

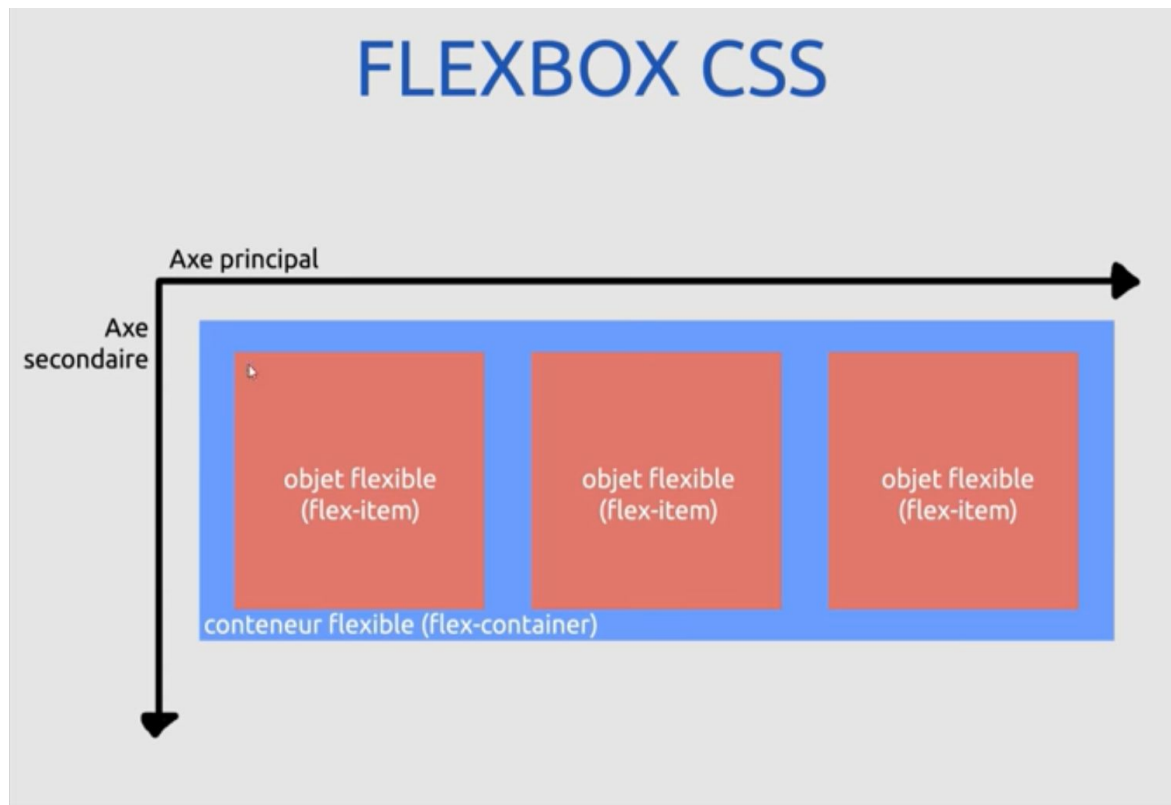


**FLEXBOX CSS**

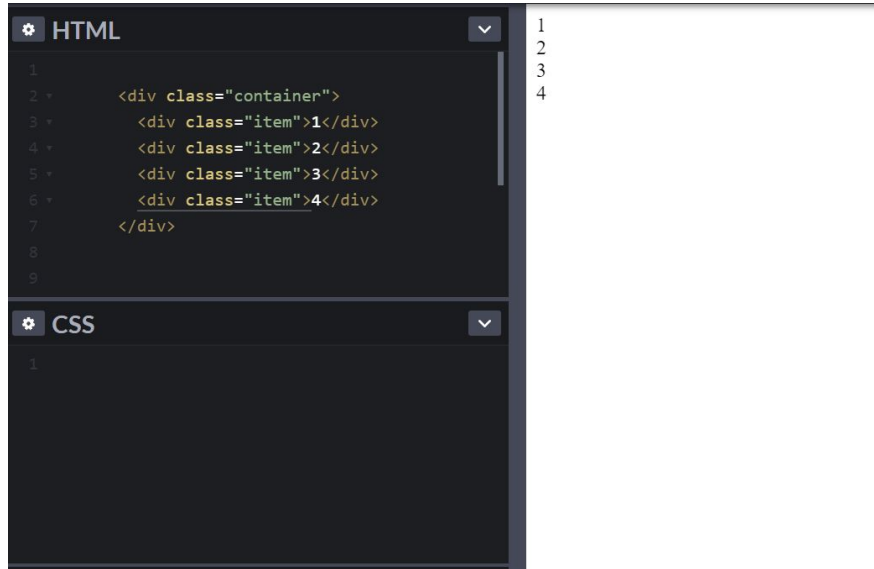
# Flexbox - Définition

Le module des boîtes flexibles, aussi appelé « *flexbox* », a été conçu comme un modèle de disposition unidimensionnel (horizontal OU vertical) et comme une méthode permettant de distribuer l'espace entre des objets d'une interface ainsi que de les aligner.

# Flexbox - Définition



# Flexbox - Comportement par défaut



```
HTML
1
2 <div class="container">
3   <div class="item">1</div>
4   <div class="item">2</div>
5   <div class="item">3</div>
6   <div class="item">4</div>
7 </div>
8
9
CSS
1
```

Voici un exemple de code HTML classique avec un comportement normal. Les éléments “div” étant de type “bloc” se mettent les uns sous les autres.

# Flexbox - Comportement par défaut

Lorsque je définis la propriété “display avec la valeur “flex” (sans autre instruction), on constate que le comportement par défaut des éléments changent.

Ils se mettent comme des éléments ‘inline’. On peut en déduire que la direction par défaut de flexbox est ‘en ligne’ ou “row”



The screenshot shows a code editor with two panels. The top panel is labeled 'HTML' and contains the following code:

```
1  
2 <div class="container">  
3   <div class="item">1</div>  
4   <div class="item">2</div>  
5   <div class="item">3</div>  
6   <div class="item">4</div>  
7 </div>  
8  
9
```

The bottom panel is labeled 'CSS' and contains the following code:


```
1 .container{  
2   display: flex;  
3 }
```

On the right side of the editor, there is a line number '1234'.

# Flexbox - Comportement par défaut

Chose important à savoir,  
La propriété “flex” n’affecte  
QUE les enfants directs.

C’est pour cela que la div  
qui a le numéro 5 n’est pas  
alignée comme les autres  
car elle n’est pas l’enfant  
direct de “container” qui est  
en “display: flex”



Flexbox right sidebar

Juc1 + Follow

HTML

```
1
2 <div class="container">
3   <div class="item">1</div>
4   <div class="item">2</div>
5   <div class="item">3</div>
6   <div class="item">4
7     <div class="item">5</div>
8   </div>
9 </div>
```

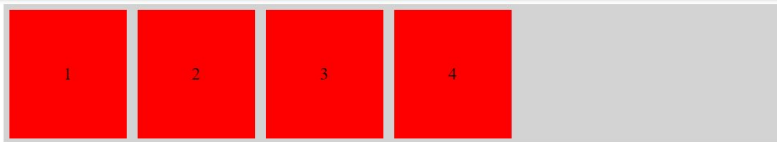
1234  
5

CSS

```
1 .container{
2   display: flex;
3 }
```

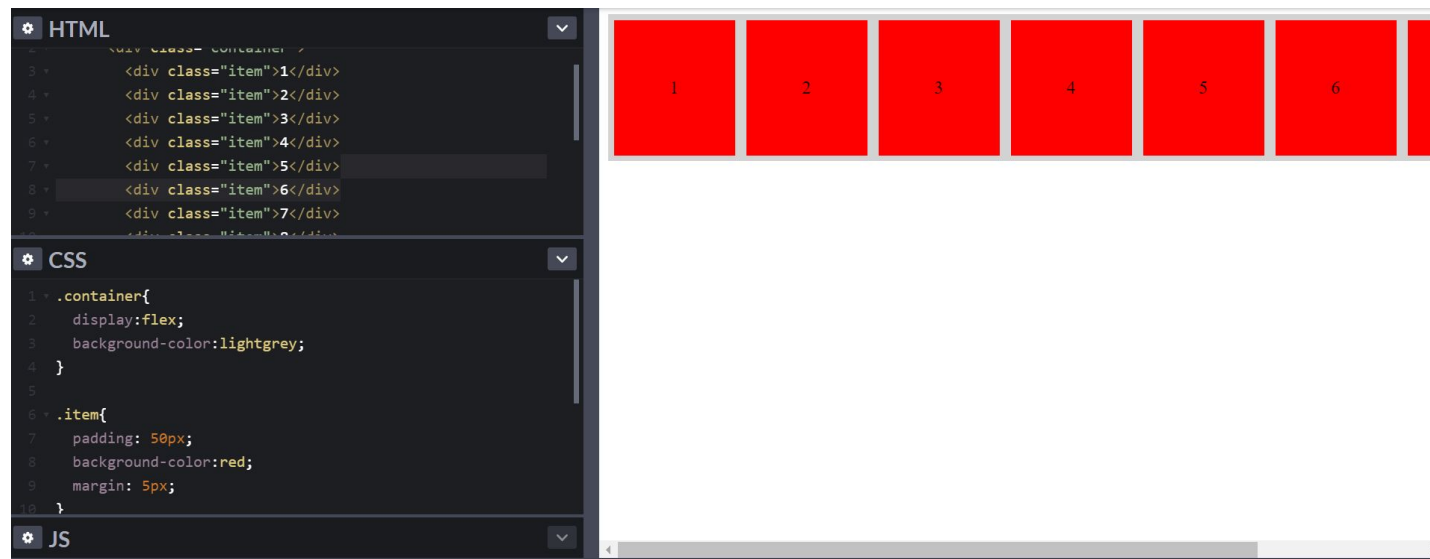
# Flexbox - Comportement par défaut

```
HTML
1 <div class="container">
2   <div class="item">1</div>
3   <div class="item">2</div>
4   <div class="item">3</div>
5   <div class="item">4</div>
6 </div>
7
8
CSS
1 .container{
2   display:flex;
3   background-color:lightgrey;
4 }
5
6 .item{
7   padding: 50px;
8   background-color:red;
9   margin: 5px;
10 }
```



Pour continuer, j'ai stylisé nos div et j'ai mis une couleur de fond au 'container' pour pouvoir le visualiser...

