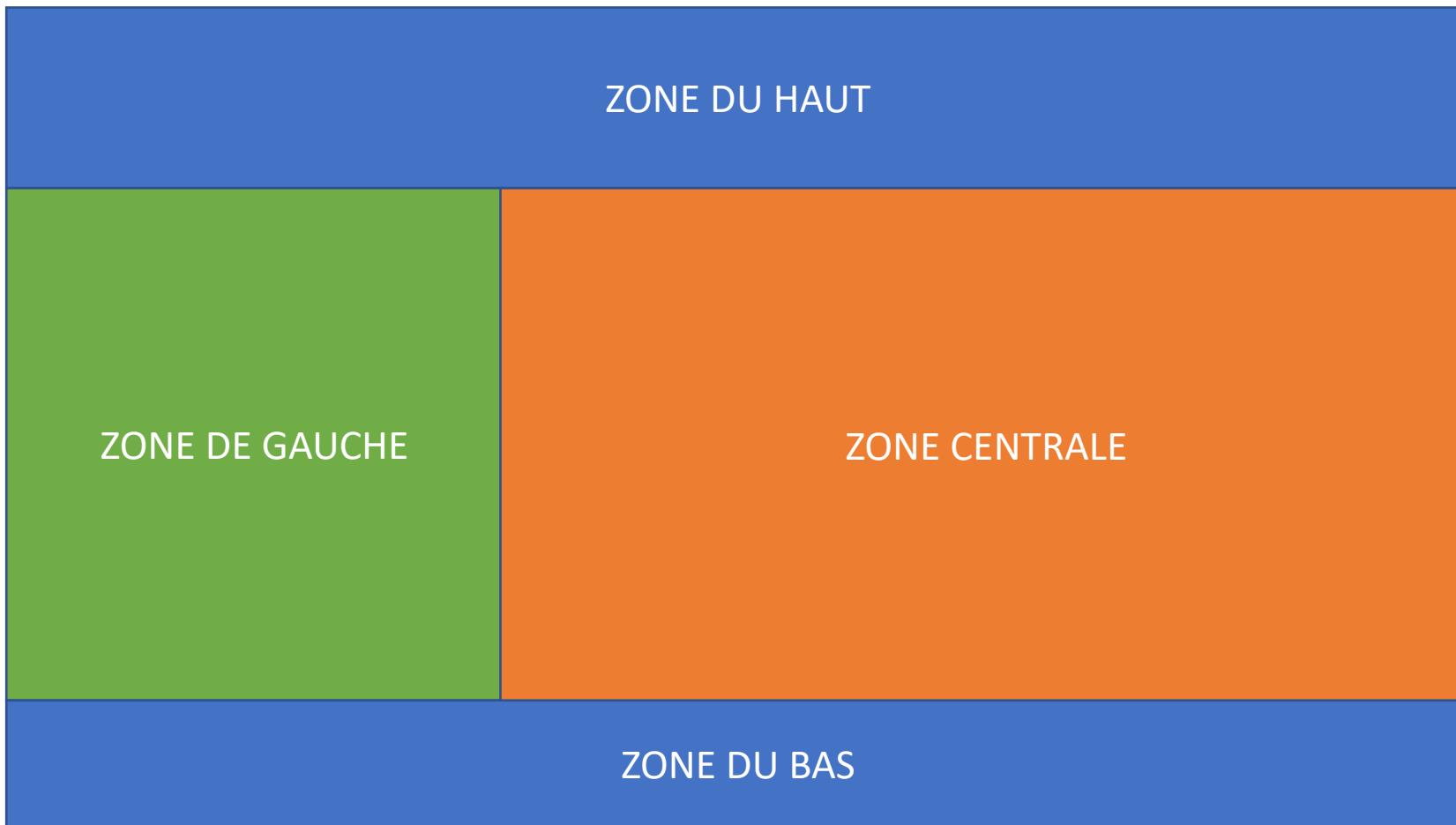
Position et affichage

Propriété float

- ➔ la propriété permet de sortir un élément de type block du flux vertical
 - ➔ la flottaison peut être à droite ou à gauche des autres éléments du flux (left ou right)
 - ➔ pour arrêter la flottaison on utilise la propriété clear qui peut prendre les valeurs : left, right ou both
- ➔ Demo

