DIGITALHOUSE

Flexbox

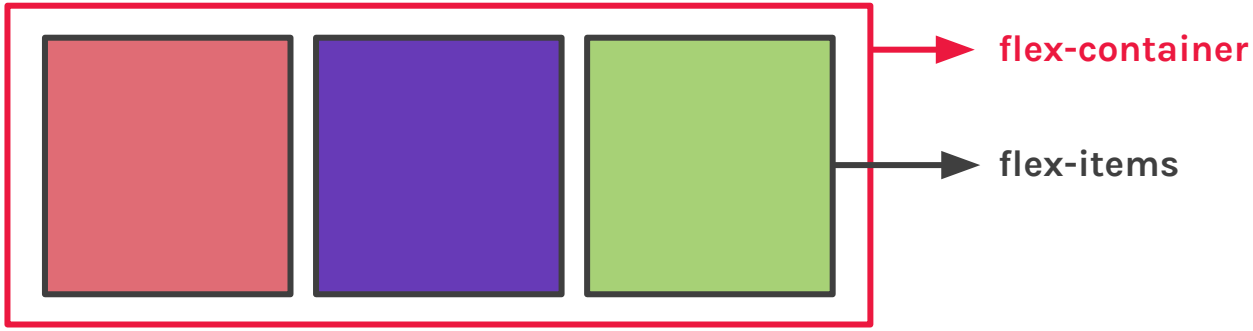
Agregar Flexbox

Para empezar a trabajar con **Flexbox** tenemos que definir un **flex-container**. Para eso usamos la propiedad **display** con el valor **flex**. De esta forma, estamos habilitando un contexto **flex**, para trabajar con los **hijos** directos del **elemento**. La propiedad **display** también puede recibir el valor **inline-flex**.

```css
.contenedor-padre {
    display: flex;
}
```

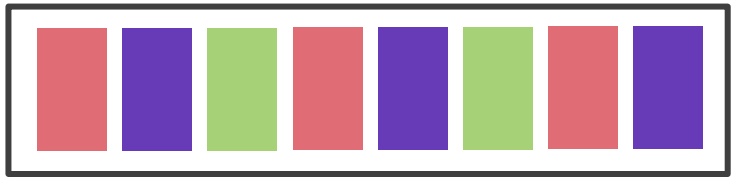
Estructura básica

Cuando hablamos de un **flex-container**, hablamos de un **elemento HTML** que contiene a **uno o más elementos**. A estos **elementos anidados** los llamamos **flex-items**. En el **flex-container** es en donde configuramos la mayoría de las propiedades **flex**.

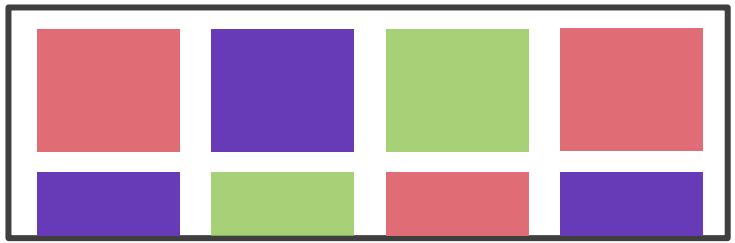


flex-wrap

Por defecto, los elementos hijos de un contenedor **flex** van a tratar de entrar todos en una misma línea.

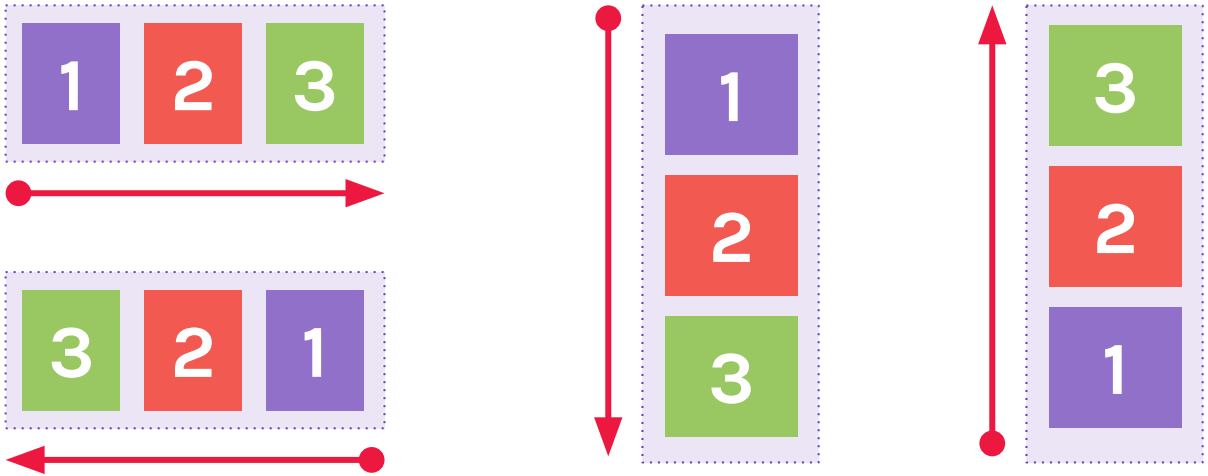


Para **aclararle** al **contenedor** que debe respetar el **ancho definido** de sus hijos usamos la propiedad **flex-wrap** con el valor **wrap**.