# Flexbox - Comportement par défaut



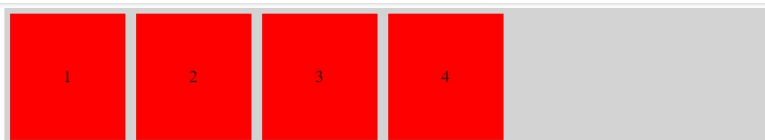
Autre chose importante, si j'ai plus d'éléments que ne peut contenir mon 'container', par défaut flexbox les mettra les uns après les autres quitte à les faire dépasser de mon écran (et c'est moche....)



# Flexbox - flex-direction

```
HTML
1 <div class="container">
2   <div class="item">1</div>
3   <div class="item">2</div>
4   <div class="item">3</div>
5   <div class="item">4</div>
6 </div>
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

CSS
1 .container{
2   display:flex;
3   background-color:lightgrey;
4   flex-direction: row;
5   flex-wrap: nowrap;
6 }
7
8 .item{
9   padding: 50px;
10  background-color:red;
11 }
```



Je viens d'ajouter 1 nouvelle propriété “flex-direction” .  
“Flex-direction” définie le sens dans lequel nous allons travailler (par défaut ‘row’)

# Flexbox - flex-direction

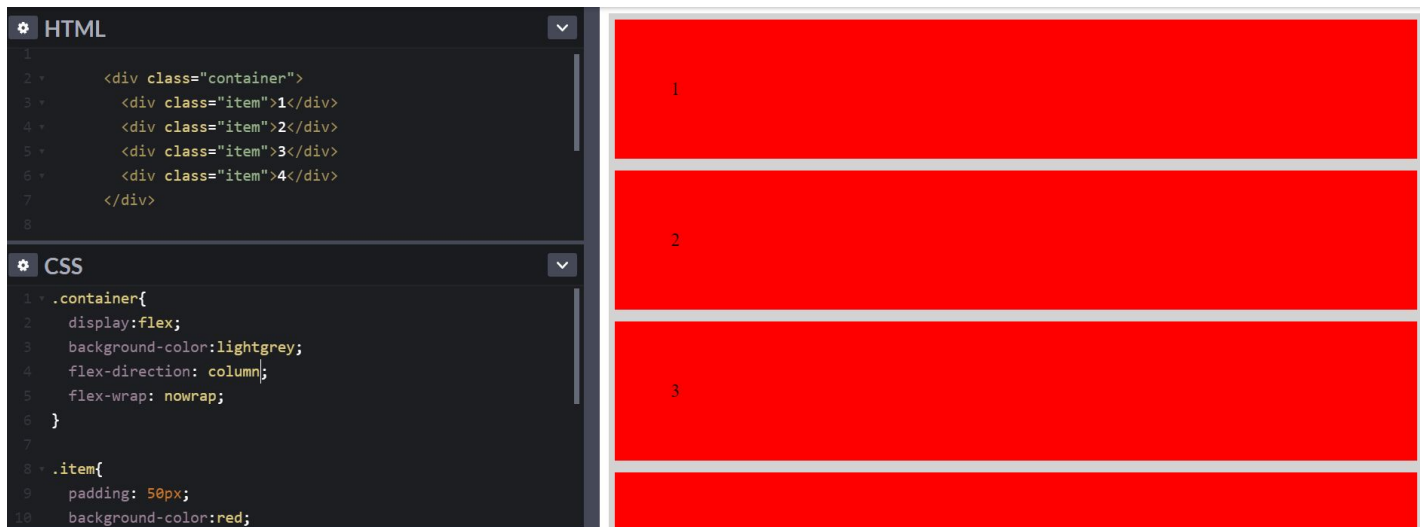
**Flex-direction: row (défaut) | column | row-reverse | column-reverse**

La propriété **flex-direction** définit la façon dont les éléments flexibles sont placés dans un conteneur flexible : elle définit l'axe principal et la direction des éléments (normale ou inversée).

Source :

<https://developer.mozilla.org/fr/docs/Web/CSS/flex-direction>

# Flexbox - flex-direction



Si je change la propriété par défaut de “flex-direction” et que je le mets en “column” à la place de “row”, on constate que les éléments se mettent les uns au dessous des autres (le comportement normal d’une div en fait....)

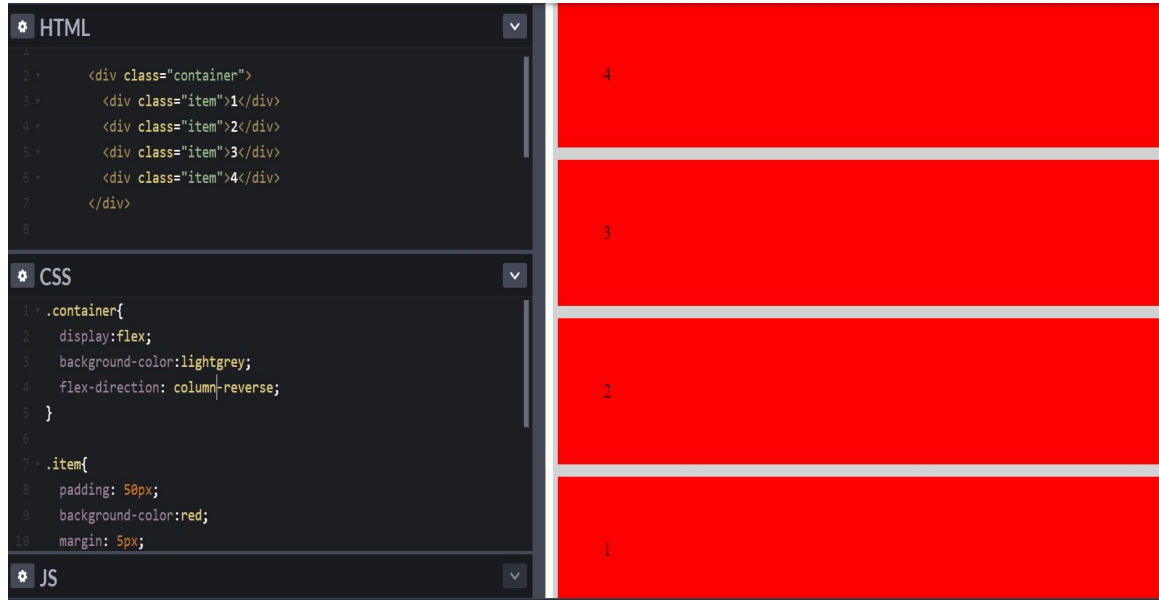
Les colonnes sont étirées mais ça, on le verra après.....

# Flexbox - Comportement par défaut



Il existe la valeur “row-reverse” qui change non seulement le sens des éléments mais aussi le sens de début (à droite au lieu de gauche)

# Flexbox - flex-direction



Il existe la valeur “column-reverse” qui change non seulement le sens des éléments mais aussi le sens de début (à droite au lieu de gauche)

# Flexbox - flex-wrap

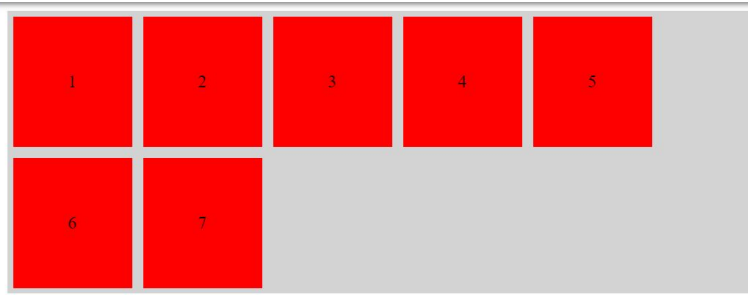
## Flex-wrap: nowrap (défaut) | wrap | wrap-reverse

La propriété `flex-wrap` indique si les éléments flexibles sont contraints à être disposés sur une seule ligne ou s'ils peuvent être affichés sur plusieurs lignes avec un retour automatique. Si le retour à la ligne est autorisé, la propriété permet également de contrôler la direction dans laquelle les lignes sont empilées.

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/flex-wrap>

# Flexbox - flex-wrap

```
HTML
1 <div class="item">1</div>
2 <div class="item">2</div>
3 <div class="item">3</div>
4 <div class="item">4</div>
5 <div class="item">5</div>
6 <div class="item">6</div>
7 <div class="item">7</div>
8 </div>
9
CSS
1 .container{
2   display: flex;
3   background-color: lightgrey;
4   flex-direction: row;
5   flex-wrap: wrap;
6 }
7
8 .item{
9   padding: 50px;
10  background-color: red;
11 }
```



Si je change “flex-wrap” de “nowrap” à “wrap”, on constate que les éléments ne sortent plus de leur “container”. (wrap = enveloppé)





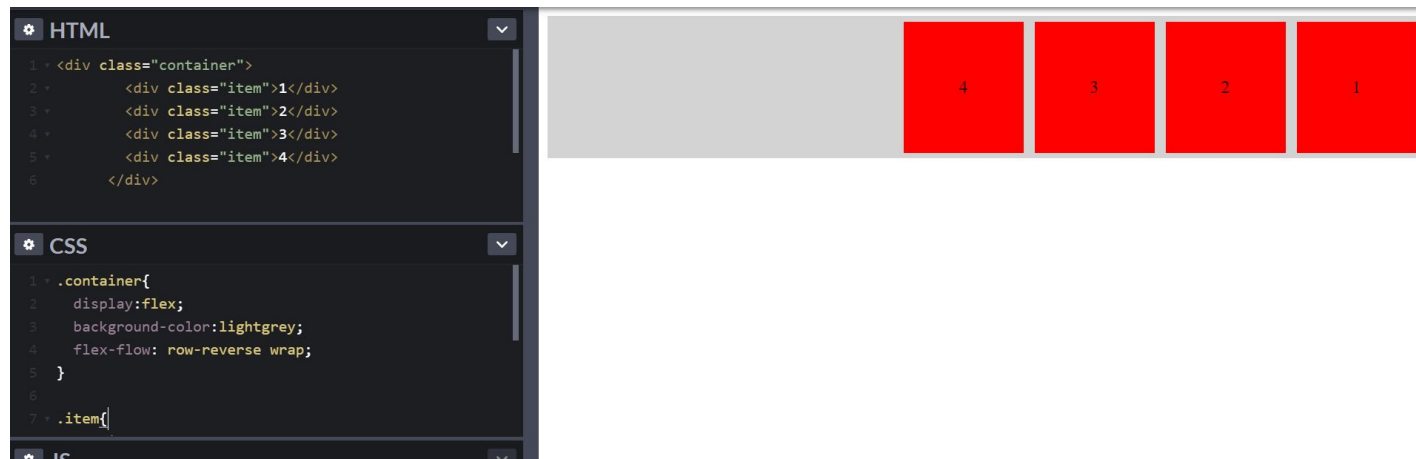
# Flexbox - flex-flow (raccourci)

**Flex-flow : <flex-direction> <flex-wrap>**

La propriété CSS `flex-flow` est une propriété raccourcie pour les propriétés `flex-direction` et `flex-wrap`.