Position et affichage

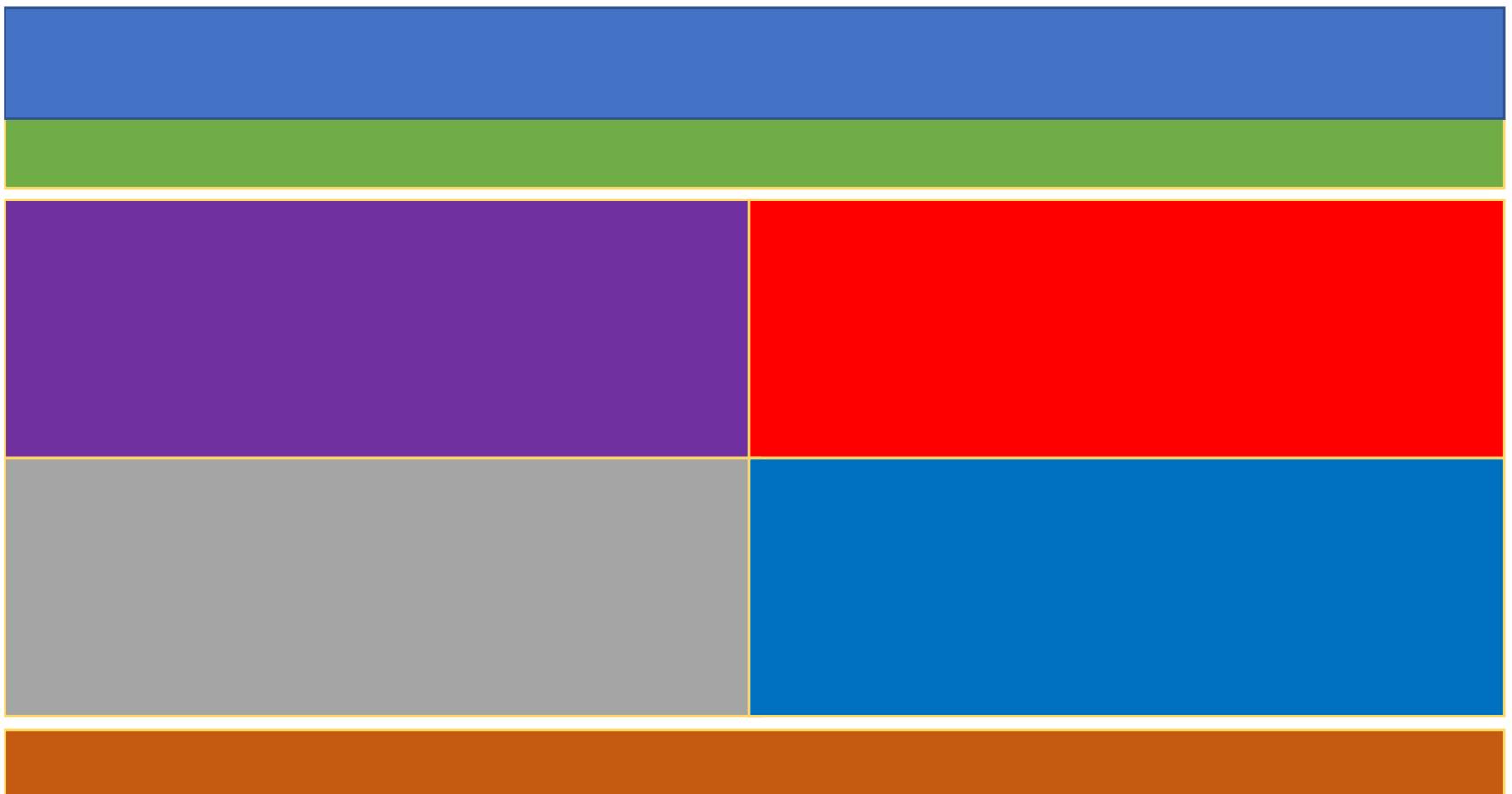
LA CORRECTION EST EN FLOAT



Position et affichage : EXERCICE

LA CORRECTION EST EN FLOAT

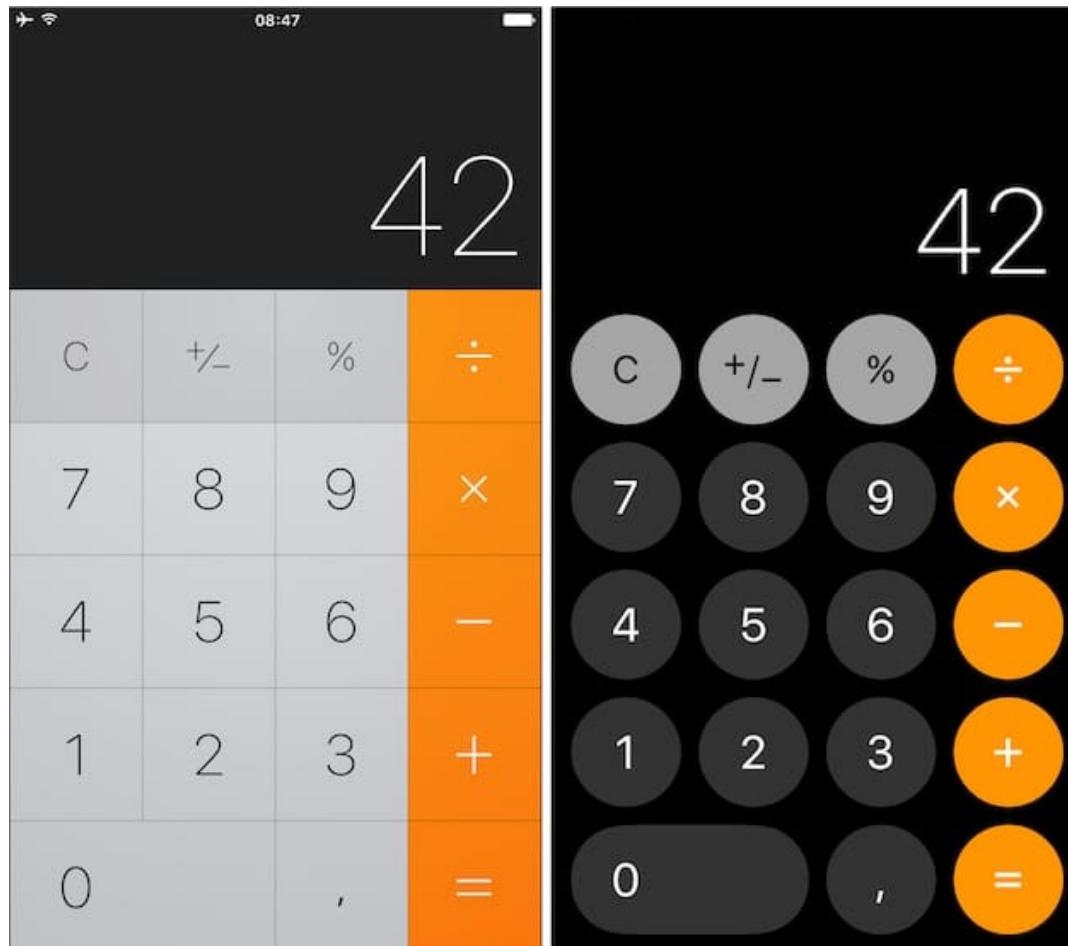