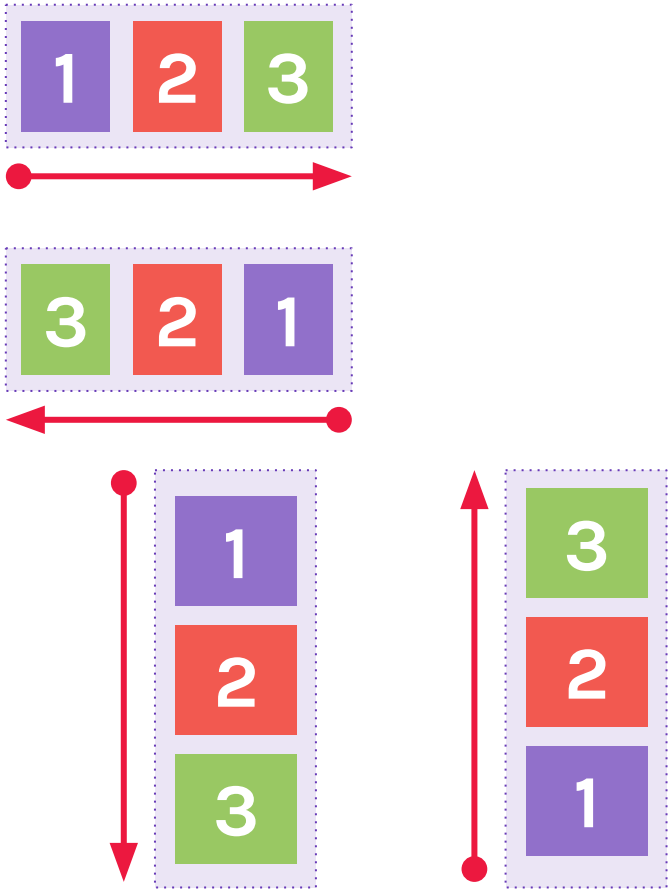
flex-direction

Con esta propiedad definimos el **main axis** (eje principal) del **contenedor**, que puede ser tanto **horizontal** como **vertical**. El **cross axis** (eje transversal) será la **dirección perpendicular** al **main axis**.



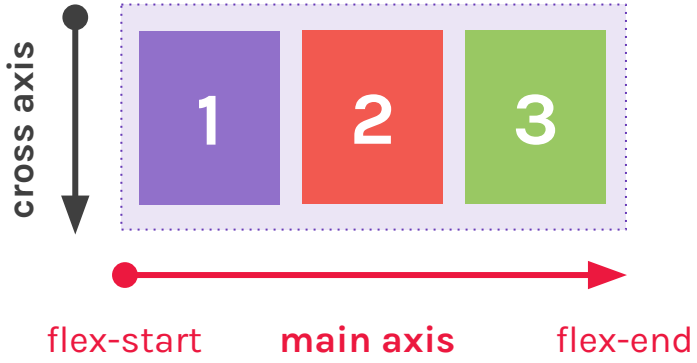
flex-direction

Con esta propiedad definimos el **main axis** (eje principal) del **contenedor**, que puede ser tanto **horizontal** como **vertical**. El **cross axis** (eje transversal) será la **dirección perpendicular** al **main axis.**



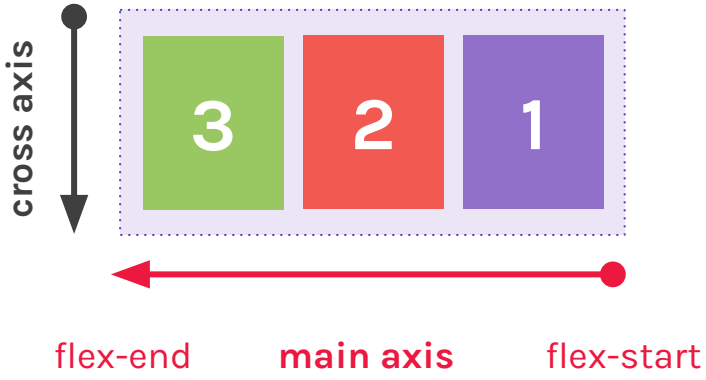
flex-direction: row

Con esta propiedad definimos el **main axis** (eje principal) horizontalmente y con el orden natural y **cross axis** (eje transversal) será la dirección perpendicular al main axis.



