Flexbox

Your new BFF

Au commencement:

Le principe de Flexbox est assez simple : On va définir un conteneur à l'intérieur duquel on placera plusieurs éléments.

Visualisez une boîte dans laquelle vous allez ranger un tas de trucs.

Sur une même page web, il est tout à fait possible d’avoir plusieurs conteneurs.

Libre à vous d'en créer autant que vous voulez pour obtenir la mise en page désirée.

Un conteneur ?

Le conteneur est une balise HTML, et les éléments sont d'autres balises HTML à l'intérieur de celle-ci.

Exemple:

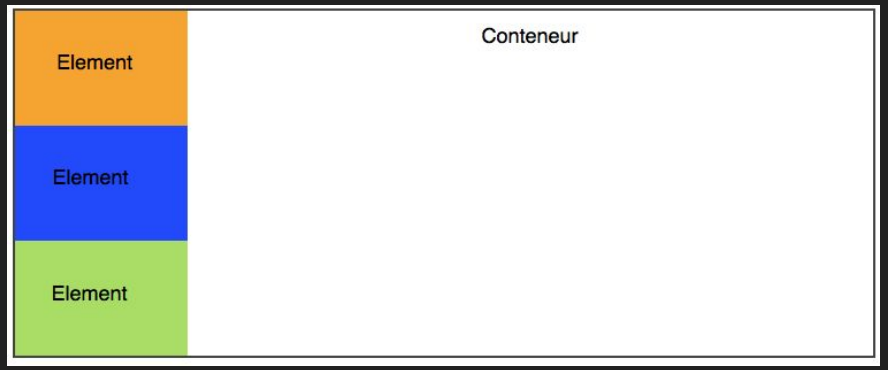
```
<div id="conteneur">

        <div class="element">Elément 1</div>

        <div class="element">Elément 2</div>

        <div class="element">Elément 3</div>

</div>
```



Ok, Flexboxons. Avec une seule propriété CSS, tout va changer.

Exemple:

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

Ce qui donnera:

Donnons du sens:

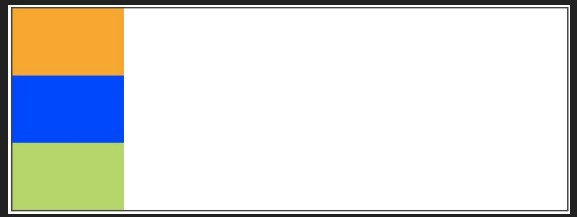
Flexbox permet d'agencer ces éléments dans le sens que l'on veut.

Avec flex-direction, on peut les positionner verticalement ou encore les inverser. Il peut prendre les valeurs suivantes :

- row : organisés sur une ligne (par défaut)
- column : organisés sur une colonne
- row-reverse : organisés sur une ligne, mais en ordre inversé
- column-reverse : organisés sur une colonne, mais en ordre inversé

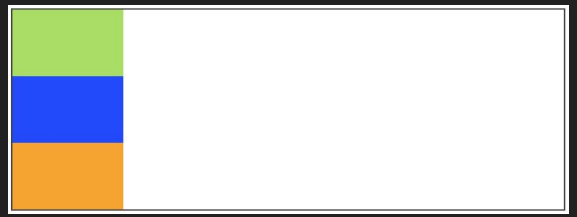
Exemple:

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



Exemple:

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



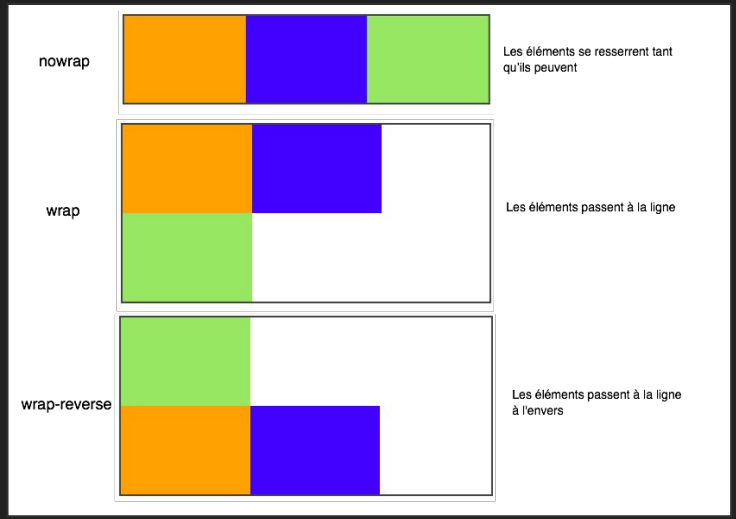
Retour à la ligne:

