VIRGINIE D ~ 2019

FLEXBOX

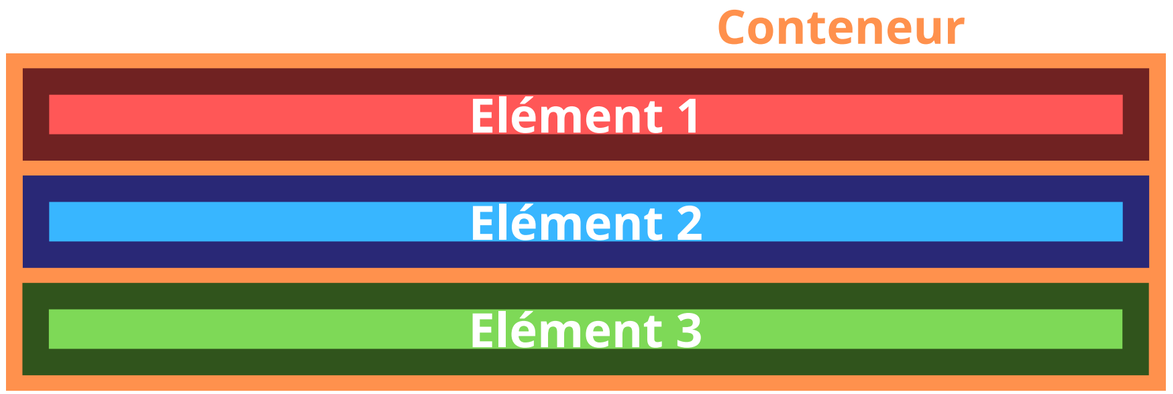
Utilisation

=> Pour utiliser Flexbox, il faut des **éléments** dans un **conteneur**. Ce conteneur sert de "boîte" dans laquelle on range les éléments.

Exemple :

```html
<div id="conteneur">
    <div class="element">Elément 1</div>
    <div class="element">Elément 2</div>
    <div class="element">Elément 3</div>
</div>
```

Résultat par défaut :



=> Par défaut, une est considérée comme un bloc. Par défaut, les blocs ne s'alignent pas. Elles se placeront donc les unes en dessous des autres. C'est à cela que Flexbox va remédier.

RAPPEL

Un bloc utilise 100% de la largeur de son conteneur (élément parent). Pour que flexbox fonctionne, il faut donc attribuer aux éléments une **largeur**. On utilise alors la propriété CSS **width**.

Exemple:

HTML

```html
<div id="conteneur">
  <div class="element1">Mon 1er élément</div>
  <div class="element2">Mon 2e élément</div>
  <div class="element3">Mon 3e  élément</div>
</div>
```

CSS

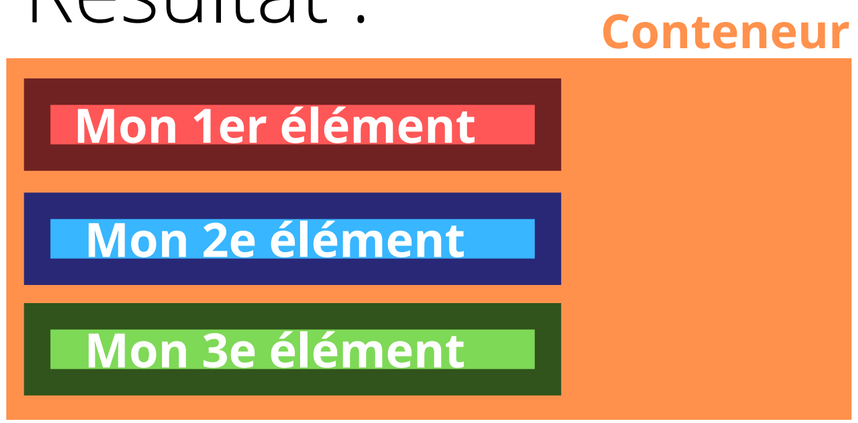
```css
#conteneur {
  background-color: orange;
  height: 90px;
}

.element1 {
  background-color: red;
  width: 150px;
}

.element2 {
  background-color: blue;
  width: 150px;
}

.element3 {
  background-color: green;
  width: 150px;
}
```