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/flex-flow>

# Flexbox - flex-flow (raccourci)



On voit que j'ai utilisé la propriété "flex-flow" avec les valeurs "row-reverse wrap".

C'est comme si j'avais utilisé "flex-direction: row-reverse; flex-wrap: wrap;"

# Flexbox - justify-content

**Justify-content : flex-start (défaut) | center | flex-end |  
space-around | space-between | space-evenly**

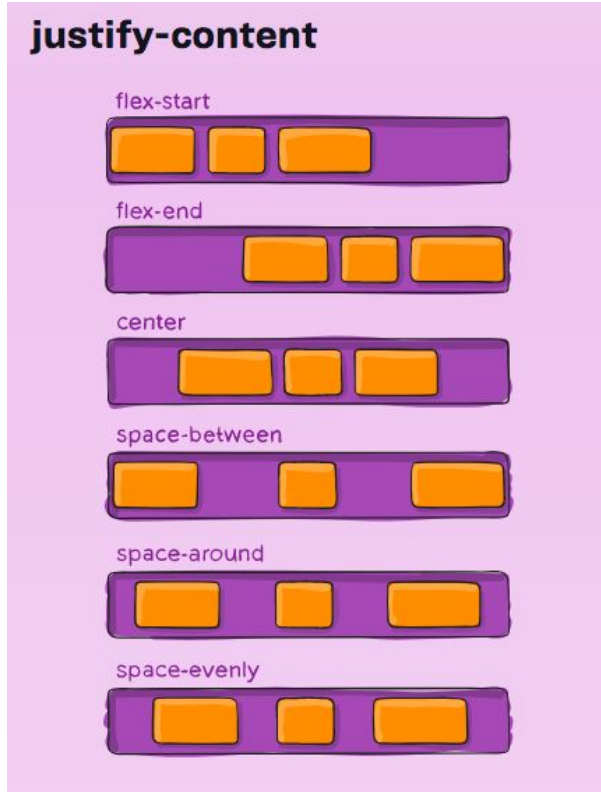
La propriété CSS **justify-content** indique la façon dont l'espace doit être réparti entre et autour des éléments selon l'axe principal d'un conteneur flexible ou selon l'axe en ligne lorsque le conteneur est une grille.

Source :

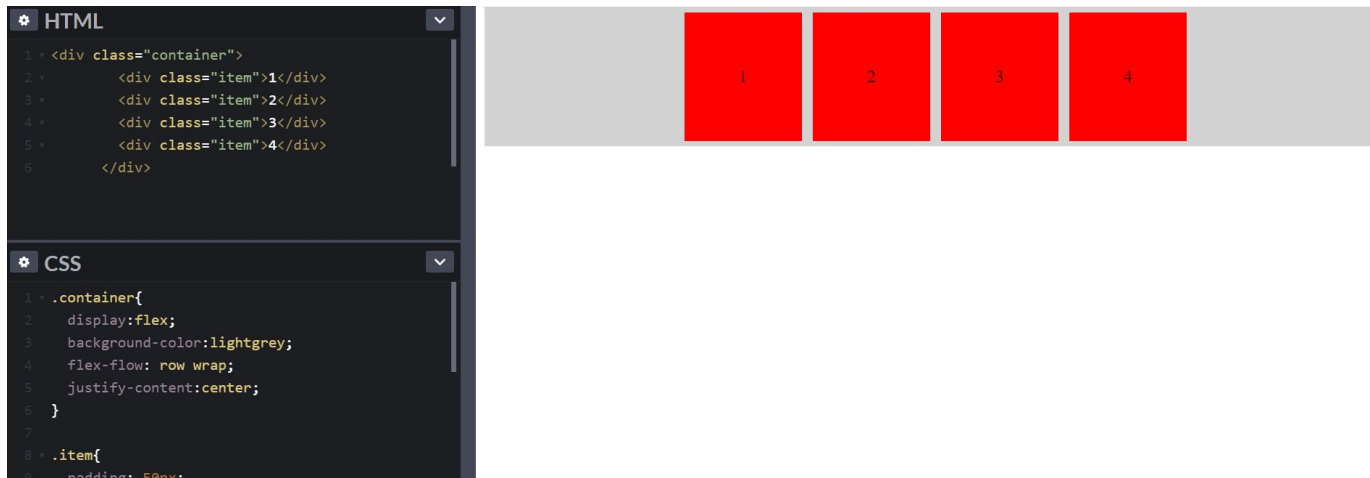
<https://developer.mozilla.org/fr/docs/Web/CSS/justify-content>

Attention, l'axe principal est horizontal par défaut (donc en "flex-direction:row") mais il est vertical si on passe en "flex-direction:column".

# Flexbox - justify-content



# Flexbox - justify-content



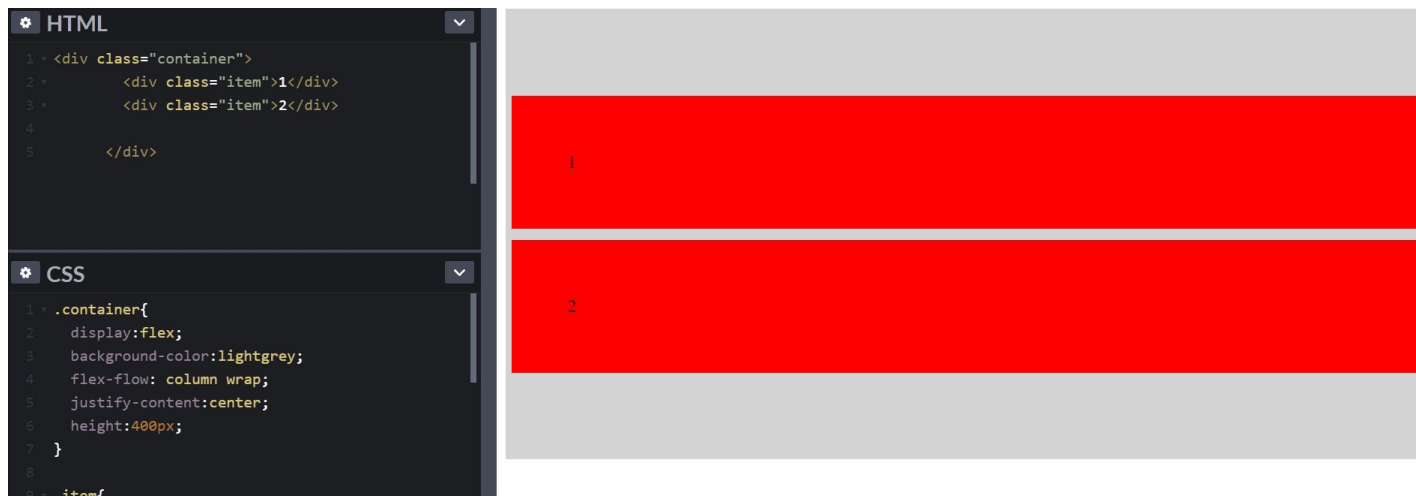
On peut voir que l'espace entre nos éléments reste le même peu importe la taille de mon écran.

# Flexbox - justify-content



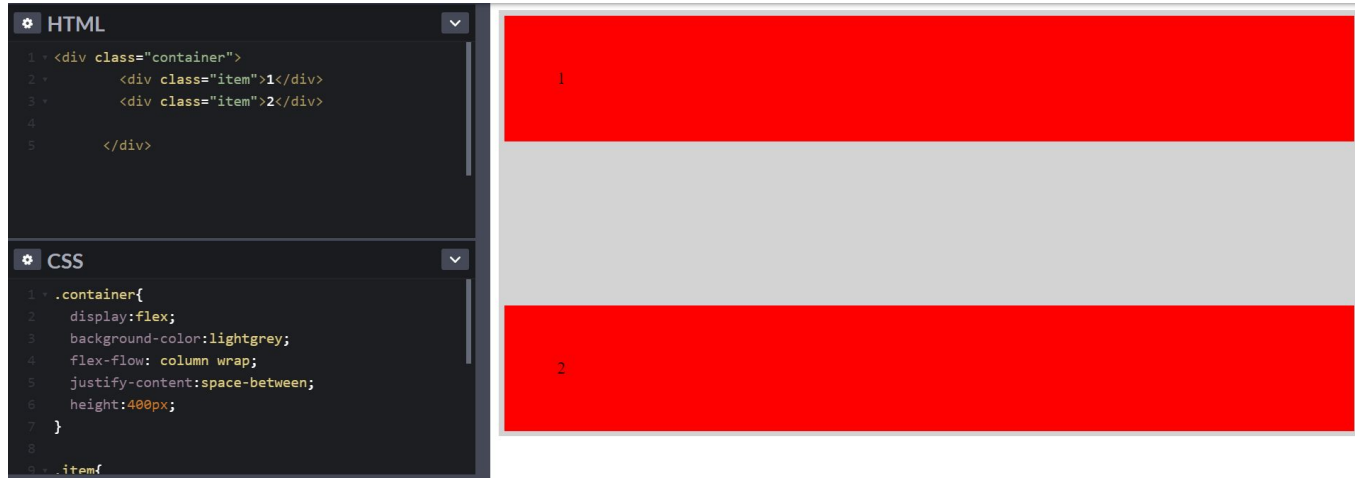
Même exemple avec “justify-content; space-around”

# Flexbox - justify-content



Voici un exemple avec une direction en colonne. Le “justify-content” est “center”. Contrairement aux deux exemples d’avant, on constate que l’alignement ne se fait plus horizontalement mais verticalement. L’axe principal a changé !

# Flexbox - justify-content



Même exemple avec “justify-content; space-between” en direction en colonne.