Par défaut, les blocs tentent de rester sur la même ligne s'ils n'ont pas la place. Si vous voulez, vous pouvez demander à ce que les blocs passent à la ligne lorsqu'ils n'ont plus la place avec flex-wrap :

- nowrap : pas de retour à la ligne (par défaut)
- wrap : les éléments vont à la ligne lorsqu'il n'y a plus la place
- wrap-reverse : les éléments vont à la ligne lorsqu'il n'y a plus la place en sens inverse

Exemple:

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



Aligner ? Aligner !

Les éléments sont organisés soit horizontalement (par défaut), soit verticalement. Cela définit ce qu'on appelle **l'axe principal**. Il y a aussi un axe secondaire (cross axis) :

- 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.

Ok donc maintenant voyons comment aligner nos éléments sur l'axe principal *et* sur l'axe secondaire.

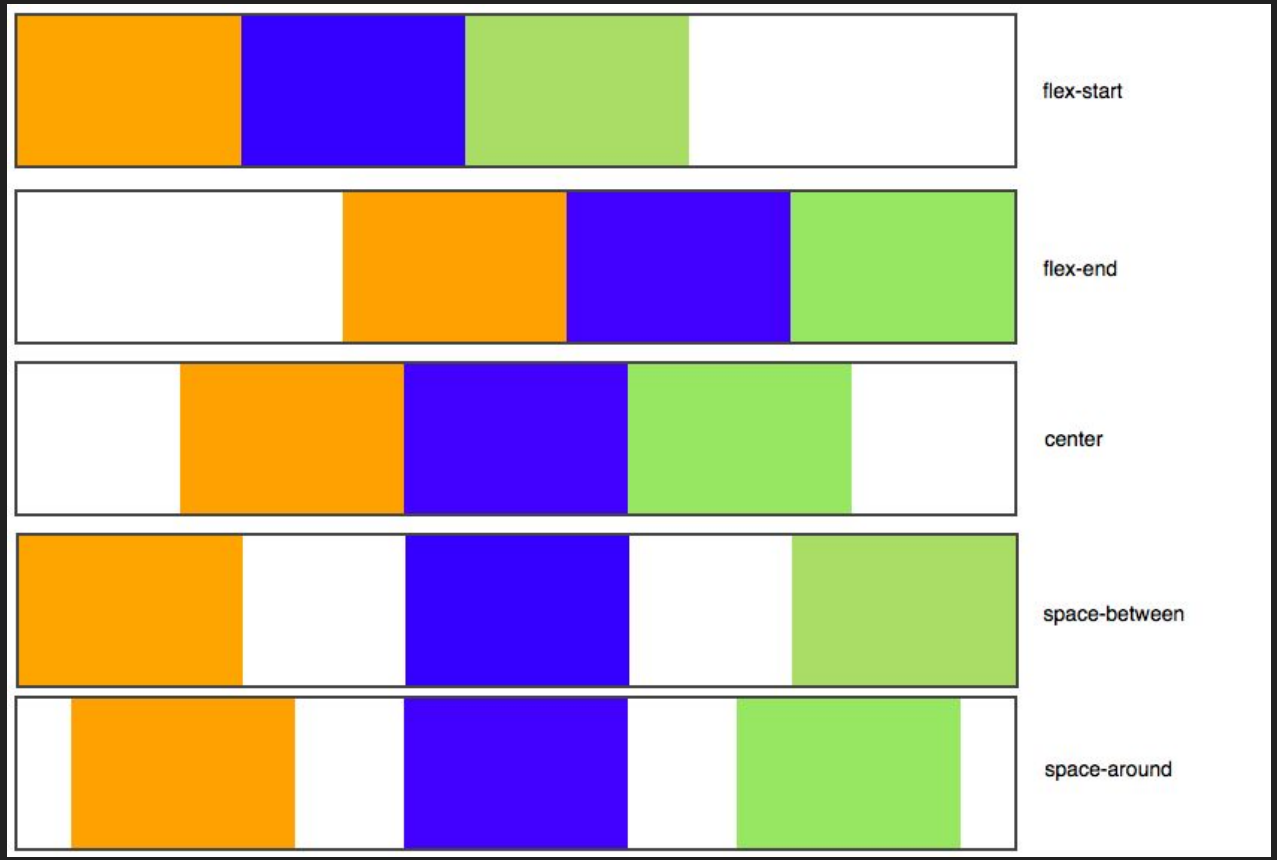
Aligner sur l'axe principal

On va partir sur des éléments organisés horizontalement (par défaut donc).
Pour changer leur alignement, on va utiliser justify-content , qui peut prendre ces valeurs :