EXERCICE 1



Position et affichage : EXERCICE

LA CORRECTION EST EN FLOAT

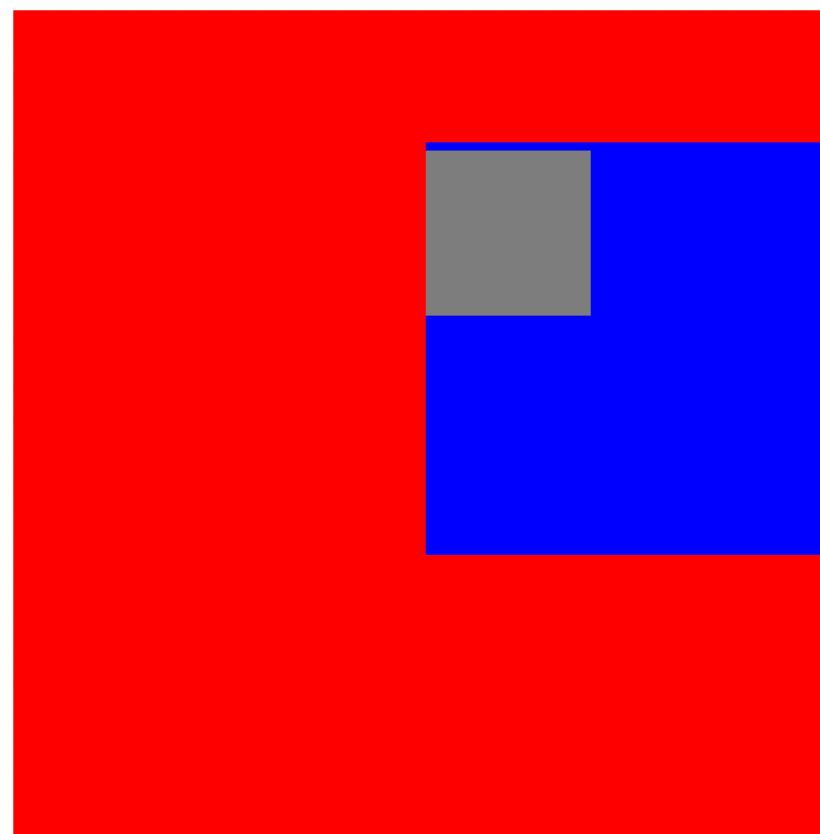
EXERCICE 2



Position et affichage : EXERCICE

EXERCICE 3

Il faut utiliser ici la position relative