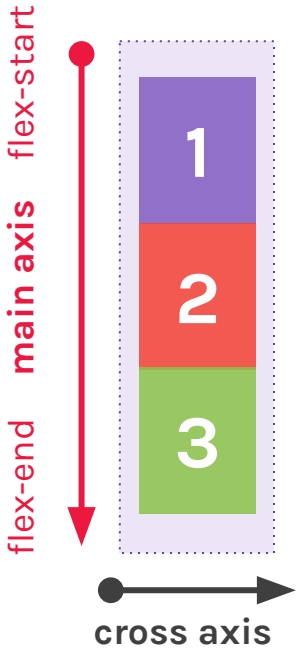
flex-direction: row-reverse

Los **ítems** se disponen en el **eje x**, de **derecha a izquierda**. En este caso, estamos invirtiendo el **inicio y fin** del **main-axis.**



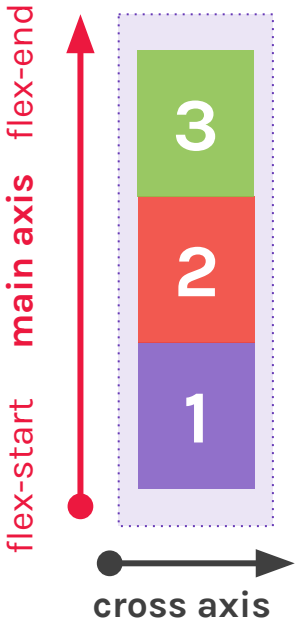
flex-direction: column

Los ítems se disponen en el eje y, de **arriba hacia abajo**.



flex-direction: column-reverse

Los ítems se disponen en el **eje y**, de **abajo hacia arriba**.
En este caso, estamos **invirtiendo el inicio y fin** del **main-axis.**

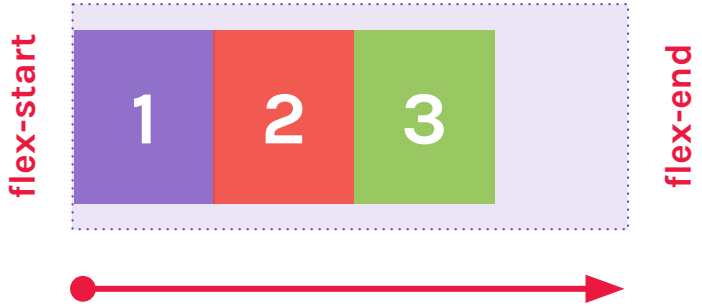


justify-content

Con esta propiedad **alineamos** los **ítems** a lo largo del **main axis**. Si es **horizontal**, se alinearán en función de la **fila**. Si es **vertical**, se alinearán en función de la **columna**.

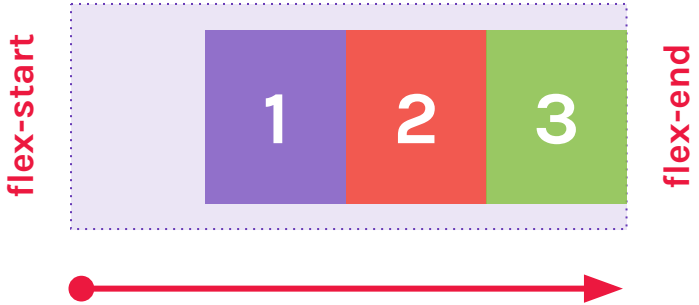
justify-content: flex-start

Los **ítems** se alinean respecto del **inicio** del **main axis** que hayamos definido.
Si no le aclaramos el **justify-content** al contenedor, **flex-start** es el valor por defecto.



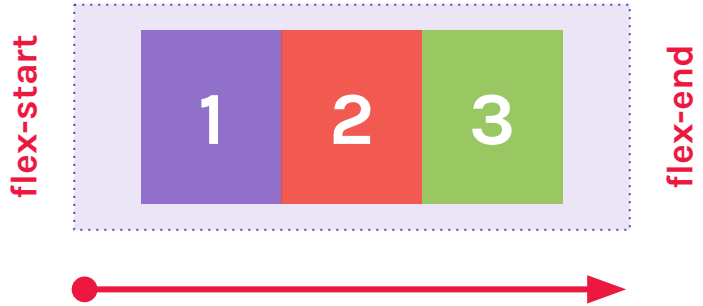
justify-content: flex-end

Los **ítems** se alinean respecto del **final** del **main axis** que hayamos definido.