- flex-start : alignés au début (par défaut)
- flex-end : alignés à la fin
- center : alignés au centre
- space-between : les éléments sont étirés sur tout l'axe (avec un espace entre eux)
- space-around : idem, les éléments sont étirés sur tout l'axe, mais avec de l'espace sur les extrémités.

Exemple:

```
#conteneur
{       display: flex;
        justify-content: space-around;
}
```



Aligner sur l'axe secondaire

Si nos éléments sont placés dans une direction horizontale (ligne), l'axe secondaire est... vertical, bravo. A l'inverse, si les éléments sont à la verticale (en colonne), l'axe secondaire est alors... horizontal, incroyable !

Avec align-items, vous pouvez changer leur alignement sur l'axe secondaire. Avec ces valeurs :

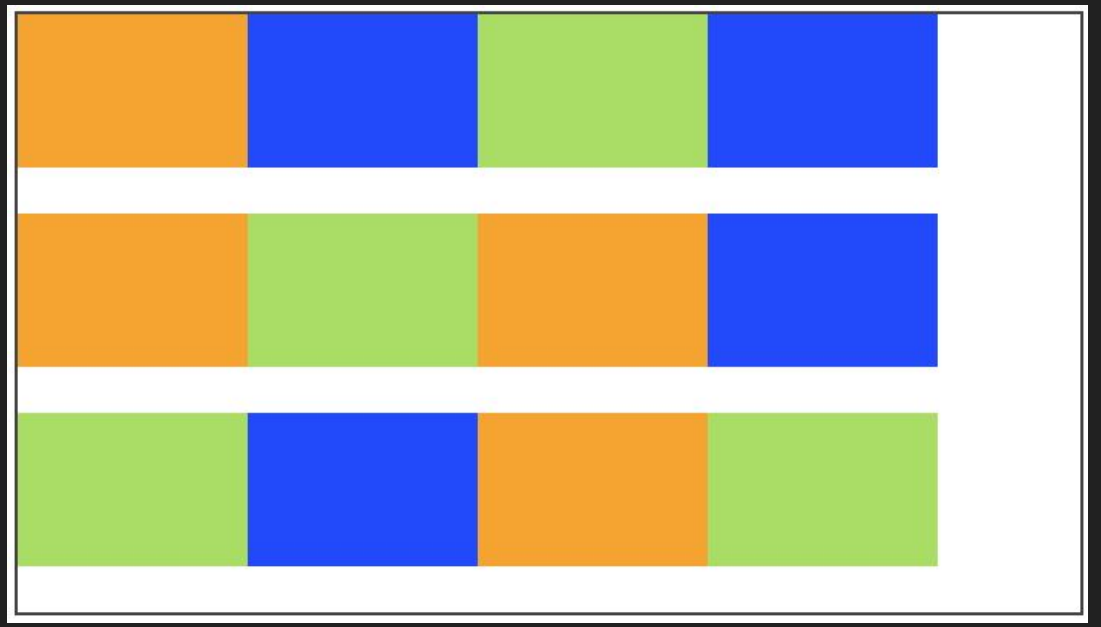
- stretch : les éléments sont étirés sur tout l'axe (valeur par défaut)
- flex-start : alignés au début
- flex-end : alignés à la fin
- center : alignés au centre
- baseline : alignés sur la ligne de base (semblable à flex-start)

Dans le cas où l'on a plusieurs lignes dans notre Flexbox, il est possible de choisir comment les répartir avec align-content.