et absolue (absolute)



Les boites flexibles : Flexbox

Les boites flexibles : flexbox

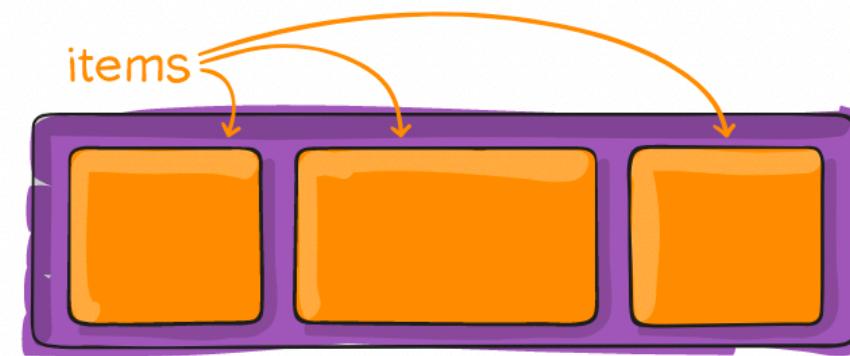
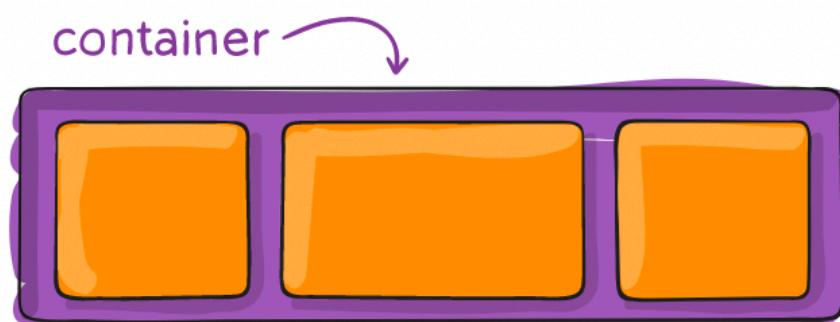
C'est quoi ?