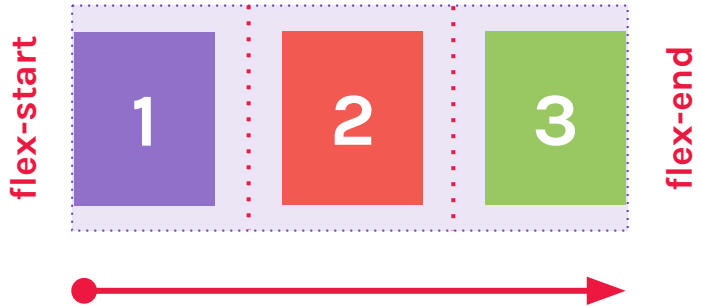
justify-content: center

Los **ítems** se alinean en el **centro** del **main axis.**



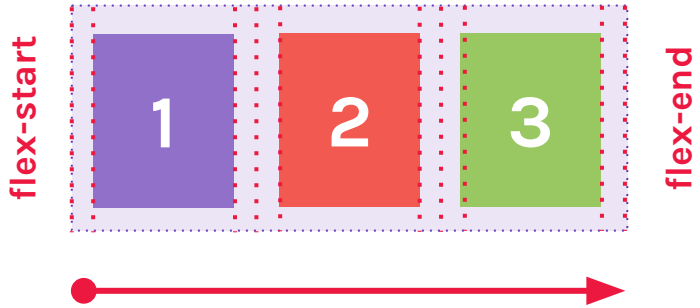
justify-content: space-between

Los **ítems** se distribuyen de manera **uniforme**. El **primer ítem** será enviado al **inicio** del **main axis**, y el **último ítem**, al **final**. El **espacio libre** se repartirá para separar los **ítems**.



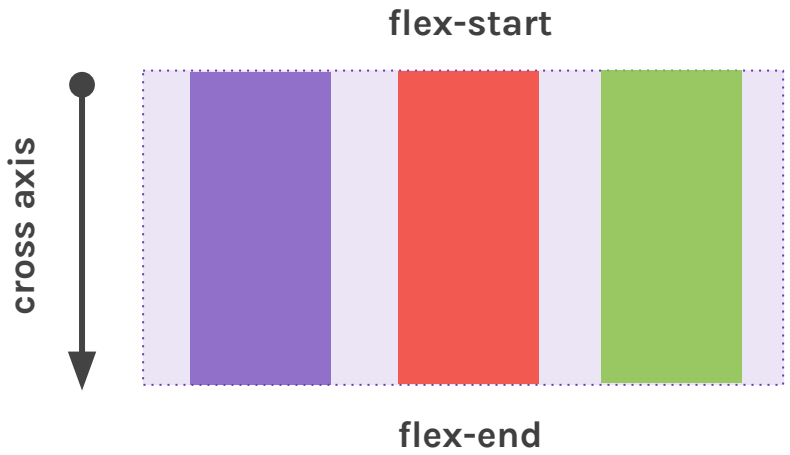
justify-content: space-around

Los **ítems** se distribuyen de manera **uniforme**. El **espacio libre** disponible se repartirá entre todos los elementos. Del espacio que le toque a cada elemento, la mitad irá a la **derecha** y la otra a la **izquierda** (o **arriba** y **abajo** en caso de que sean **columnas**).



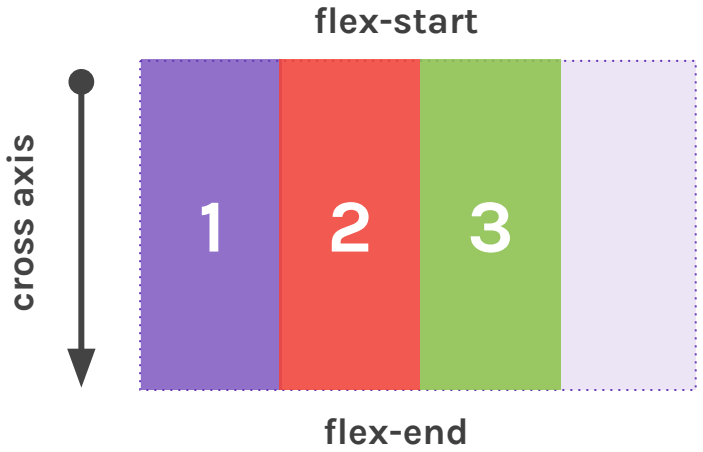
align-items

Con esta propiedad alineamos los **ítems** a lo largo del **cross axis**. Si no aclaramos esta propiedad, el valor por defecto es **stretch**, en otras palabras, los ítems ocuparán todo el **espacio disponible** en el **cross axis**.



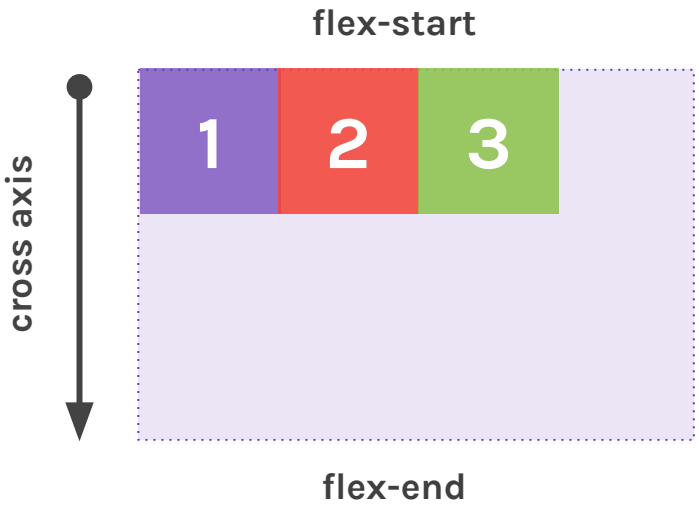
align-items: stretch

Los **ítems** se ajustan para abarcar todo el **contenedor**. Si el **cross axis** es **vertical**, se ajustan en función de la **columna**. Si el **cross axis** es **horizontal**, se ajustan en función de la **fila**.