Exemple:

```
<div id="conteneur">
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
        <div class="element"></div>
</div>
```

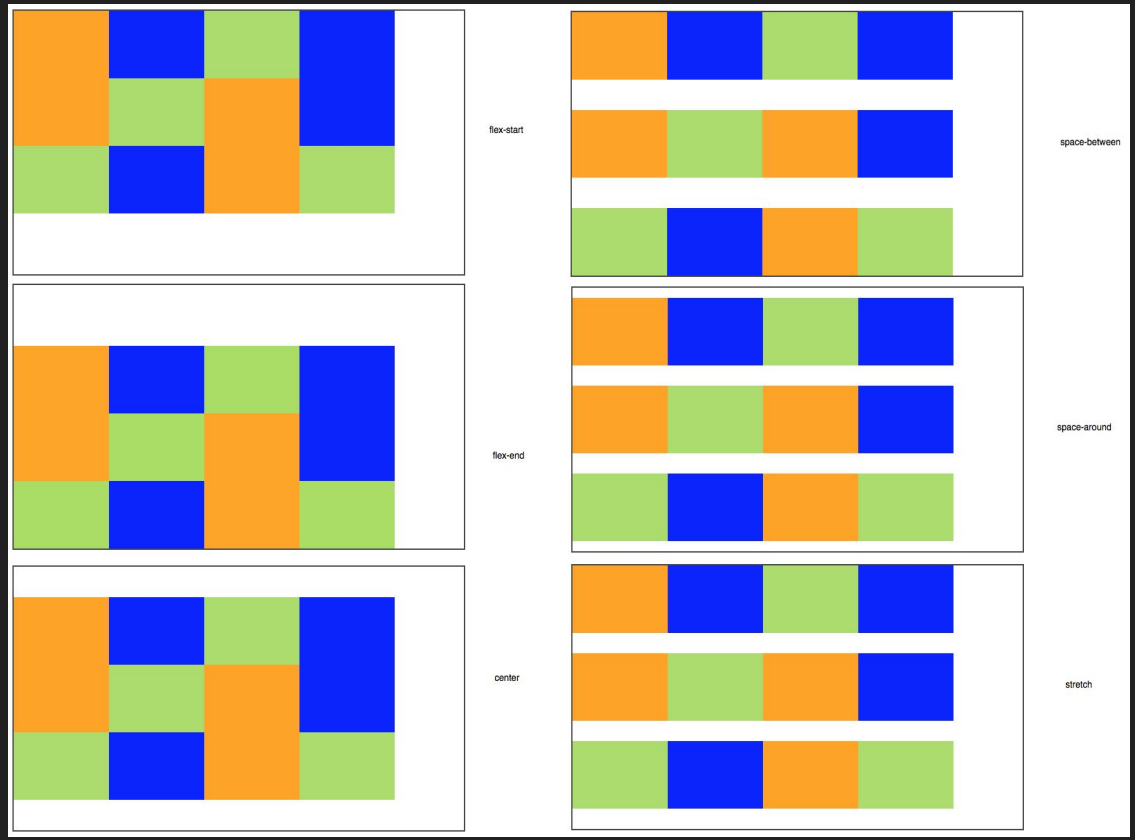
```
#conteneur
{       display: flex;
        flex-wrap: wrap;
}
```



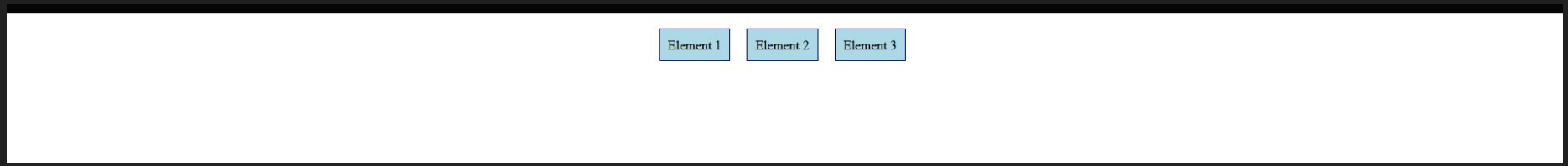
Ok, regardons comment les lignes réagissent avec la nouvelle propriété align-content.