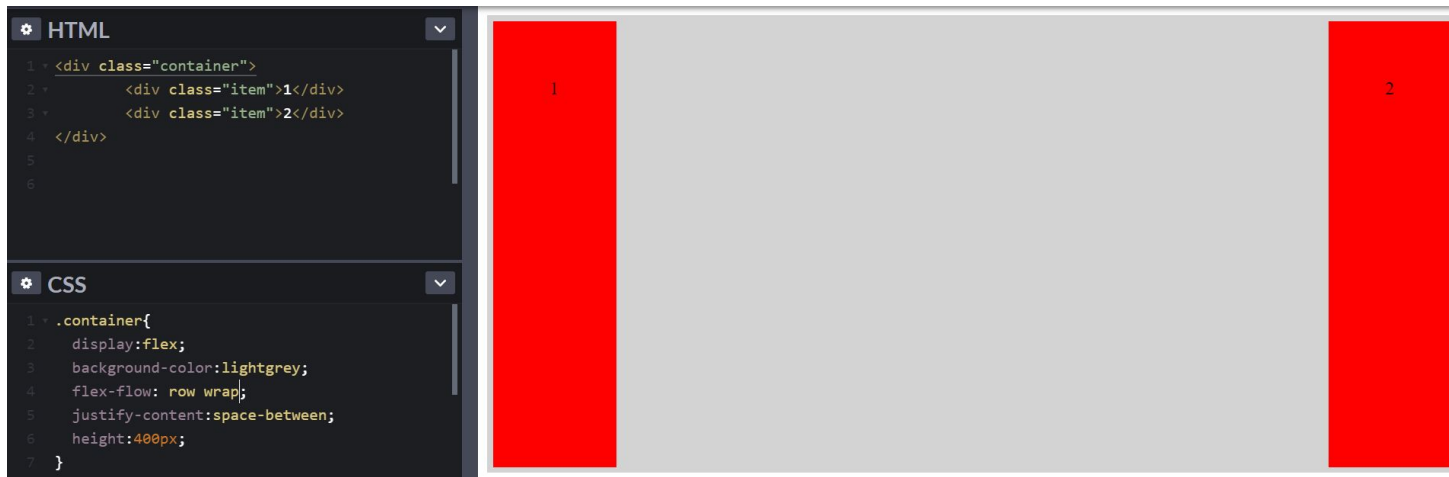
# Flexbox - align-items

**Align-items : stretch (défaut) | flex-start | center | flex-end | baseline**

La propriété CSS **align-items** définit la valeur sur l'ensemble des éléments-fils directs. Pour les boîtes flexibles, cette propriété contrôle l'alignement par rapport à l'axe secondaire (*cross axis*).

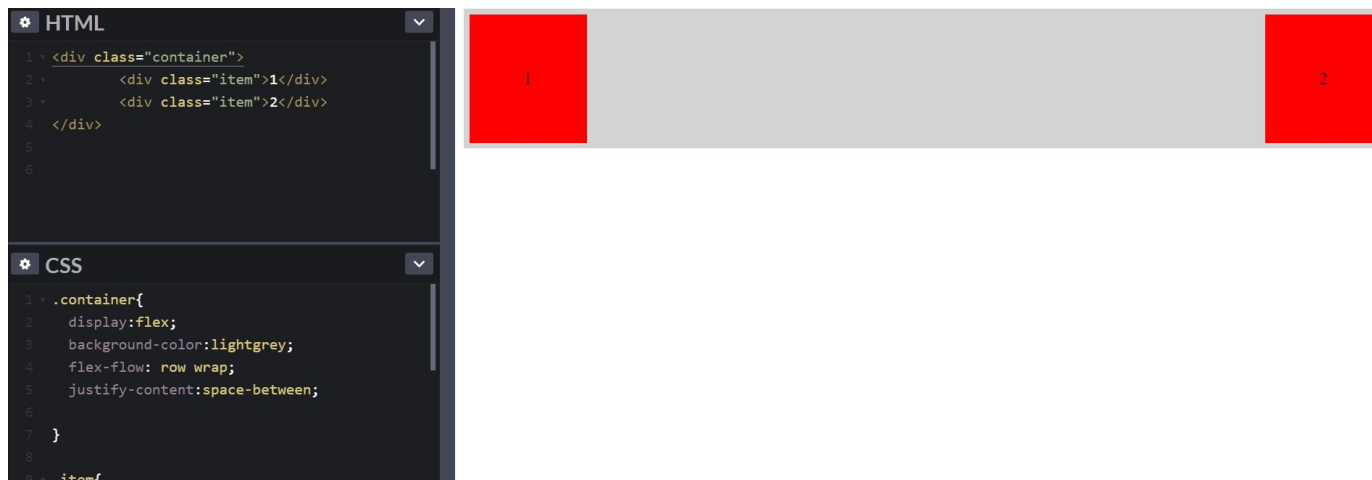
Source : <https://developer.mozilla.org/fr/docs/Web/CSS/align-items>

# Flexbox - align-items



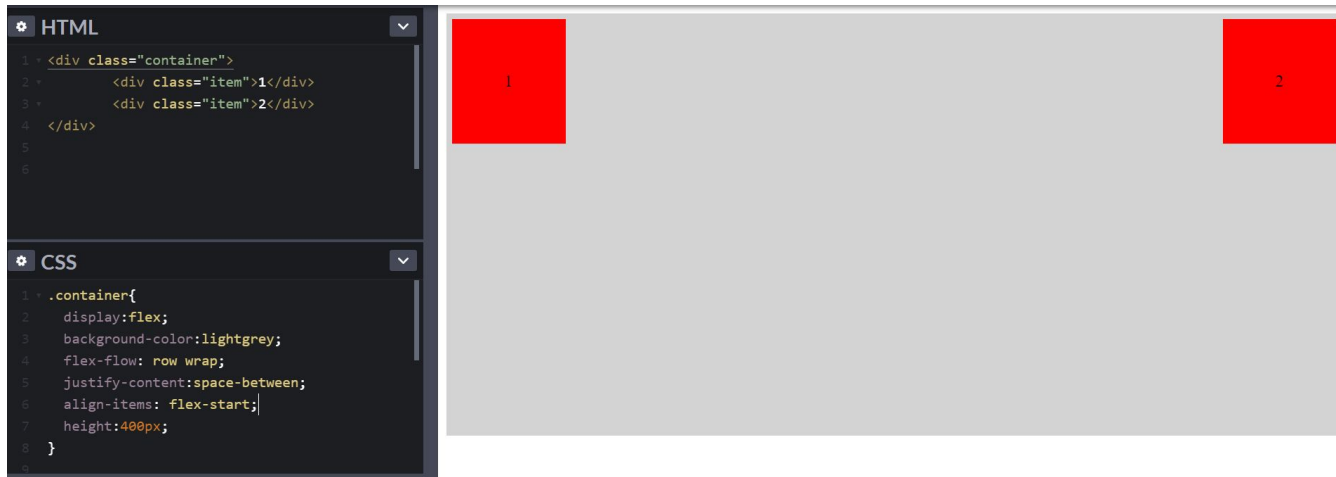
Quand je mets mes éléments en “flex-direction: row” l’axe principal est horizontal. L’axe secondaire est vertical. Et comme par défaut, la valeur est “stretch” (étiré), le carré est allongé vers le bas.

# Flexbox - align-items



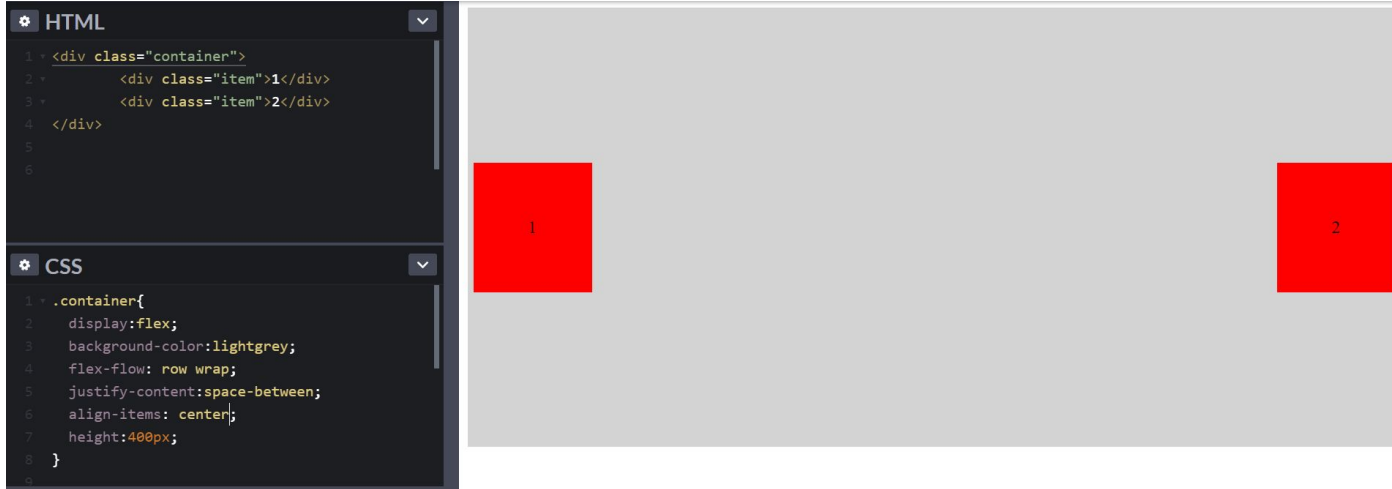
Alors, je sais, vous allez me dire que tout à l'heure, l'effet “stretch” n’y était pas. C’est parce-que tout à l’heure, je n’avais pas définis de hauteur (“height”) à mon conteneur. Et par défaut, les éléments de type “bloc” prennent la hauteur de leur contenu, donc nous avons un carré parfait.