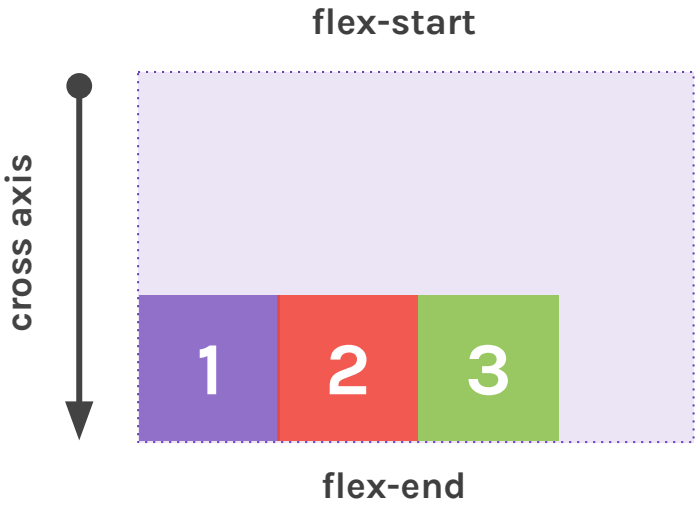
align-items: flex-start

Los **ítems** se alinean al **inicio** del **cross-axis**.



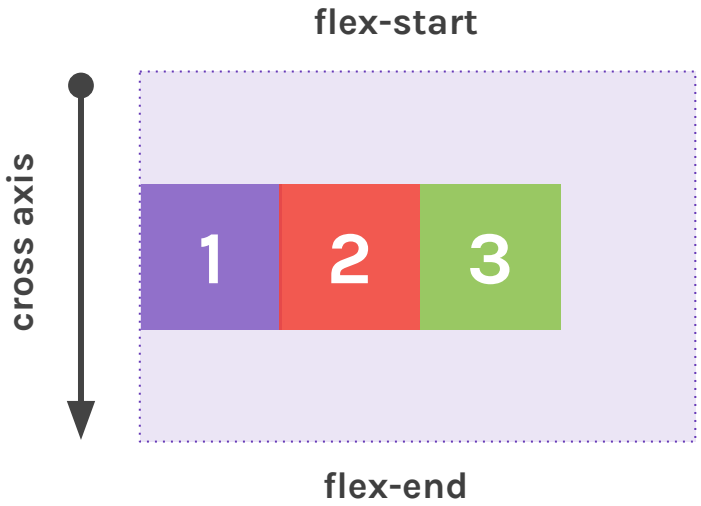
align-items: flex-end

Los **ítems** se alinean al **final** del **eje transversal**.



align-items: center

Los **ítems** se alinean al **centro** del **eje transversal**.



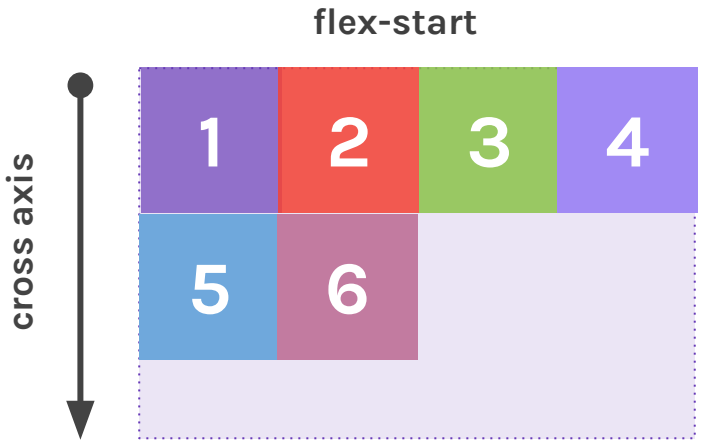
align-content

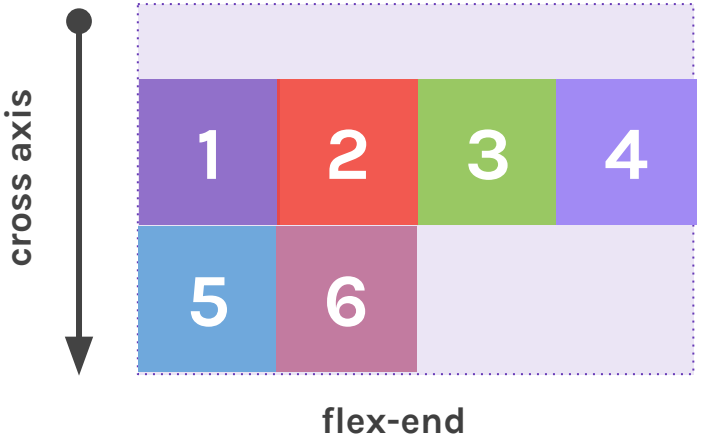
Si tenemos un **contenedor** de una **sola línea** *(donde flex-flow se establece como **no-wrap**)* utilizaremos **align-items**, pero en el caso de que estemos trabajando con contenedores **multilínea** debemos utilizar **align-content**.

