HTML/CSS – Module 3

Support de cours

Qu'est-ce que le flexbox CSS ?

Le module CSS3 `Flexbox Layout` fournit une façon efficace de disposer, aligner et distribuer l'espace entre des balises enfants (nommées items) dans une balise parente (nommée conteneur flexbox), même lorsque leurs dimensions sont inconnues et/ou dynamiques - d'où le terme "flex" pour flexible.

L'idée principale est de donner à un élément contenant (la balise conteneur) la possibilité de changer les largeur et hauteur des éléments contenus (les balises items), afin de remplir au mieux l'espace disponible, et de s'adapter à tous les terminaux et toutes les tailles d'écrans. Un conteneur flexible permet aux items de s'étendre pour occuper la place disponible ou au contraire les réduit pour leur éviter de déborder.

Les bases du flexbox

Le module flexbox est une série de propriétés à appliquer principalement sur la balise conteneur. Notez toutefois, qu'il existe quelques propriétés qui pourront s'appliquer sur les items, mais elles sont moins nombreuses et concernent surtout le dimensionnement et l'ordre des items.

Qu'est-ce qu'un conteneur et un item dans le module flexbox ?

Comme son nom l'indique, la balise **conteneur** joue le rôle de boîtes qui va contenir les **items**. Dit autrement c'est la balise **parente**. Les **items** sont quant à eux les **enfants directs** (ou balise de **premier niveau**) qui se trouvent à l'intérieur de la balise **parente**. Voici un extrait de code qui montre la structure de base nécessaire pour faire du flexbox :

```
<div><!-- balise conteneur (ou balise parente) -->
    <article> <!-- item (ou balise enfant direct) -->
        <h2>titre 1</h2> <!-- cette balise n'est pas un item -->
        <p>Lorem ipsum dolor sit amet.</p> <!-- cette balise n'est pas un item -->
    </article>
    <article> <!-- item -->
        <h2>titre 2</h2>
        <p>Doloremque qui nisi neque suscipit!</p>
    </article>
    <article> <!-- item -->
        <h2>titre 3</h2>
        <p>Voluptates explicabo voluptate nobis similique!</p>
    </article>
    <article> <!-- item -->
        <h2>titre 4</h2>
        <p>Iste, alias. Minus, eligendi enim.</p>
    </article>
</div>
```

Dans cet exemple, la balise ** ** est le conteur flex, parce qu'elle est la balise parente des 4 balises **<article>** qui elles sont les items de la ** ** parce qu'elles sont les enfants directs (ou enfants de

premier niveau. En revanche les balise **<h2>** et **<p>** ne sont pas des items de la , parce qu'elles ne sont pas des enfants directs.

L'importance de la balise conteneur flex
La balise **conteneur flex** joue un rôle primordial dans la mise en place des **items** dans l'espace de la page. En effet, c'est sur cette balise que vous appliquerez l'essentiel des **propriétés CSS flexbox**.

Les propriétés flexbox

Commençons par les propriétés qui s'appliquent à l'élément parent, c'est-à-dire la balise conteneur flex.

La propriété display: flex;
On applique cette propriété généralement sur une balise de type block. C'est ainsi qu'on définit un **conteneur flex**.

Exemple HTML :

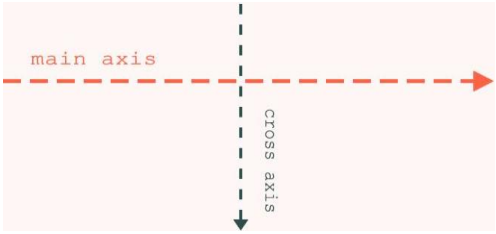
```
    <div class="flex"><!-- balise conteneur (ou balise parente) -->
       <article> <!-- item (ou balise enfant direct) -->
       </article>

       <article> <!-- item (ou balise enfant direct) -->
       </article>

       <article> <!-- item (ou balise enfant direct) -->
       </article>

       <article> <!-- item (ou balise enfant direct) -->
       </article>
    </div>
```

Exemple CSS :

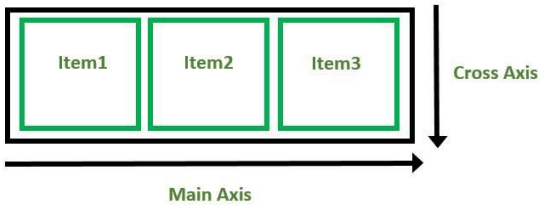
```
.flex{
    display: flex;
}
```

Dès que cette propriété est appliquée, cela crée un **contexte flex** sur ladite balise qui la transforme **en conteneur flex**, changeant immédiatement tous les enfants directs de celles-ci en **items**. Par défaut, les **items** s'alignent **horizontalement** et se placent à gauche dans le **conteneur flex**.