- Flexbox permet de faire des mises en page plus évoluées et plus simplement
- On va pouvoir définir des conteneurs flexibles
- Logiquement flexbox doit remplacer logiquement les « float »
- A été totalement adopté par la W3C seulement en 2017

Les boites flexibles : flexbox

■ Les éléments

- Deux éléments à différencier :
- Le conteneur (appelé flex container)
- Les éléments du conteneur (appelé flex items)



Source : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Les boites flexibles : flexbox

■ display : flex

Ceci définit un conteneur flexible; inline ou block selon la valeur donnée.
Il permet un contexte flexible pour tous ses enfants directs.

```
.container {  
    display: flex; /* or inline-flex */  
}
```

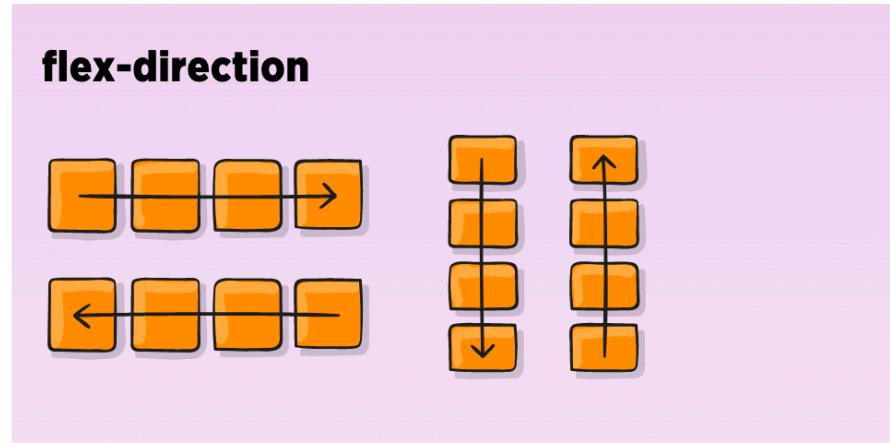
css

Source : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Les boites flexibles : flexbox

flex-direction

- Flex-direction établit l'axe principal, définissant ainsi la direction dans laquelle les éléments flexibles sont placés dans le conteneur flexible.
- Flexbox est (à part l'emballage optionnel) un concept de disposition unidirectionnel. Pensez aux éléments flexibles comme étant principalement disposés en lignes horizontales ou en colonnes verticales.



Les boites flexibles : flexbox

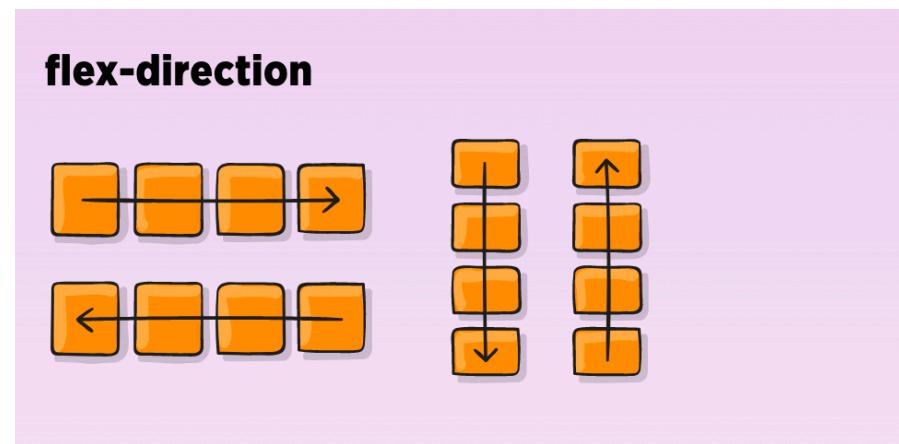
flex-direction

- row (default)
- row-reverse

Si on choisit la valeur row ou row-reverse, l'axe principal sera aligné avec la direction « en ligne » (*inline direction*) (c'est la direction logique)

- column: same as row but top to bottom
- column-reverse: same as row-reverse but bottom to top

Si on choisit la valeur column ou column-reverse, l'axe principal suivra la direction de bloc (*block direction*) et progressera le long de l'axe perpendiculaire au sens d'écriture.