Con esta propiedad alineamos los **ítems** a lo largo del **cross axis** cuando los contenedores flexibles incluyen de **varias líneas** *(donde flex-flow se establece en wrap o wrap-reverse).*

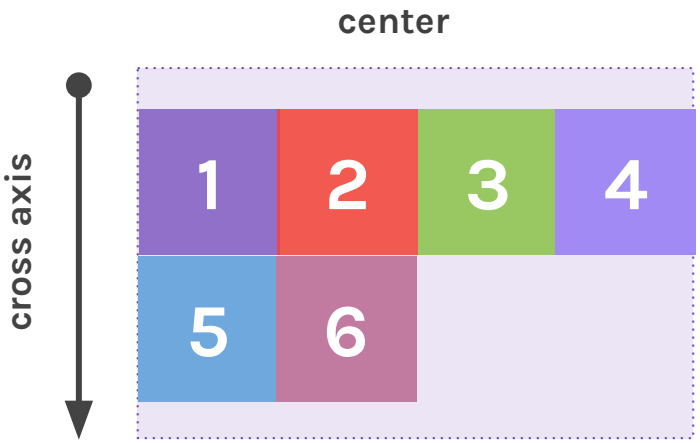
Los valores que admite la propiedad **align-content** son similares a los que podemos utilizar para **align-items**. Vale la pena probar como funcionan ambas propiedades para entenderlas bien.

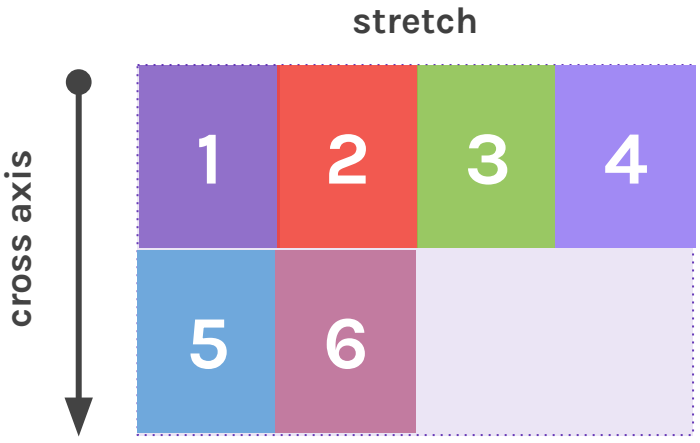
align-content: flex-start | flex-end



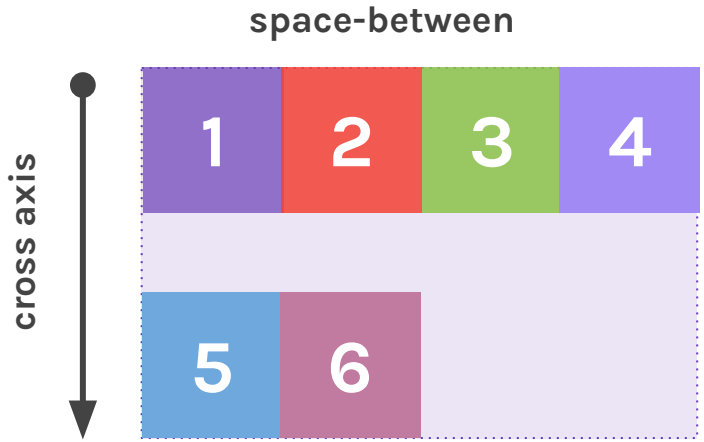


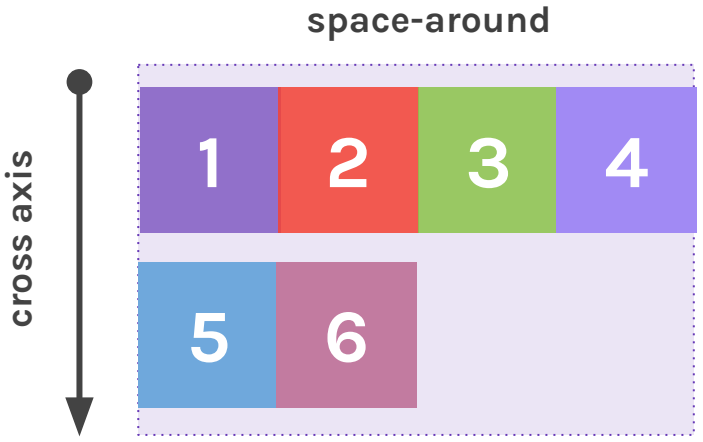
align-content: center | stretch





align-content: space-between | space-around





flex items: order

Con esta propiedad controlamos el **orden** de cada **ítem**, *sin importar el orden original* que tengan en la estructura HTML. Esta propiedad recibe un **número entero**, positivo o negativo, como valor.