Ce phénomène s'explique parce que la propriété **display: flex;** ajoute automatiquement **deux axes virtuels** dans la balise **conteneur flex**. Un axe **horizontal** appelé **main axis** (en rouge sur l'image) et un axe **secondaire vertical perpendiculaire**, appelé **cross axis** (en noir sur l'image).

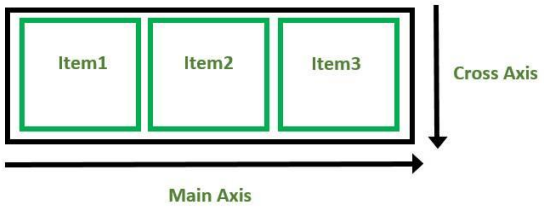


Par défaut et en l'absence d'autres propriétés **flex** sur la balise **conteneur flex**, les **items** s'alignent sur **l'axe principale** (main axis) en **partant de la gauche** (en occident).

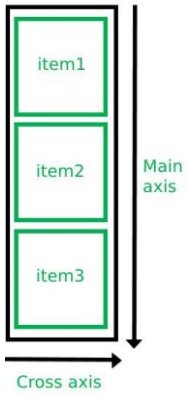


Propriété flex-direction :
Cette propriété permet de changer la direction de **l'axe principal** (main axis). Il existe plusieurs valeurs possibles :

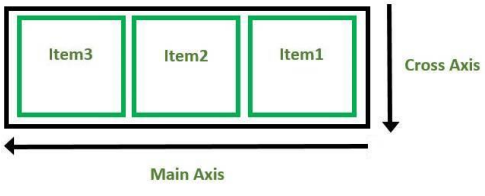
- **flex-direction: row;** C'est la valeur par défaut. C'est pourquoi le **main axis** est à l'horizontal. A noter qu'en occident l'axe va de la gauche vers la droite.



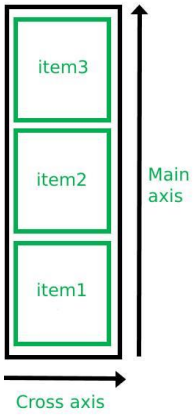
- **flex-direction: column;** C'est valeur fait basculer le **main axis** de **90 degrés dans le sens des aiguille d'une montre**. Les items s'alignent sur l'axe principal l'un en dessous de l'autre dans l'ordre d'arrivée dans le code HTML (le premier item est premier, le dernier item est dernier).



- **flex-direction: row-revers;** Cette valeur laisse le **main axis** à l'horizontal, mais force l'alignement des items en partant de la droite (en occident).

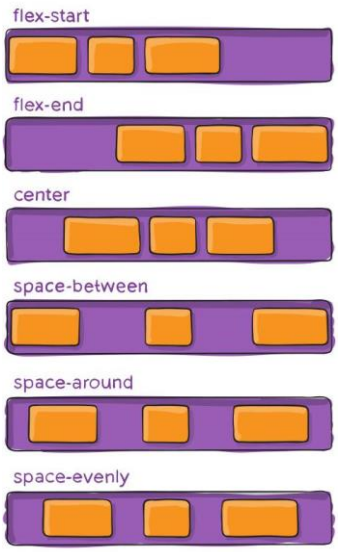


- **flex-direction: column-revers;** C'est valeur fait basculer le **main axis** de **90 degrés dans le sens des aiguille d'une montre**. Les items s'alignent sur l'axe principal l'un en dessous de l'autre en inversant l'ordre (le dernier est premier, le premier est dernier).



La propriété justify-content :
La propriété **justify-content** définit l'alignement le long de l'axe principal. Elle permet de distribuer l'espace excédentaire lorsque tous les items flex sur une ligne sont inflexibles ou lorsqu'ils ont atteint leur taille maximale. Elle contrôle aussi l'alignement des items lorsqu'ils débordent. Cette propriété est associée au **main axis**.

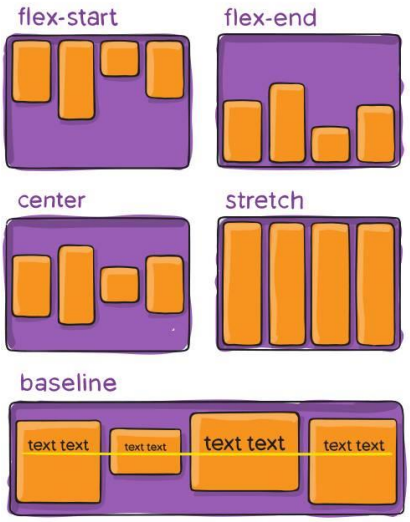
- **justify-content: flex-start;** (par défaut) : les **items** sont regroupés en début de ligne
- **justify-content: flex-end** : les **items** sont regroupés en fin de ligne
- **justify-content: center** : les **items** sont centrés le long de la ligne
- **justify-content: space-between** : les **items** sont répartis sur la ligne; le premier est collé du côté start (à gauche), le dernier du côté end (à droite). L'espace entre chaque item est réparti de façon égale.
- **justify-content: space-around** : les **items** sont répartis sur la ligne avec un espacement égal autour de chacun.
- **justify-content: space-evenly** : les **items** sont répartis sur la ligne avec un espacement égal entre eux.