# Flexbox - align-items



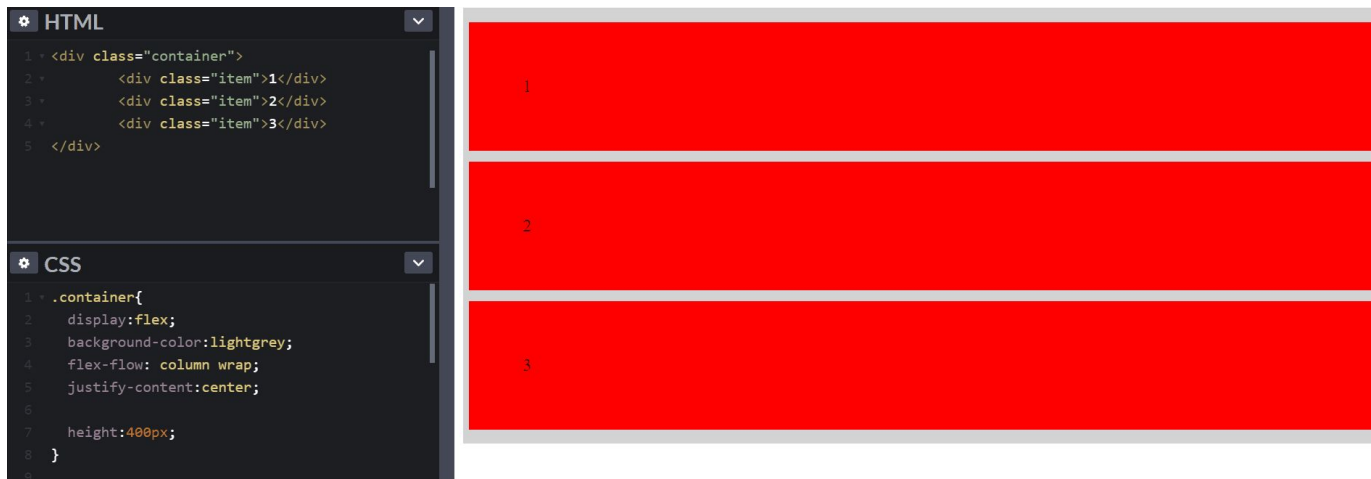
Pour avoir de nouveau nos jolis petits carrés, il faut leur mettre une autre valeur que “stretch”. Dans l’exemple ci-dessus, “flex-start”. Et comme on parle d’axe secondaire et que je suis en horizontal (“row”), cela les positionnent en haut du container.

# Flexbox - align-items



Ici, j'ai mis le "align-items: center" (centré)

# Flexbox - align-items



Du coup, même combat pour le “flex-direction: column” à la différence près que l’axe secondaire n’est plus vertical mais horizontal.

# Flexbox - align-items



A votre avis, quel est le “flex-direction” pour obtenir ces résultats ?





# Flexbox - align-content

**Align-content : flex-start | flex-end | center | space-around | space-between**

La propriété CSS **align-content** définit la façon dont l'espace est réparti entre et autour des éléments le long de l'axe en bloc du conteneur (c'est-à-dire l'axe orthogonal à l'axe d'écriture) lorsque celui-ci est [un conteneur de boîte flexible](#) et le long de l'axe principal lorsque le conteneur est une grille.

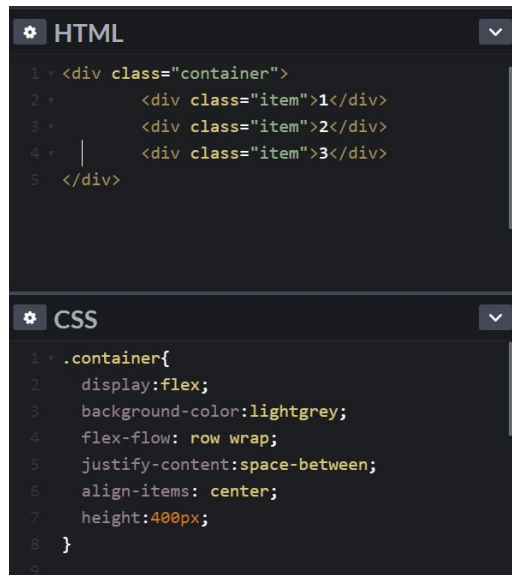
Source : <https://developer.mozilla.org/fr/docs/Web/CSS/align-content>







# Flexbox



Jusqu'à présent nous avons vu comment gérer nos éléments uniquement depuis son parent et de façon groupé. Cependant, il est possible de manipuler un élément spécifique en le sélectionnant directement

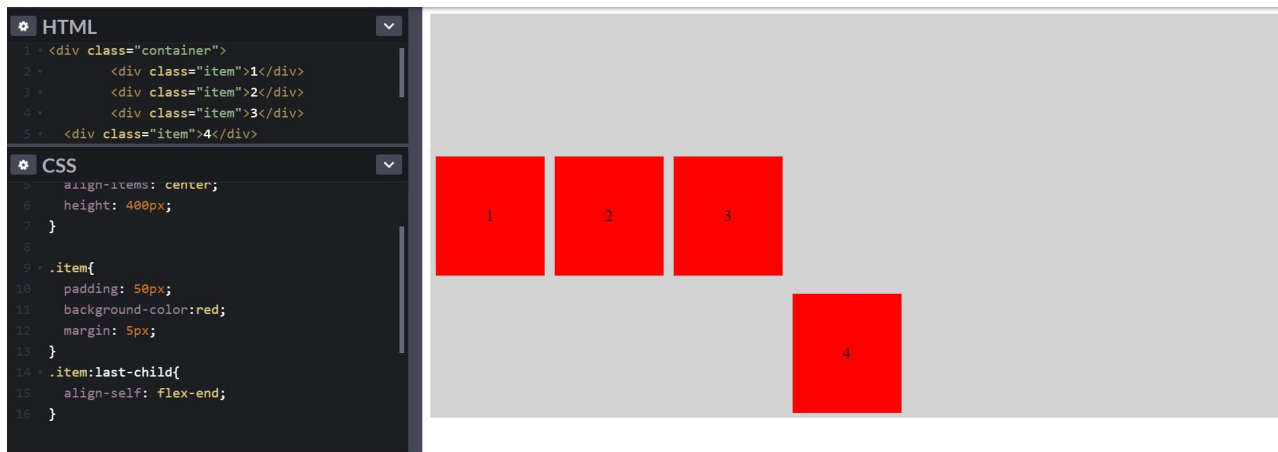
# Flexbox - align-self

**Align-self** : stretch (défaut) | flex-start | center | flex-end

La propriété CSS **align-self** permet d'aligner les objets flexibles d'une ligne flexible ou d'une grille en surchargeant la valeur donnée par [align-items](#).

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/align-self>

# Flexbox



Dans l'exemple ci-dessus, nous avons positionné le dernier élément en “flex-end” alors que ses frères sont en “center”. Cette mise en page est possible car le dernier enfant à la propriété “align-self” qui a un effet uniquement sur lui.





# Flexbox - order

## Order : <int>

La propriété **order** définit l'ordre avec lequel on dessine les éléments d'un conteneur d'éléments flexibles ou d'une grille d'éléments. Les éléments sont appliqués dans l'ordre croissant des valeurs de **order**. Les éléments ayant la même valeur pour **order** seront appliqués dans l'ordre selon lequel ils apparaissent dans le code source du document..

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/order>

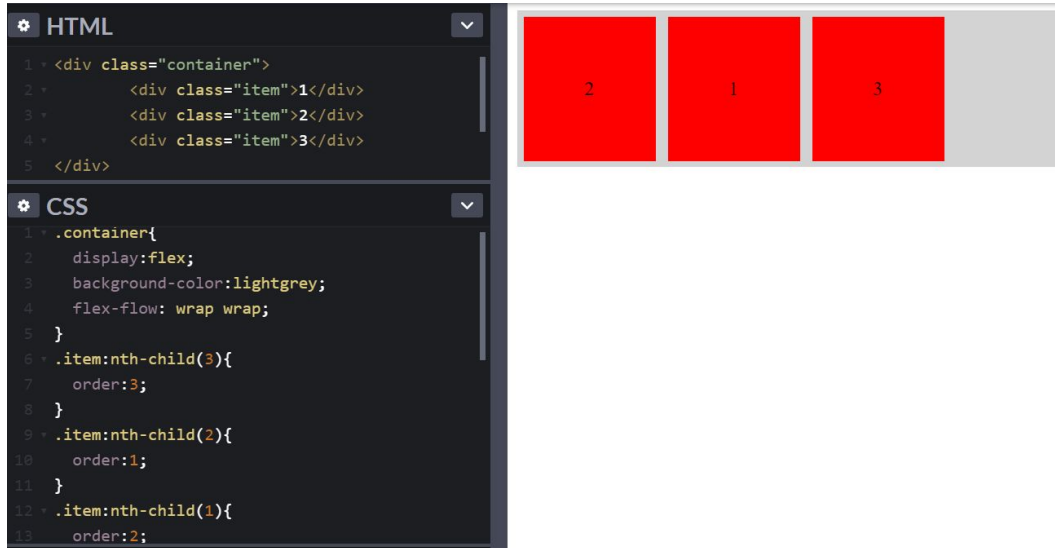
# Flexbox - order



Dans l'exemple ci-dessus, j'ai mis un "order" à 1 à l'élément numéro 2. Mais dans le rendu, on constate qu'il est passé en deuxième position....

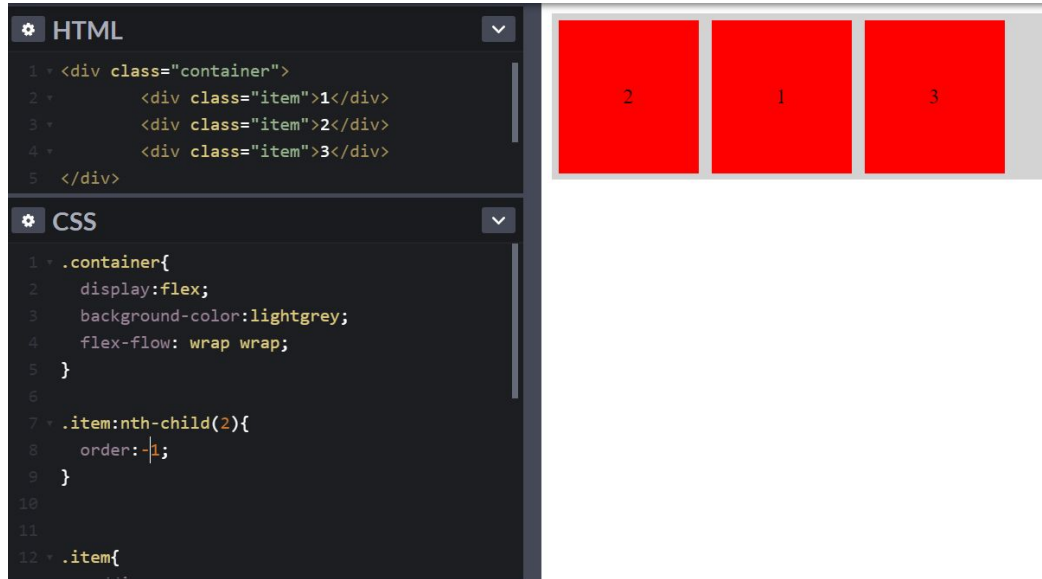
Par défaut, tous les éléments ont un "order" à 0, donc 1 étant supérieur à 0 et 0, il se retrouve en dernière position.