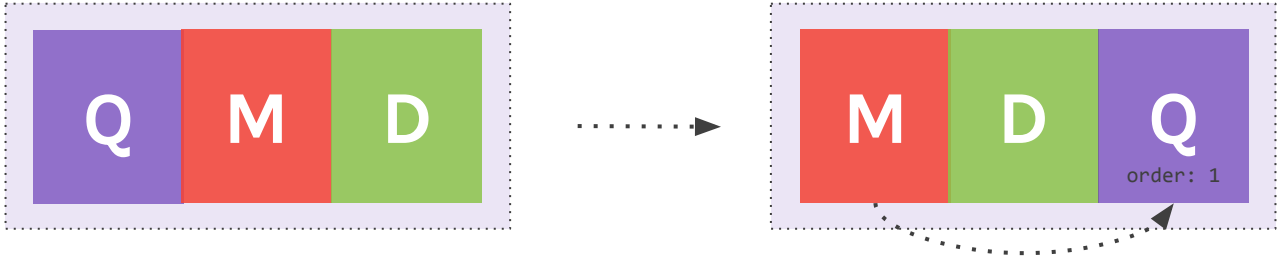
Por defecto, todos los ítems flex tienen un order: 0 implícito, aunque no se especifique.

```css
.caja {
    order: 1;
}
```

order: número positivo

Si le asignamos a la **caja Q** (que posee la clase caja-q) la propiedad **order** con valor **1**, esta pasará al final de la fila por ser el *número más alto*. Recordemos que, *por defecto*, el valor del orden de cada ítem es **0**.

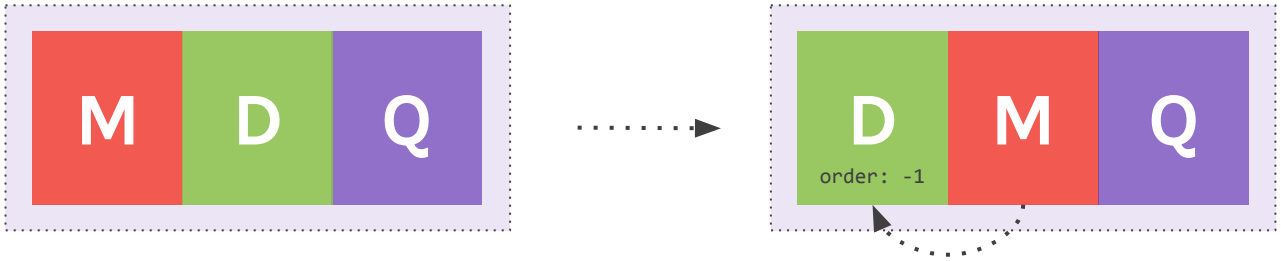
```css
.caja-q {
    order: 1;
}
```



order: número negativo

Si ahora le asignamos a la **caja D** la propiedad order con un **-1** como valor, esta pasará al comienzo de la fila. Colocando al ítem con el orden más pequeño primero.

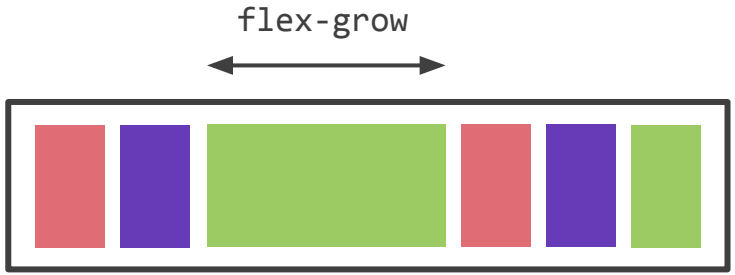
```css
.caja-d {
    order: -1;
}
```



flex-grow

Con esta propiedad definimos cuánto puede llegar a **crecer** un **ítem** en caso de disponer de **espacio libre** en el **contenedor**.

Configura un crecimiento flexible para el elemento.