La propriété align-items :
La propriété **align-items** définit la façon dont les items d'une ligne sont disposés le long de l'axe secondaire **cross axis**.

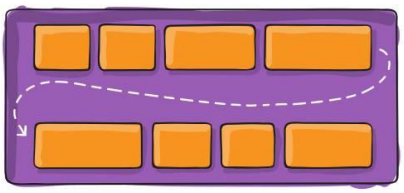
- **align-items: stretch** (par défaut) : les items sont étirés de haut en bas jusqu'à remplir le container. Les items sont alignés sur le bas de l'axe cross axis, s'il y a suffisamment d'espace verticale. Sinon, cet effet ne se voit pas.
- **align-items: flex-start** : l'item est placé en haut de l'axe **cross axis**, s'il y a suffisamment d'espace verticale. Sinon, cet effet ne se voit pas.
- **align-items: flex-end** : Les items sont alignés sur le bas de l'axe **cross axis**, s'il y a suffisamment d'espace verticale. Sinon, cet effet ne se voit pas.
- **align-items: center** : les items sont centrés sur l'axe **cross axis**, s'il y a suffisamment d'espace verticale. Sinon, cet effet ne se voit pas.
- **align-items: baseline** : les items sont alignés sur la ligne de base du texte qu'ils contiennent.

A noter que si l'ont fait bascule l'axe principal (**main axis**) de **90 degrés** l'axe secondaire (**cross axis**) qui est, **par défaut vertical**, se place alors à **l'horizontal** et dès lors la propriété **align-items** s'applique sur **l'horizontale** et non plus sur la verticale. Cette propriété est donc liée par défaut à l'axe vertical.