# Flexbox - order



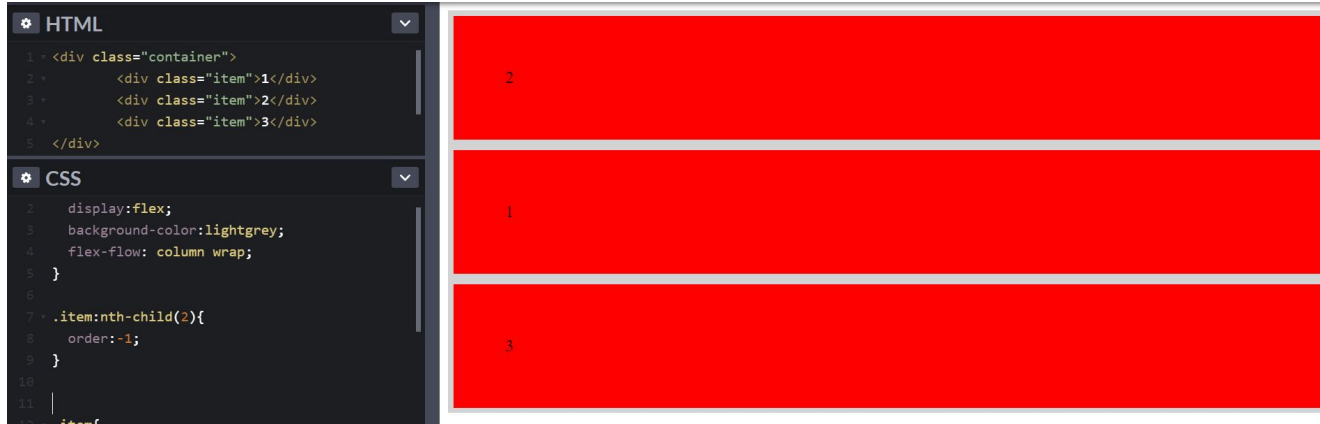
Si l'on souhaite un ordre précis, il sera peut-être nécessaire de mettre un "order" sur tous les éléments pour ensuite leur définir l'ordre voulu.

# Flexbox - order



Toujours dans mon exemple, si je souhaitais mettre l'élément 2 en premier sans forcément mettre un "order" sur tous les éléments, je peux mettre un "order" à -1. Les autres éléments étant à 0, l'élément 2 passera devant....

# Flexbox - order



Ca marche aussi pour les colonnes.

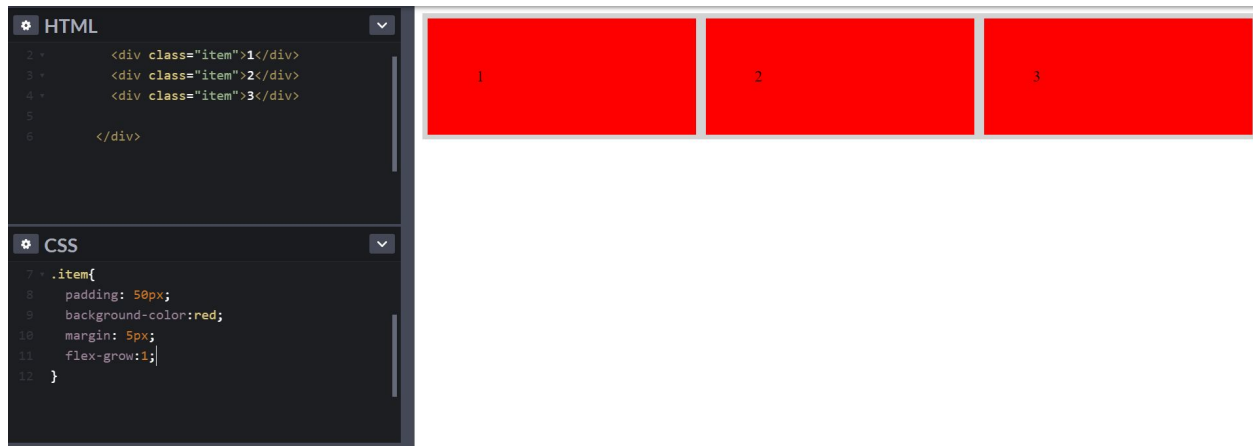
# Flexbox - flex-grow

**Flex-grow : <int>**

La propriété CSS **flex-grow** définit le facteur d'expansion d'un élément flexible selon sa dimension principale. Elle indique la quantité d'espace restant que l'élément devrait consommer dans un conteneur flexible relativement à la taille des autres éléments du même conteneur.

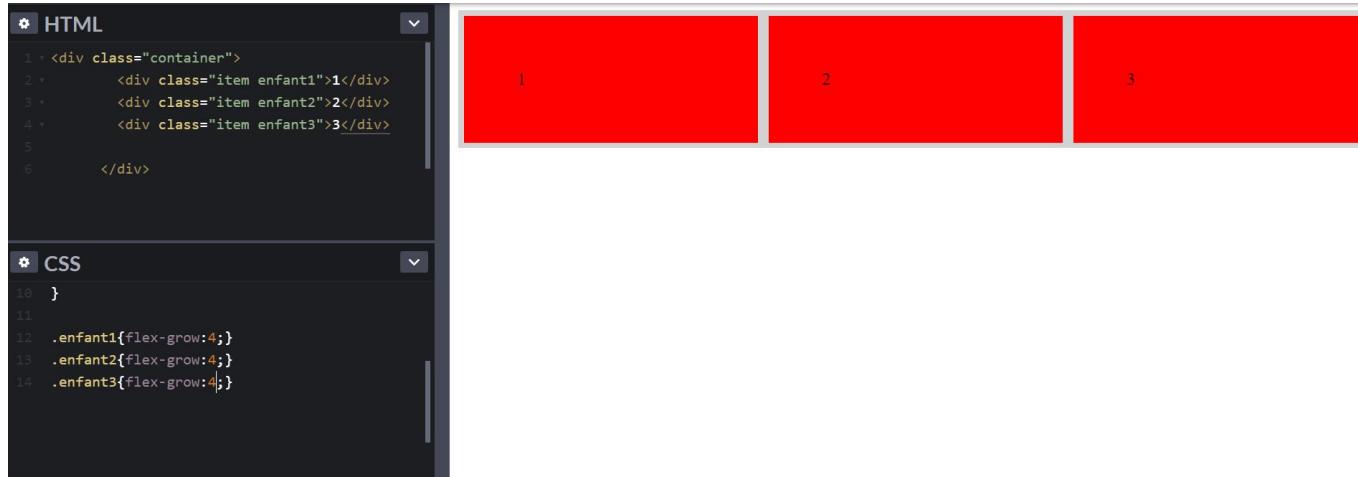
Source : <https://developer.mozilla.org/fr/docs/Web/CSS/flex-grow>

# Flexbox - flex-grow



Tout “order” , “flex-grow” ne s’applique pas sur le parent mais bien sur les enfants. Dans l’exemple, j’ai mis un “flex-grow” de 1 à tous mes éléments. En gros, je leur demande d’occuper tout l’espace disponible mais de façon équitable.

# Flexbox - flex-grow



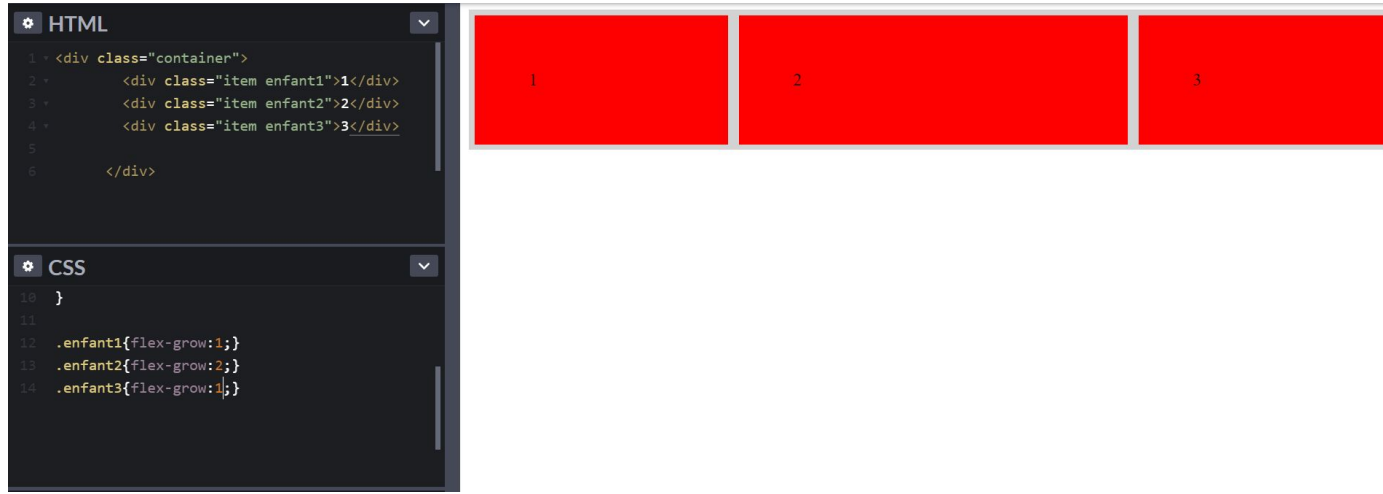
De même que si je mets un “flex-grow” à 4 à tous les éléments enfants, cela ne changera rien (par rapport à “flex-grow:1”).

Comme ils ont le même “flex-grow”, ils auront le même espace alloué.