Résultat :



RAPPEL

width : propriété css qui définit la **largeur** d'un élément
height : propriété css qui définit la **hauteur** d'un élément

Par défaut, l'élément prendra les dimensions de son contenu

=> Attention à ne pas abuser des propriétés width & height. En effet, un contenu doit pouvoir s'adapter aux différents écrans. En "forçant" ces propriétés, vous pourriez avoir de mauvaises surprises. N'utilisez celles-ci que lorsque c'est nécessaire & prévoyez leurs adaptation.

Propriétés flexbox

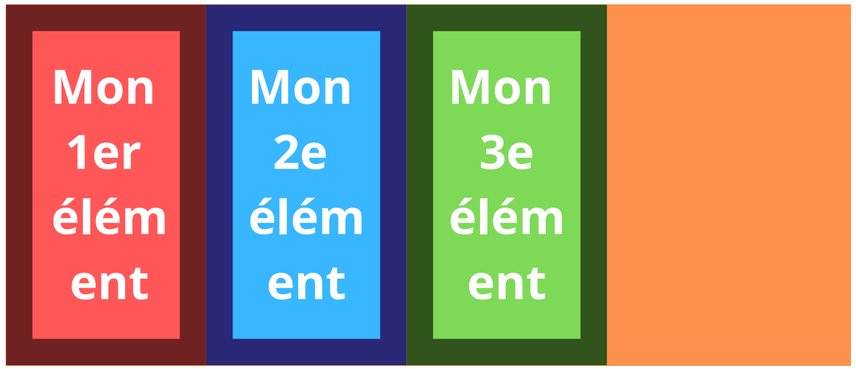
display : flex

=> Une fois nos éléments prêts, on utilise la propriété css **display**, avec l'attribut **flex** directement **sur le conteneur de nos éléments**. Nos éléments s'alignent alors en prenant toute la hauteur de leur conteneur.

CSS :

```css
#conteneur {
  background-color: orange;
  height: 90px;
  display:flex;
}
```

Résultat :

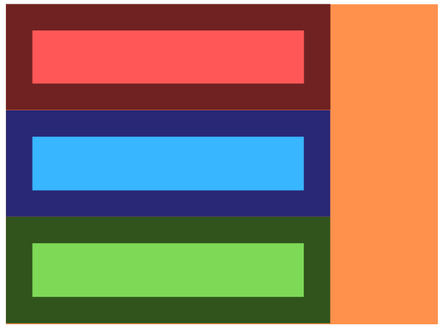


flex-direction

=> Il existe quatre propriétés permettant d'orienter nos éléments.

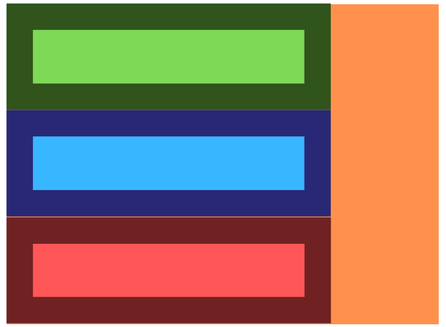
column

```
#conteneur {
  display:flex;
  flex-direction: column
}
```



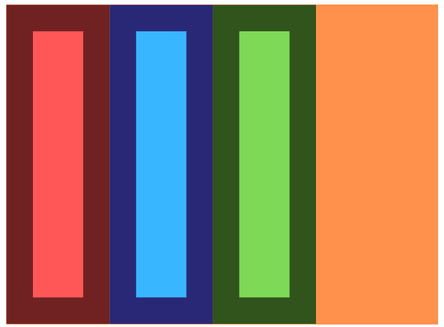
column-reverse

```
conteneur {
 display:flex;
 flex-direction: column-revers
```



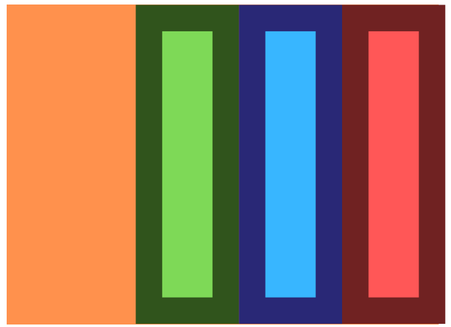
row

```
#conteneur {
  display:flex;
  flex-direction: row
}
```



row-reverse

```
#conteneur {
  display:flex;
  flex-direction: row-reverse;
}
```



flex-wrap

=> L'attribut wrap est utilisé pour le retour à la ligne lorsque les éléments atteignent le bord du conteneur. On peut l'utiliser de deux façons différentes.