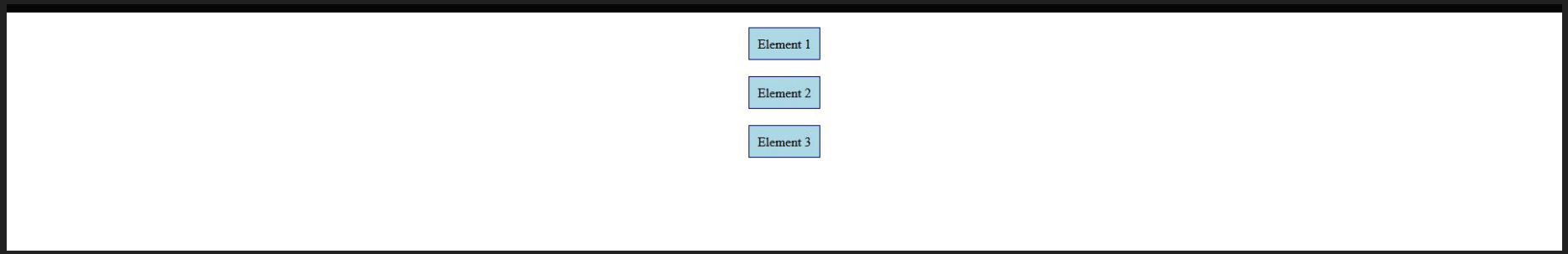
Voici ses valeurs :

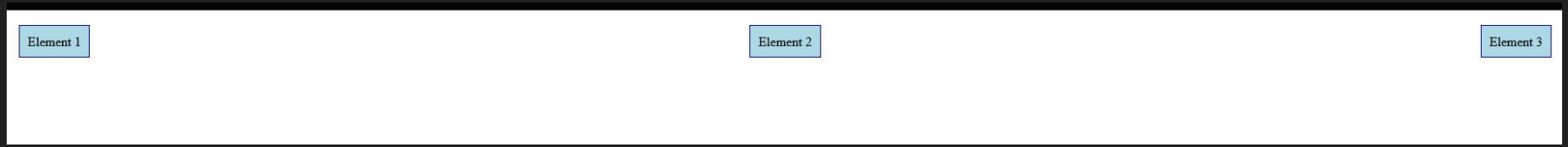
- flex-start : les éléments sont au début
- flex-end : les éléments sont à la fin
- center : les éléments sont au centre
- space-between : les éléments sont séparés avec un espace entre eux
- space-around : même chose, mais il y a un espace au début et à la fin
- stretch (par défaut) : les éléments s'étirent pour occuper tout l'espace

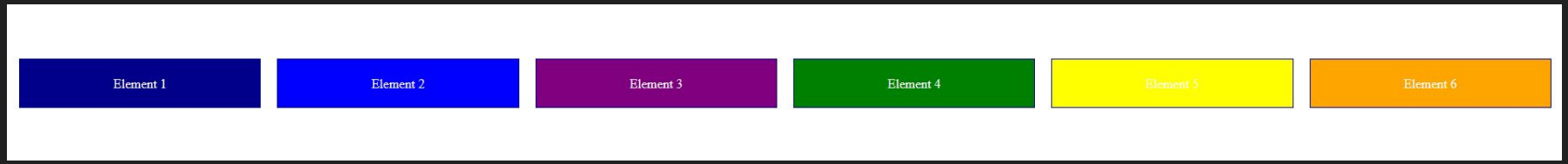


GO to VS CODE!


Element 1
Element 2
Element 3


Element 1
Element 2
Element 3


Element 1
Element 2
Element 3


Element 1
Element 2
Element 3
Element 4
Element 5
Element 6

La propriété CSS `flex` est un raccourci pour définir les trois propriétés suivantes :

- `flex-grow`
- `flex-shrink`
- `flex-basis`

Dans l'exemple que j'ai donné, la valeur `1 0 200px` de la propriété `flex` signifie :

- `flex-grow: 1` : L'élément peut grandir s'il y a de la place disponible dans le conteneur flex.
- `flex-shrink: 0` : L'élément ne peut pas rétrécir s'il y a une pénurie d'espace disponible dans le conteneur flex.
- `flex-basis: 200px` : La taille de base de l'élément est de 200px.

En d'autres termes, cette propriété permet de définir la façon dont l'élément va s'adapter à l'espace disponible dans le conteneur flex. Dans cet exemple, tous les éléments ont une taille de base de 200px et peuvent grandir si nécessaire pour remplir l'espace disponible dans le conteneur flex.

https://codepen.io/enxaneta/pen/adLPwv

https://marina-ferreira.github.io/tutorials/css/flexbox/

https://css-tricks.com/wp-content/uploads/2022/02/css-flexbox-poster.png

https://flexboxfroggy.com/#fr