Le “flex-grow” va se baser sur 100% de place, qu’il va diviser par le nombre d’enfant, puis faire  $100/3$  soit 33.3333333333%

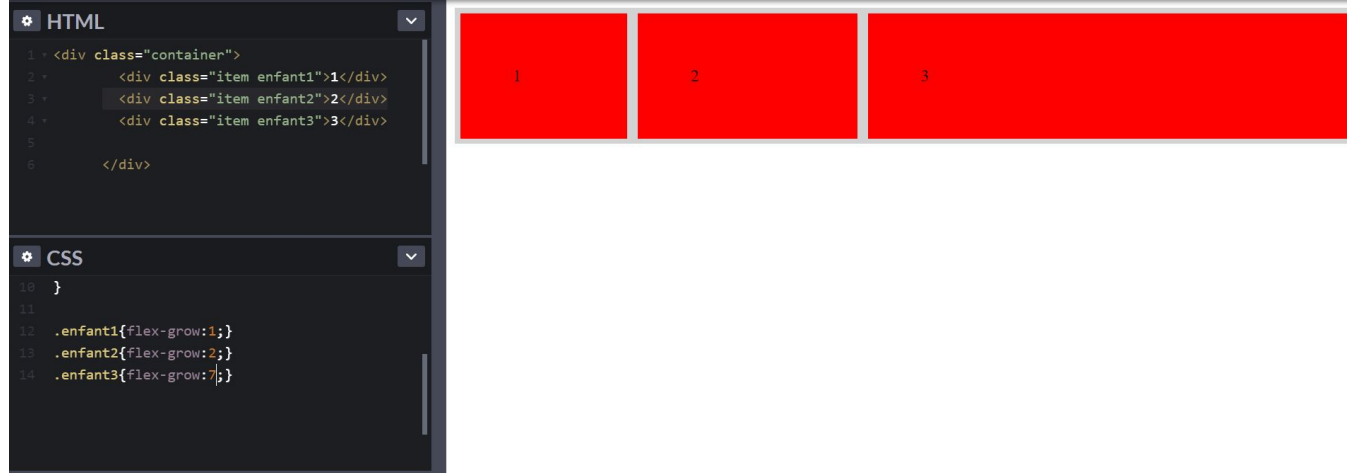


# Flexbox - flex-grow



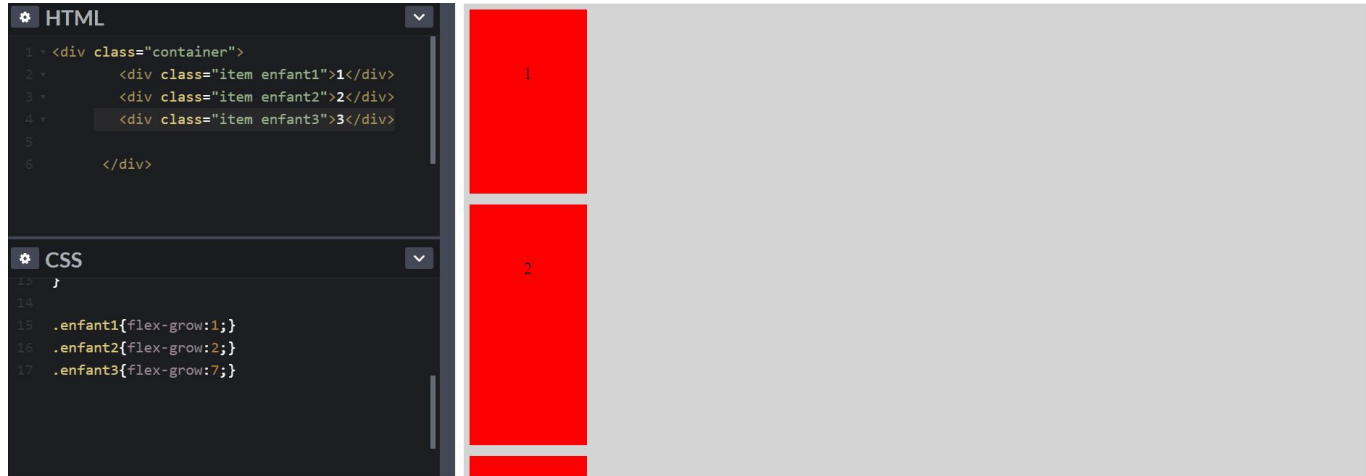
Dans ce cas là, l'enfant 2 à un “flex-grox” de 2 alors que les enfants 1 et 3 ont un “flex-grow” à 1. L'enfant 2 n'est pas 2 fois plus grand que les autres mais va prendre 1 unité de plus que ses voisins.

# Flexbox - flex-grow



Encore un exemple....

# Flexbox - flex-grow



Et en colonne aussi, ça fonctionne.

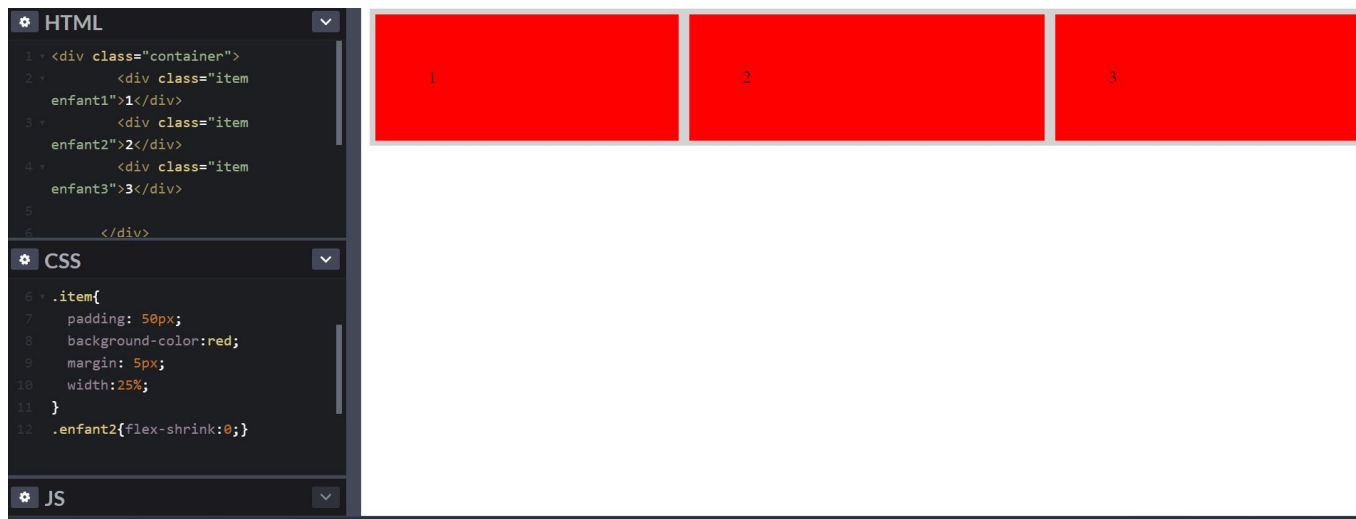
# Flexbox - flex-shrink

## Flex-shrink : <int>

La propriété **flex-shrink** définit le facteur de rétrécissement d'un élément flexible. Si la taille de l'ensemble des éléments flexibles est supérieure à la taille du conteneur, les éléments seront comprimés selon leur facteur **flex-shrink**.

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/flex-shrink>

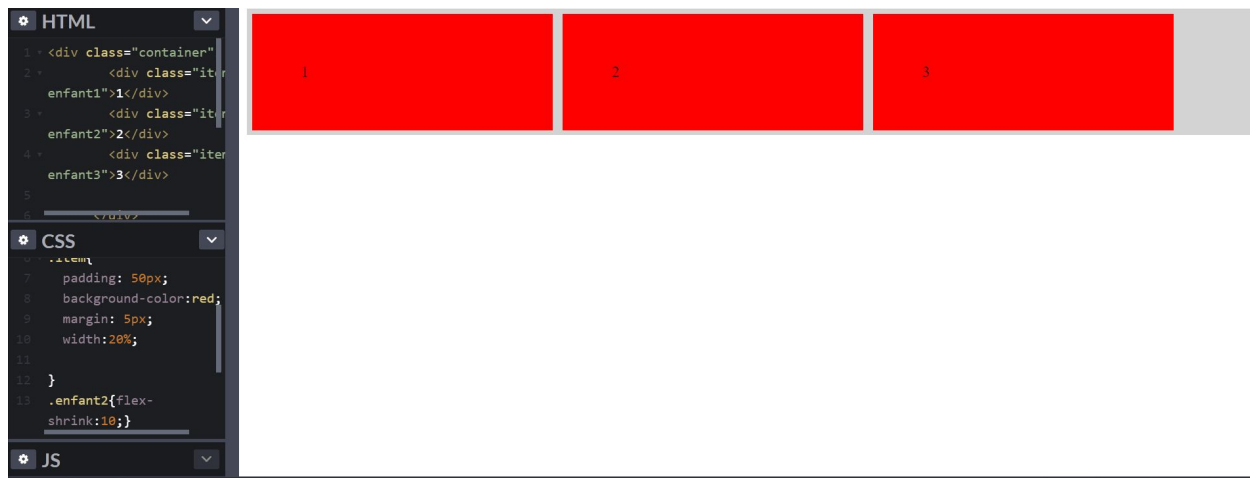
# Flexbox - flex-shrink



Le “flex-shrink” permet de gérer le rétrécissement des éléments. Par défaut (et contrairement au “flex-grow” qui est de 0) il est de 1. Si je mets un “flex-shrink” à 0 à 1 élément, je lui dis de ne pas rétrécir (et de rester proportionnel à la taille de l’écran).

Pour l’exemple, je suis en “flex-flow: row nowrap” (donc par défaut) et j’ai mis une largeur de 25% à chaque élément

# Flexbox - flex-shrink



En revanche, si je mets un “flex-shrink” à 10, je demande à l’enfant 2 de rétrécir 10 fois plus vite que ses frères (ou sœurs). Il faut garder en tête que c’est la largeur par rapport aux autres éléments et non la largeur de l’écran.

# Flexbox - flex-basis

## Flex-basis : <int>

La propriété **flex-basis** détermine la base de flexibilité utilisée comme taille initiale principale pour un élément flexible. Cette propriété détermine la taille de la boîte de contenu sauf si une autre boîte est visée par [box-sizing](#).

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/flex-basis>

# Flexbox - flex-basis

“flex-basis” ne s'applique qu'aux éléments “flex”. Les conteneurs flex (qui ne sont pas également des éléments “flex”) ignorent “flex-basis” mais peuvent utiliser la largeur et la hauteur.

“flex-basis” ne fonctionne que sur l'axe principal. Par exemple, si vous êtes dans “flex-direction : column”, la propriété “width” sera nécessaire pour dimensionner les éléments “flex” horizontalement.

“flex-basis” n'a aucun effet sur les éléments flex absolument positionnés. Les propriétés “width” et “height” seraient nécessaires. Les éléments “flex” absolument positionnés ne participent pas à la mise en page “flex”.