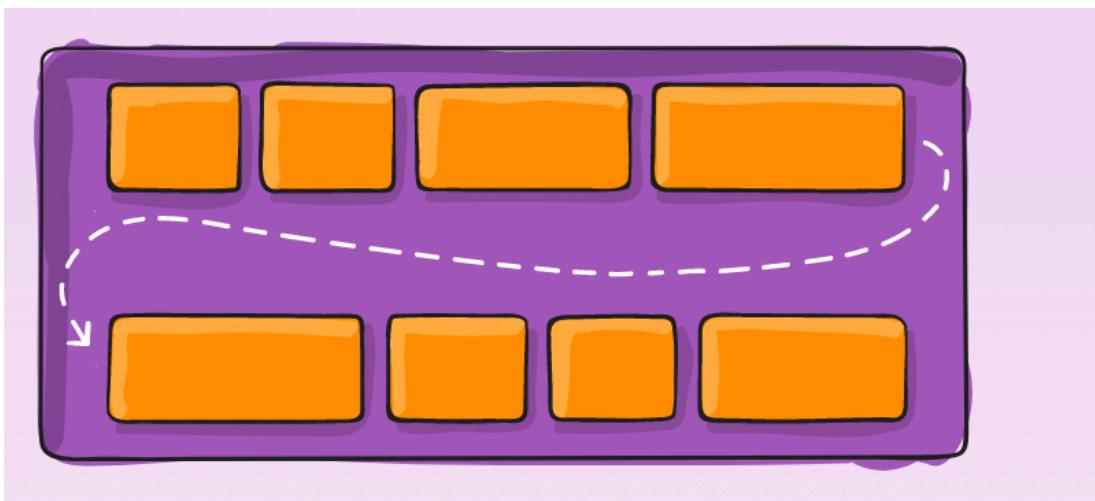


Les boites flexibles : flexbox

flex-wrap

→ Par défaut, les éléments flexibles essaieront tous de tenir sur une seule ligne.

Vous pouvez modifier cela et autoriser les éléments à être enveloppés selon vos besoins avec cette propriété.



```
css  
.container {  
  flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

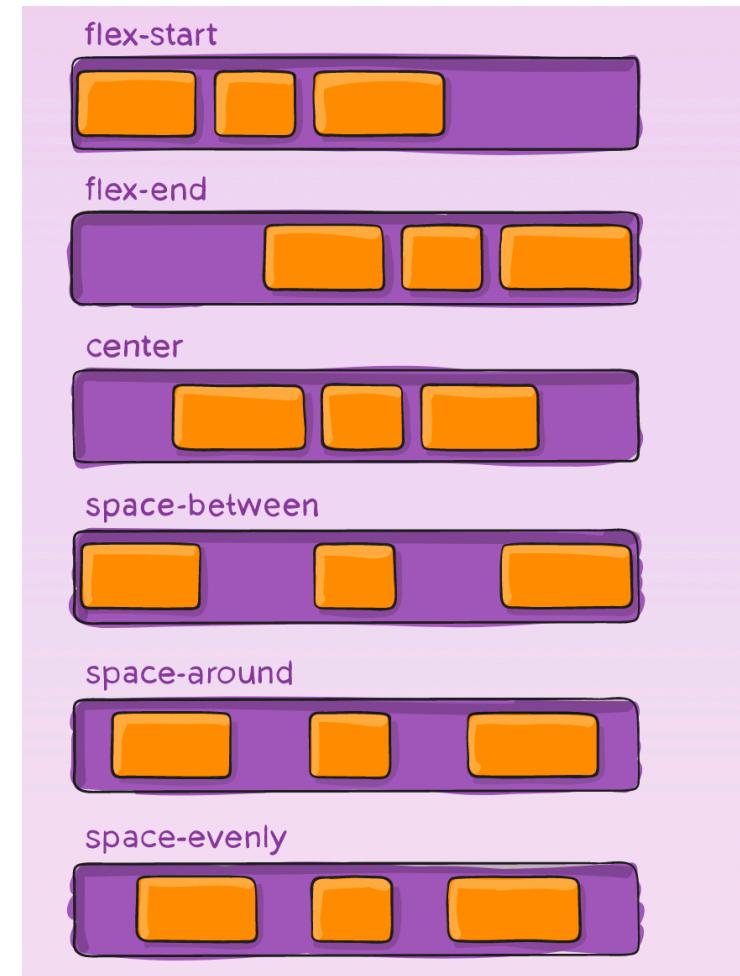
- nowrap (par défaut): tous les éléments flex seront sur une seule ligne
- wrap: les éléments flex seront enroulés sur plusieurs lignes de haut en bas.
- wrap-reverse: les éléments flex seront enroulés sur plusieurs lignes de bas en haut.

Les boites flexibles : flexbox

justify-content

→ Ceci définit l'alignement le long de l'axe principal.
Il aide à répartir l'espace libre supplémentaire restant lorsque tous les éléments flexibles d'une ligne sont rigides ou flexibles mais ont atteint leur taille maximale. Il exerce également un certain contrôle sur l'alignement des éléments lorsqu'ils dépassent la ligne.