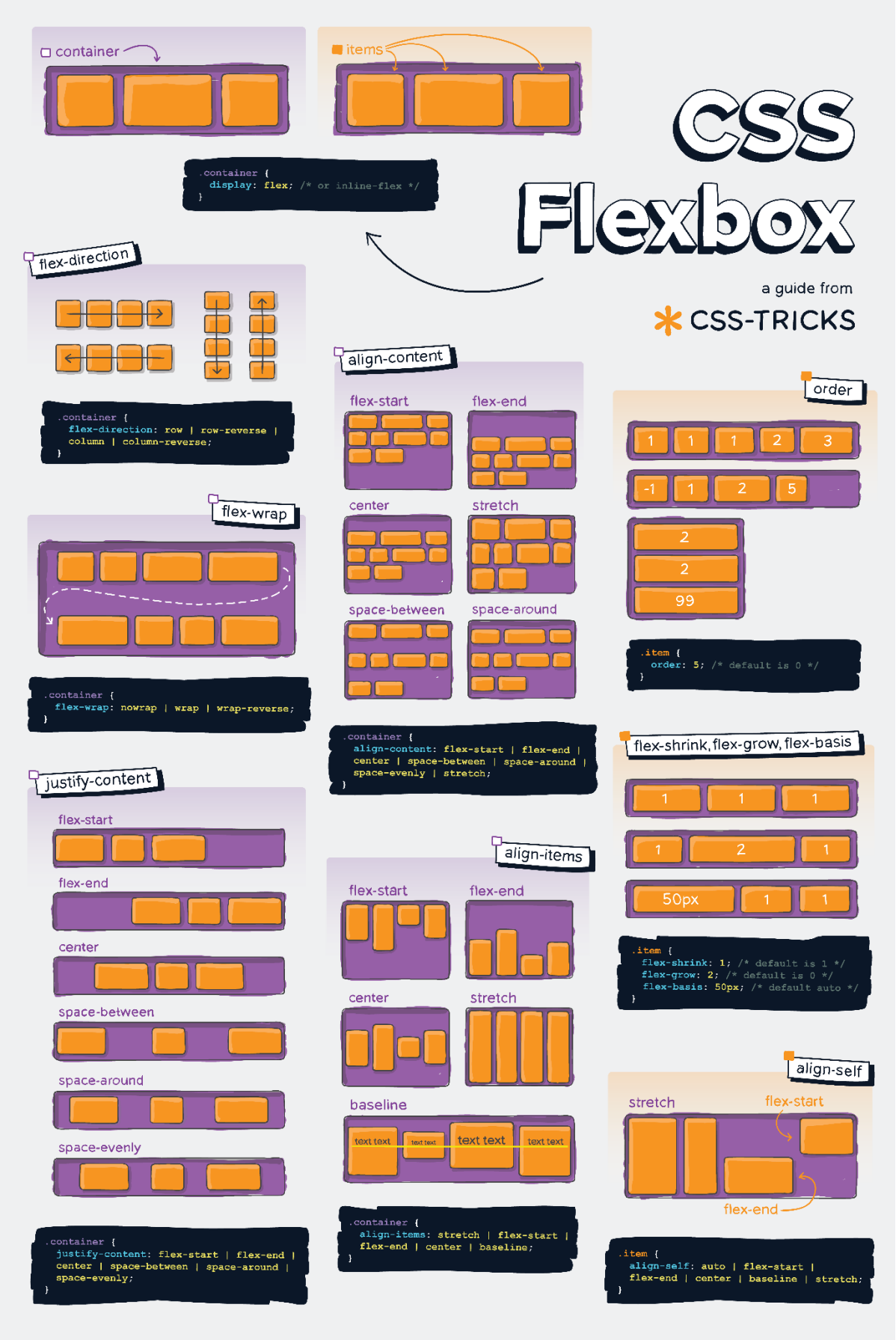


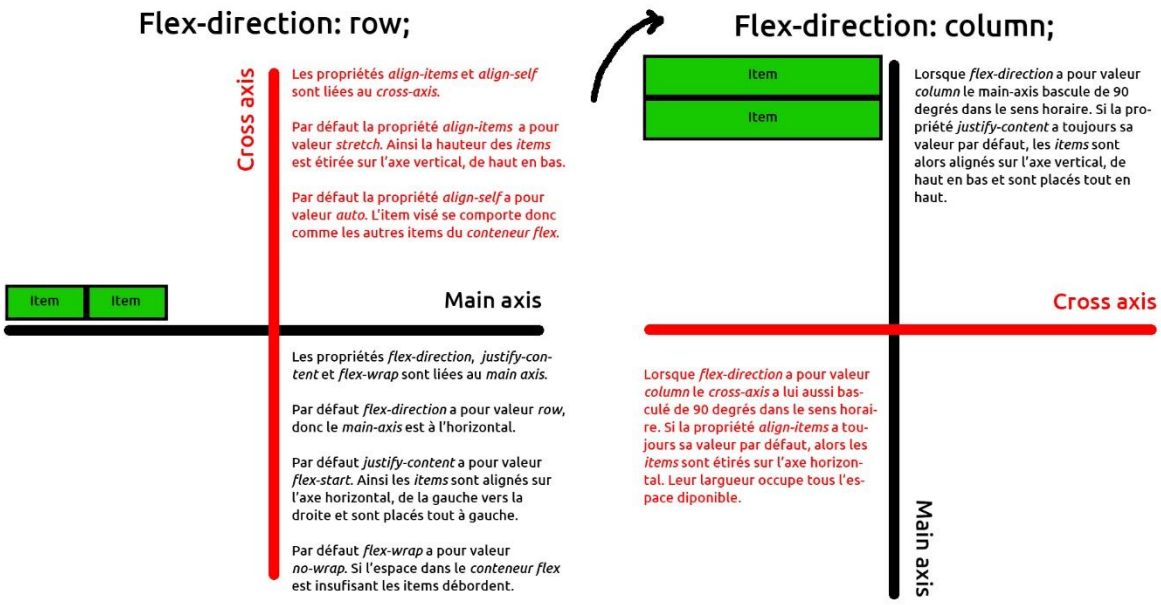
Propriété flex-wrap :
La propriété **flex-wrap** définit si **le conteneur flex** force ou pas le retour à la ligne des items s'ils n'ont pas assez d'espace dans ledit conteneur, pour rester sur une seule ligne. Il n'existe que deux valeurs possibles :

- **flex-wrap: no-wrap;** C'est la valeur par défaut qui force les **items** à rester en ligne, même s'il n'y a pas assez d'espace dans le **conteneur flex**, quitte à déborder.
- **flex-wrap: wrap;** Cette valeur force les **items** passer à la ligne, s'il n'y a pas assez d'espace dans le **conteneur fle**x.



Récapitulatifs :





Pour en savoir plus :
Le guide complet du flexbox : https://la-cascade.io/articles/flexbox-guide-complet
A Complete Guide to Flexbox : https://css-tricks.com/snippets/css/a-guide-to-flexbox/