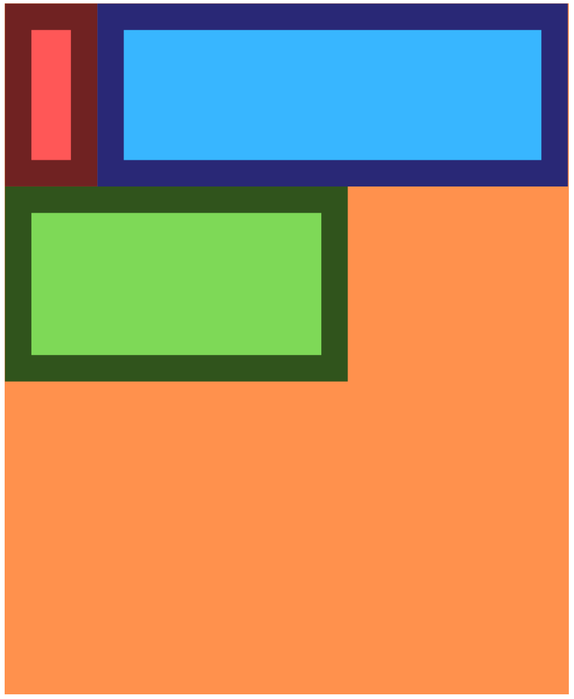
wrap

```
#conteneur {
  display: flex;
  flex-wrap: wrap;
}


.element1 {
  background-color: red;
  width: 10%;
}

.element2 {
  background-color: blue;
  width: 90%;
}

.element3 {
  background-color: green;
  width: 60%;
}
```



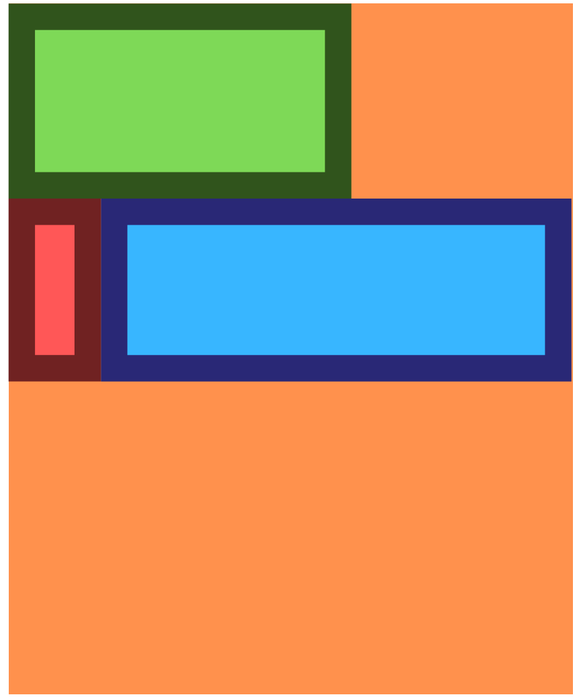
wrap-reverse

```
#conteneur {
  display: flex;
  flex-wrap: wrap-reverse;
}


.element1 {
  background-color: red;
  width: 10%;
}

.element2 {
  background-color: blue;
  width: 90%;
}

.element3 {
  background-color: green;
  width: 60%;
}
```