```
.container {  
  justify-content: flex-start | flex-end | center |  
}
```

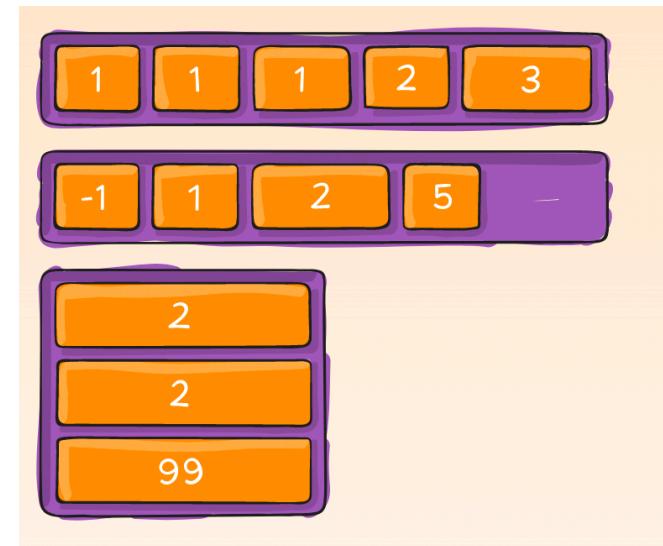


Les boites flexibles : flexbox

order

→ Par défaut, les éléments flexibles sont disposés dans l'ordre source. Cependant, la propriété order contrôle l'ordre dans lequel ils apparaissent dans le conteneur flex.

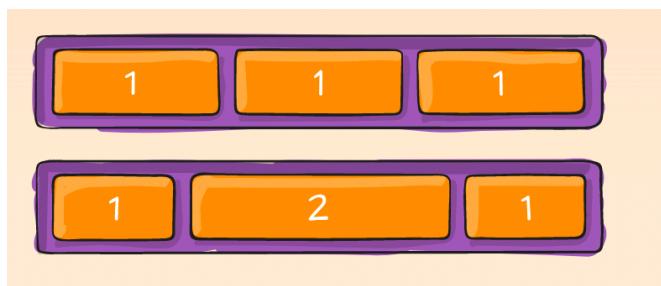
```
css  
.item {  
    order: 5; /* default is 0 */  
}
```



Les boites flexibles : flexbox

flex-grow

- Cela définit la capacité d'un élément flexible à croître si nécessaire. Il accepte une valeur sans unité qui sert de proportion. Il dicte la quantité d'espace disponible à l'intérieur du conteneur flexible que l'élément doit occuper.
- Si flex-grow est défini sur 1 pour tous les éléments, l'espace restant dans le conteneur sera distribué de manière égale à tous les enfants. Si l'un des enfants a une valeur de 2, l'espace restant prendrait deux fois plus d'espace que les autres (ou il essaiera de le faire, au moins).