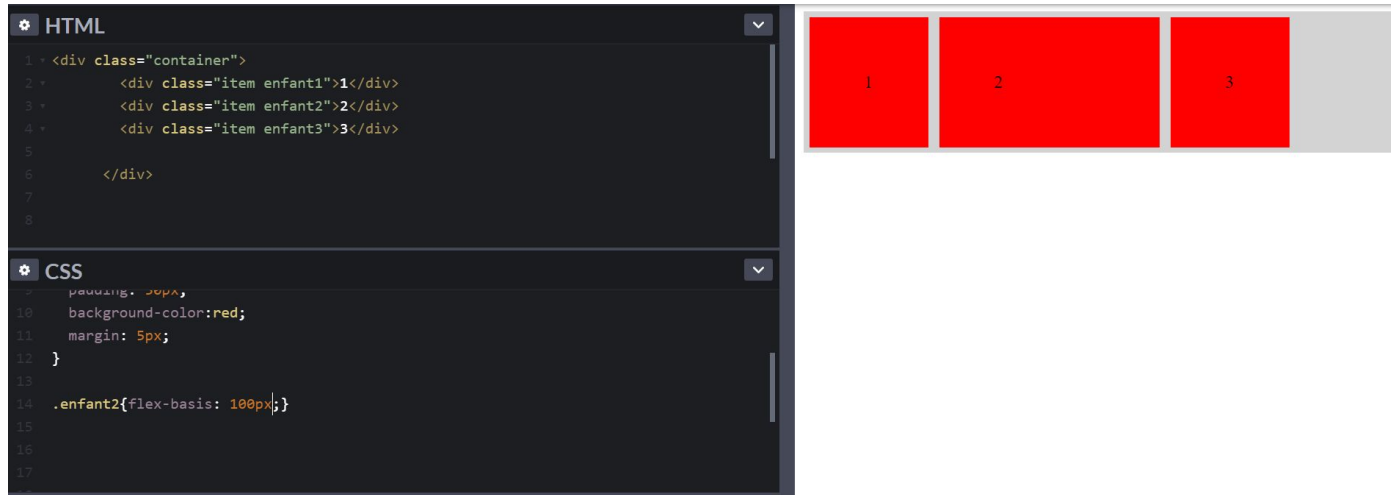
En utilisant la propriété “flex”, trois propriétés - flex-grow, flex-shrink et flex-basis - peuvent être combinées en une seule déclaration. En utilisant la largeur, la même règle nécessiterait plusieurs lignes de code.



# Flexbox - flex-basis

```
flex-grow : 1 ; /* il peut croître */  
flex-shrink : 1 ; /* il peut rétrécir */  
flex-basis : 225px ; /* égal à min-width */
```

# Flexbox - flex-basis



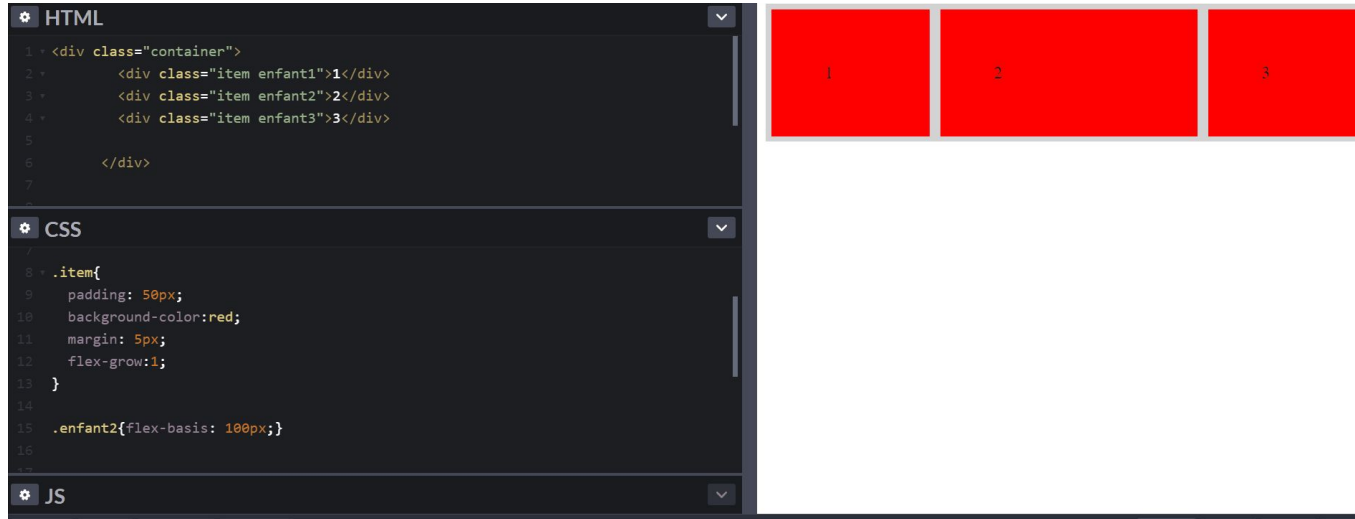
Si j'applique un "flex-basis" à 200px à mon deuxième élément, on voit qu'il est plus grand que les deux autres.

# Flexbox - flex-basis



Si je passe en “flex-direction: column” on voit bien que le “flex-basis” s’applique sur l’axe principal.

# Flexbox - flex-basis



Si je mets un “flex-grow” de 1 à tous les éléments, je dis que je souhaite qu’ils occupent TOUT l’espace disponible de manière égal mais j’ai également donné un “flex-basis” de 100 pixels (comme un “min-width”) à l’élément 2. On peut constater qu’il est légèrement plus grand que ses frères. C’est parce-que l’enfant 2 va prendre 100px plus le reste de largeur qu’il a besoin pour remplir l’espace restant du conteneur.

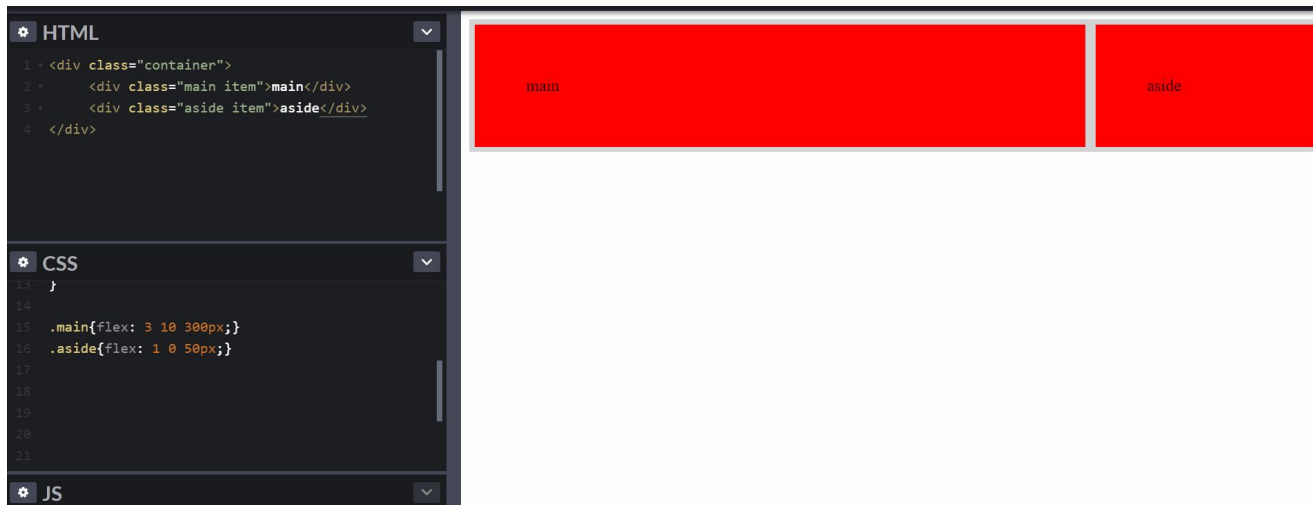
# Flexbox - flex

**Flex: <flex-grow> <flex-shrink> <flex-basis>**  
**(Défaut flex: 0 1 auto;)**

La propriété **flex** est une propriété raccourcie qui définit la capacité d'un élément flexible à modifier ses dimensions afin de remplir l'espace disponible de son conteneur. Les propriétés détaillées correspondantes à cette propriété raccourcie sont [flex-grow](#), [flex-shrink](#) et [flex-basis](#). Les éléments flexibles peuvent être étirés ou réduits pour utiliser un espace proportionnel à leur coefficient de grossissement ou de rétrécissement afin de ne pas dépasser d'un conteneur.

Source : <https://developer.mozilla.org/fr/docs/Web/CSS/flex>

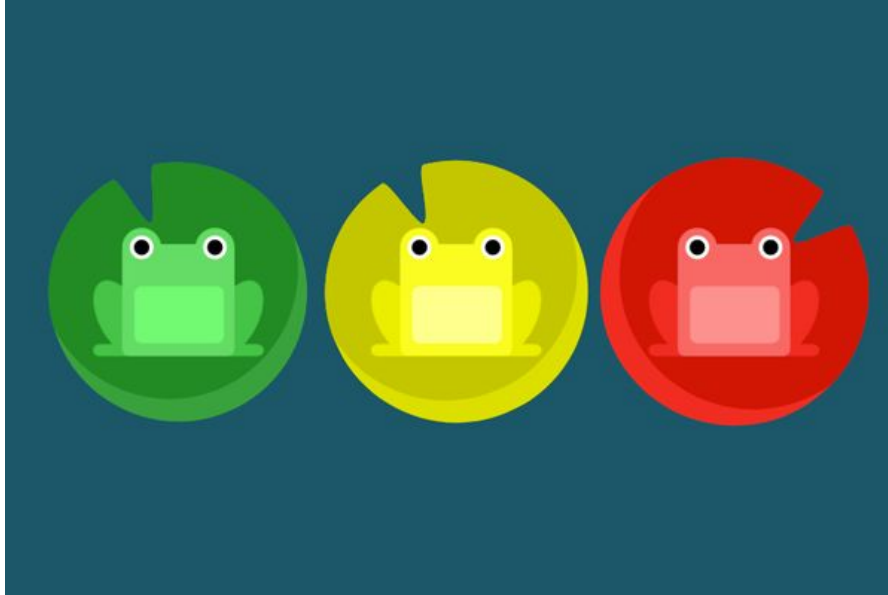
# Flexbox - flex



flex: 3 10 300px = flex-grow:3; flex-shrink:10; flex-basis:300px;

Ce qui veut dire que “main” a 2 unité de plus que “aside”, qu’il va se réduire 10 fois plus vite (en proportion de la taille de son frère) et qu’il a une taille minimum de 300 pixels.

# Flexbox - Jouons !



<https://flexboxfroggy.com/>