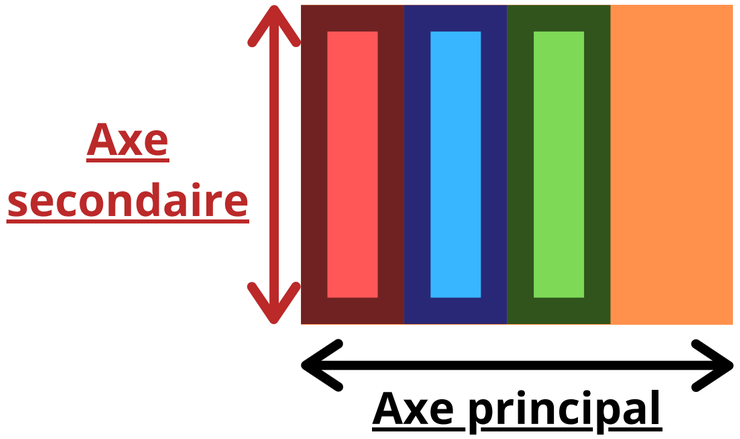


=> Il existe aussi l'attribut **nowrap** ; dans ce cas, les éléments se resserreront autant qu'ils le peuvent

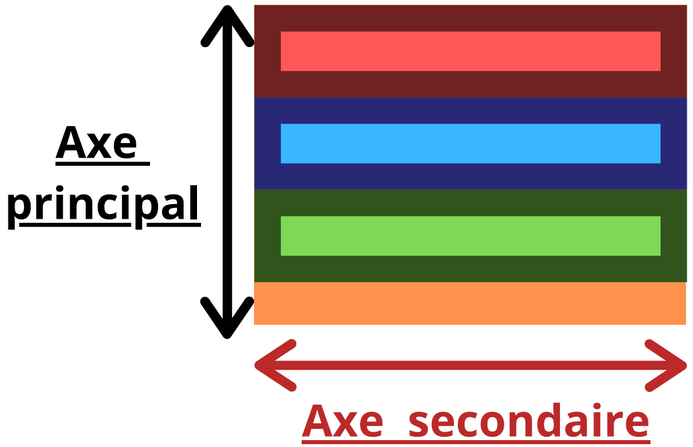
Alignements

=> Les éléments sont organisés soit horizontalement (par défaut), soit verticalement. Cela définit ce qu'on appelle **l'axe principal**. Le second axe devient **l'axe secondaire**.

Si vos éléments sont organisés **horizontalement**, l'axe **secondaire** est l'axe **vertical**.

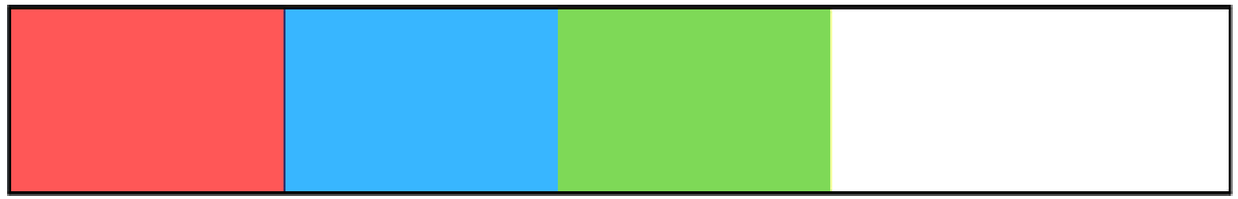


Si vos éléments sont organisés **verticalement**, l'axe **secondaire** est l'axe **horizontal**.

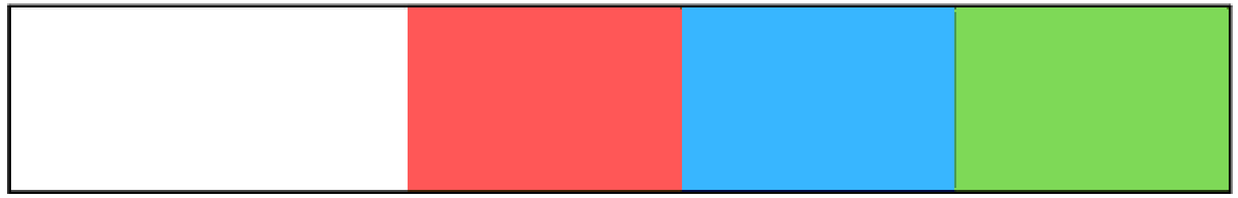


justify-content

=> On utilise la propriété justify-content pour jouer sur l'alignement des éléments. On peut l'utiliser de cinq façons différentes. **Cela fonctionne également si votre axe principal est vertical**



justify-content : flex-start;



justify-content : flex-end;

justify-content : center;