```
css  
.item {  
  flex-grow: 4; /* default 0 */  
}
```

Les boites flexibles : flexbox

■ flex-shrink

- ➔ Cela définit la capacité d'un élément flexible à se rétrécir si nécessaire.
- ➔ Pas de valeur négative

```
.item {  
  flex-shrink: 3; /* default 1 */  
}
```

css

Bootstrap



Bootstrap

C'est quoi ?

- Bootstrap est un framework CSS.
- Un framework correspond à un ensemble de librairies regroupées dans un but précis et possédant des règles internes que doivent suivre les utilisateurs.
- Bootstrap est un ensemble de fichiers CSS et JavaScript fonctionnant ensemble et qu'on va pouvoir utiliser pour créer des designs complexes de manière relativement simple.
- Ces ensembles de règles sont enfermés dans des classes et nous n'aurons donc qu'à utiliser les classes qui nous intéressent afin d'appliquer un ensemble de styles à tel ou tel élément HTML.

Bootstrap

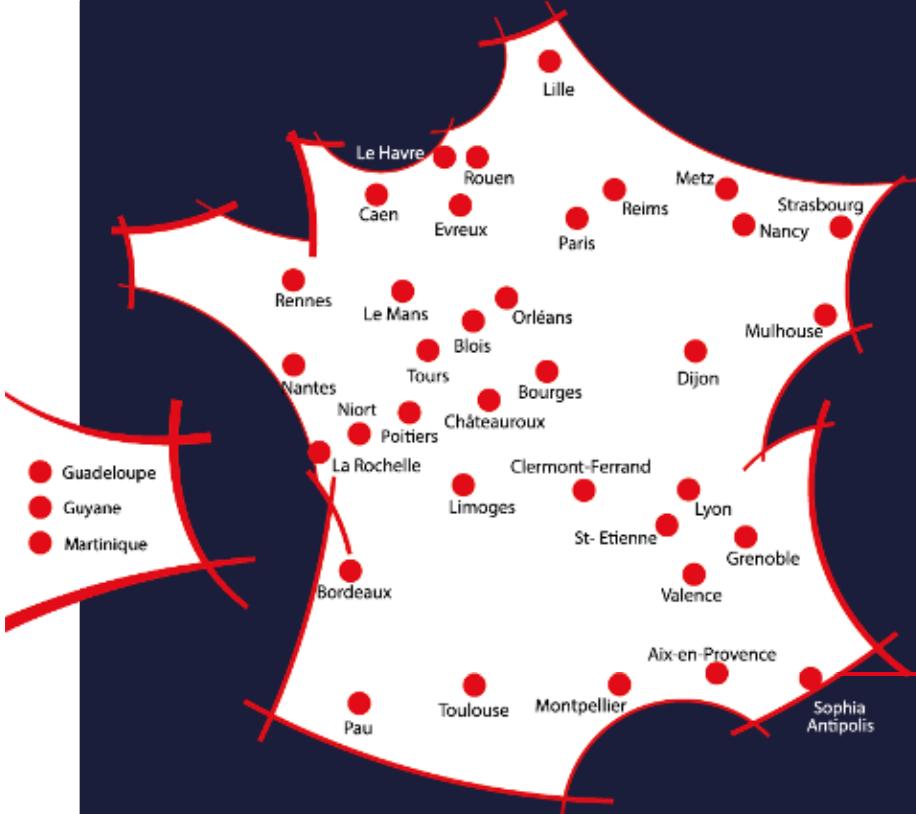
■ Utilisation de Boostrap

- ➔ Gain de temps
 - ➔ Responsive
 - ➔ Compatibilité des navigateurs
 - ➔ Sécurité
 - ➔ Basé sur le modèle des boîtes flexibles
-
- ➔ Prédéfini
 - ➔ On tendance à l'utiliser lorsqu'on sait bien le maîtriser

Bootstrap

■ Utilisation de Boostrap

- ➔ Télécharger les fichiers Bootstrap (CSS et JavaScript) sur le site <https://getbootstrap.com/> puis les lier à nos fichiers HTML comme n'importe quel autre fichier CSS et JavaScript ;
- ➔ Utiliser un CDN (Content Delivery Network ou réseau de distribution de contenu) et inclure le lien vers les fichiers dans nos fichiers HTML.
Il apporte le contenu direct dans notre page



Découvrez également
l'ensemble des stages à votre disposition
sur notre site m2iformation.fr

m2iformation.fr