flex-grow

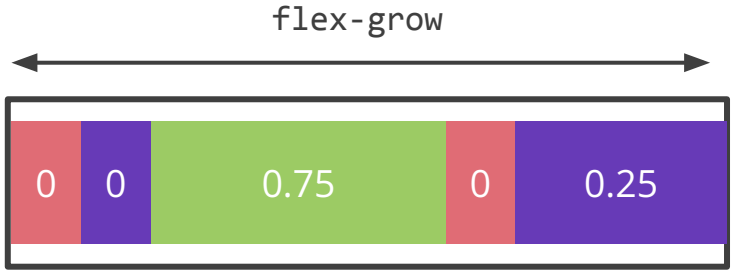
Si ambos **ítems** tienen la propiedad **flex-grow** con valor **1**, a medida que el contenedor se agrande, irán abarcando el espacio disponible en partes iguales.

```css
.caja-a, .caja-b {
    flex-grow: 1;
}
```

flex-grow

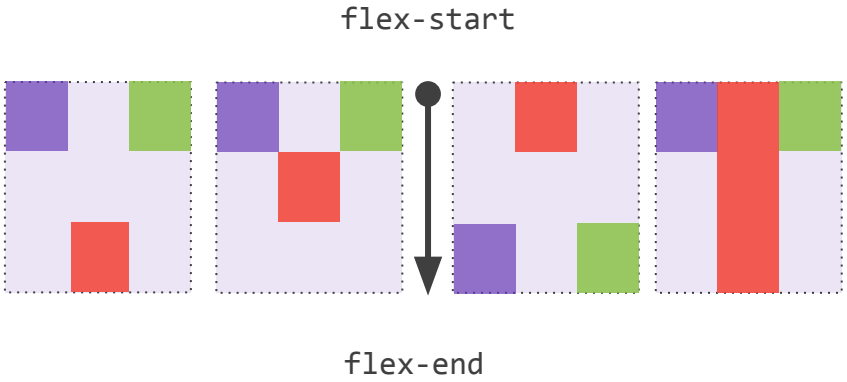
El número que le asignamos a **flex-grow** determina qué cantidad de espacio disponible dentro del contenedor flexible tiene que ocupar ese ítem.

1 equivale al **100%** del espacio disponible, y **0** al **0%**. Podemos usar cualquier valor en el medio, como **0.25** para el **25%**.



align-self

Nos permite **alinear**, sobre el **cross axis**, a cada **ítem** al que le apliquemos esta propiedad, *independientemente* de la alineación que se haya definido en el contenedor **flex** con **align-items**.



flex-grow

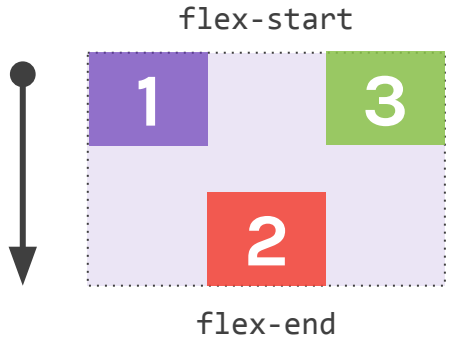
Si ambos **ítems** tienen la propiedad **flex-grow** con valor **1**, a medida que el contenedor se agrande, irán abarcando el espacio disponible en partes iguales.

```css
.caja-a, .caja-b {
    flex-grow: 1;
}
```

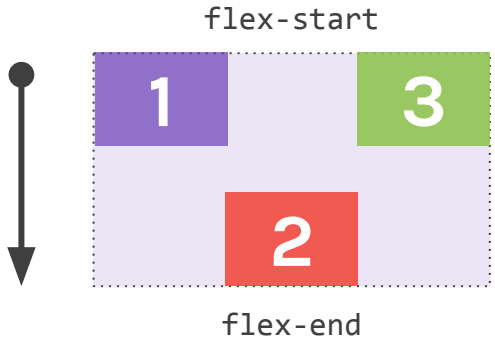
align-self: flex-end

Con **flex-end**, el ítem se alínea al final del eje transversal.

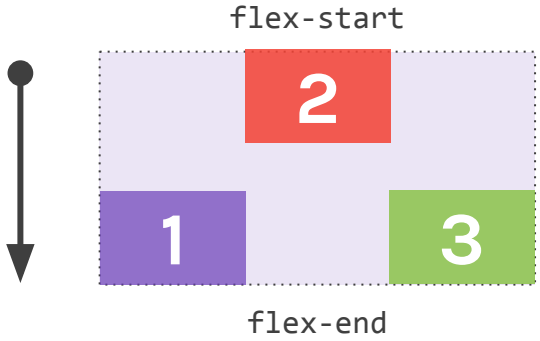
```css
.contenedor-padre {
    align-items: flex-start;
}
.caja-dos {
    align-self: flex-end;
}
```



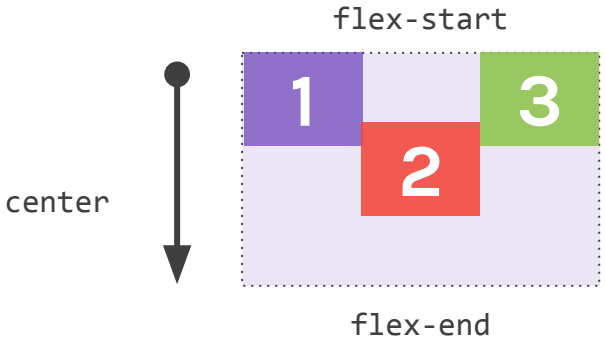
align-self: flex-end



align-self: flex-start



align-self: center



align-self: stretch