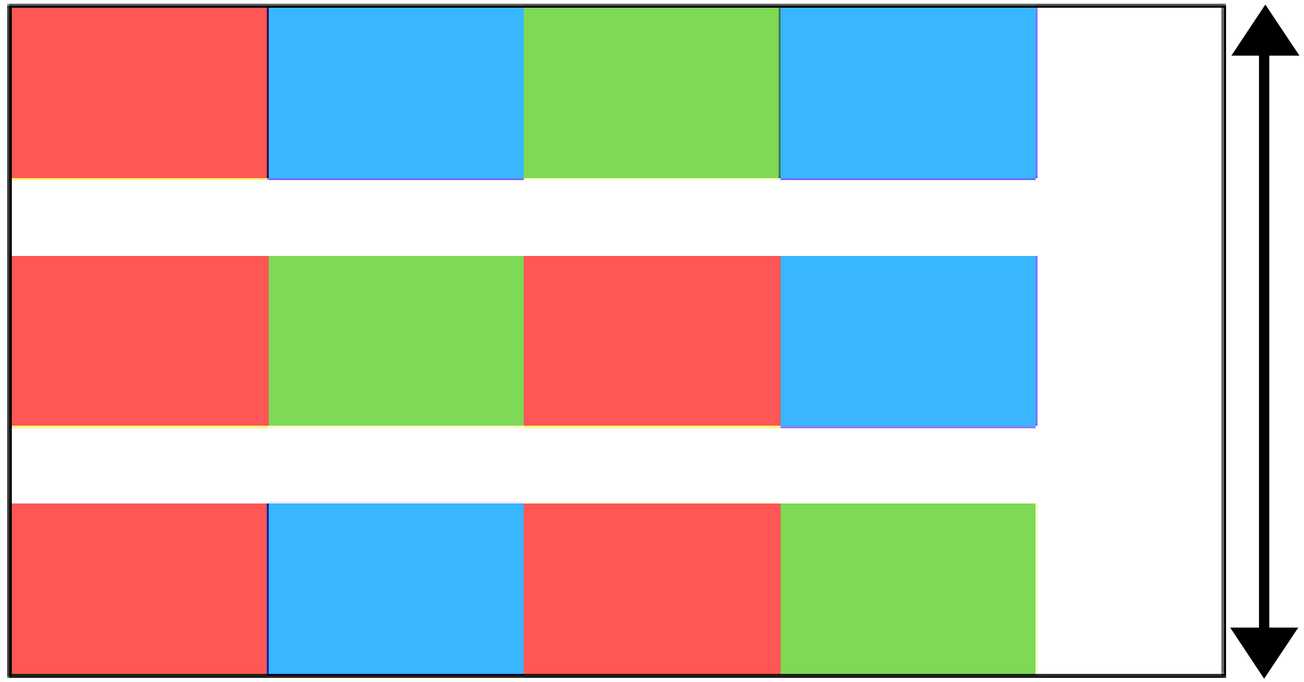
justify-content : space-between;

justify-content : space-around;

align-items

=> Si on a plusieurs retours à la ligne, on peut également ordonner les éléments de l'axe secondaire. On utilise alors la propriété **align-items**. Ce sont les mêmes attributs que pour justify-content, mais cela s'appliquera sur l'axe secondaire.

Exemple :



align-items : space-between;

Ici, le space-between s'applique
verticalement,
les élements restent horizontalement collés.

order

=> On utilise la propriété **order** pour réorganiser les éléments sans avoir à modifier le code HTML !

Exemple :

```css
.element1 {
  background-color: red;
  order: 2;
}

.element2 {
  background-color: blue;
  order: 1;
}

.element3 {
  background-color: green;
  order: 3;
}
```