Los items en Flexbox

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Flexbox nos da la posibilidad de aplicarle **propiedades** directamente a cada **ítem** para poder **manipularlos** por **separado** y tener mayor control.

Índice

1 | order

order

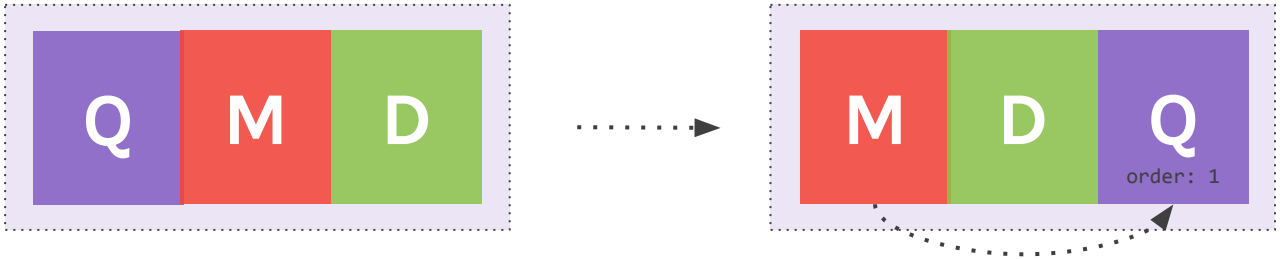
Con esta propiedad **controlamos** el **orden** de cada ítem, sin importar el orden original que tengan en la estructura HTML. Esta propiedad recibe un **número entero**, **positivo o negativo**, como **valor**. Por defecto, todos los ítems flex tienen un `order: 0` implícito, aunque no se especifique.

```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`.

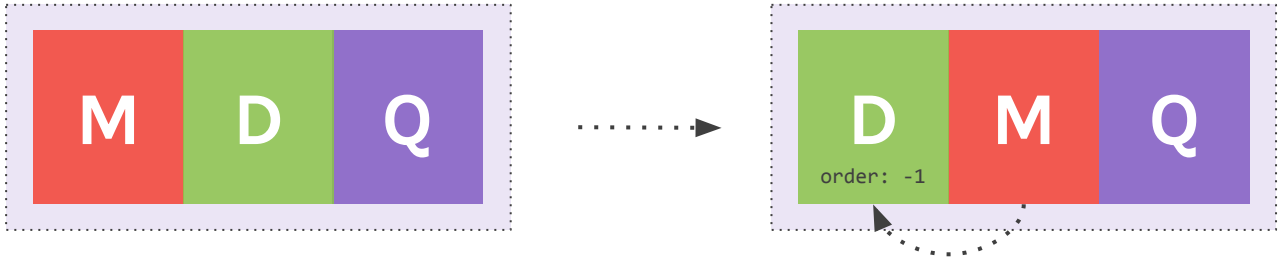
```css
.caja-q {
    order: 1;
}
```



order: número negativo

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

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

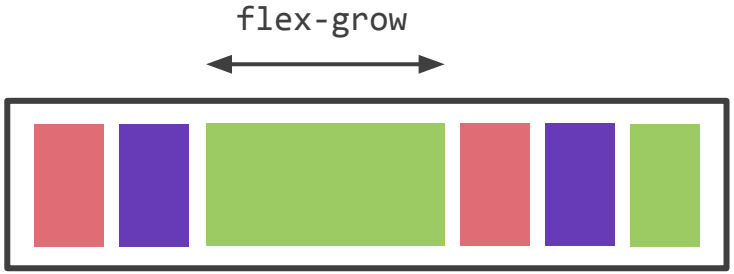


Las cajas se irán **ordenando respetando** la **secuencia** desde los números **negativos** hacia los **positivos**.

2 | flex-grow

flex-grow

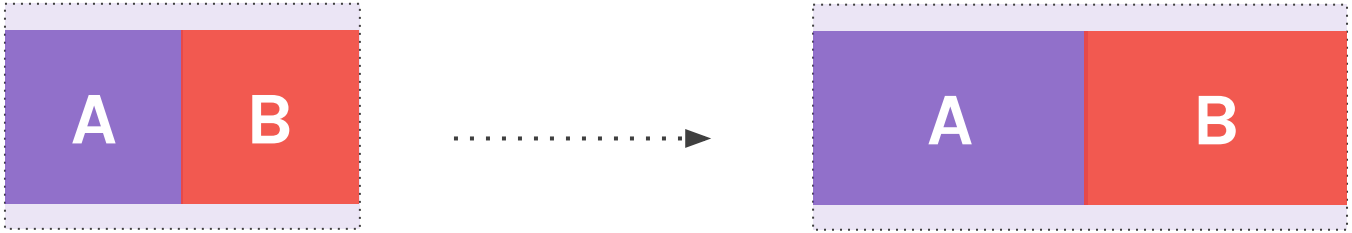
Con esta propiedad definimos cuánto puede llegar a **crecer** un **ítem** en caso de **disponer** de **espacio libre** en el contenedor. Configura un crecimiento flexible para el elemento.



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.

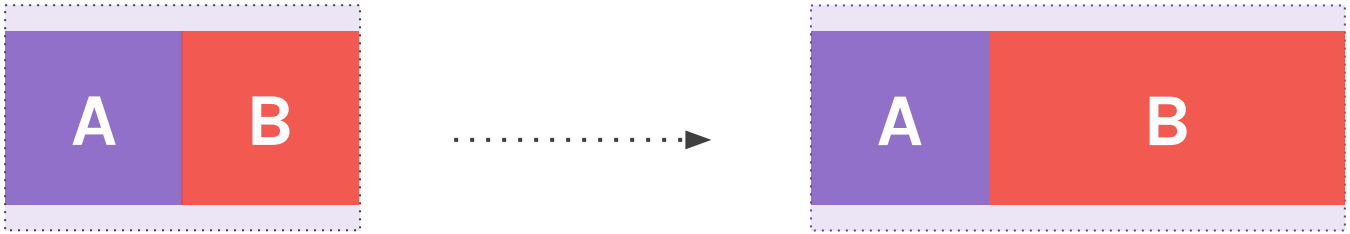
```css
.caja-a, .caja-b {
    flex-grow: 1;
}
```



flex-grow

Si un solo **ítem** tienen la propiedad `flex-grow`, este intentará ocupar el espacio libre disponible, a medida que el contenedor se agrande, según la proporción que definamos con el valor.

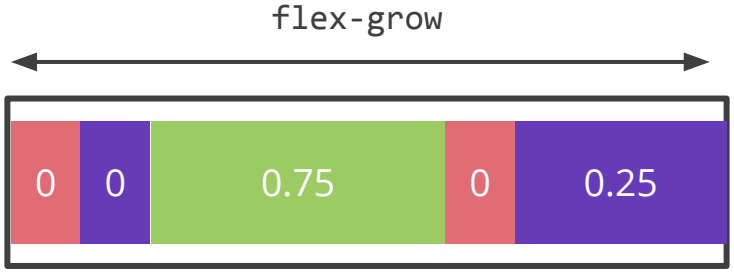
```css
.caja-b {
    flex-grow: 1;
}
```



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.

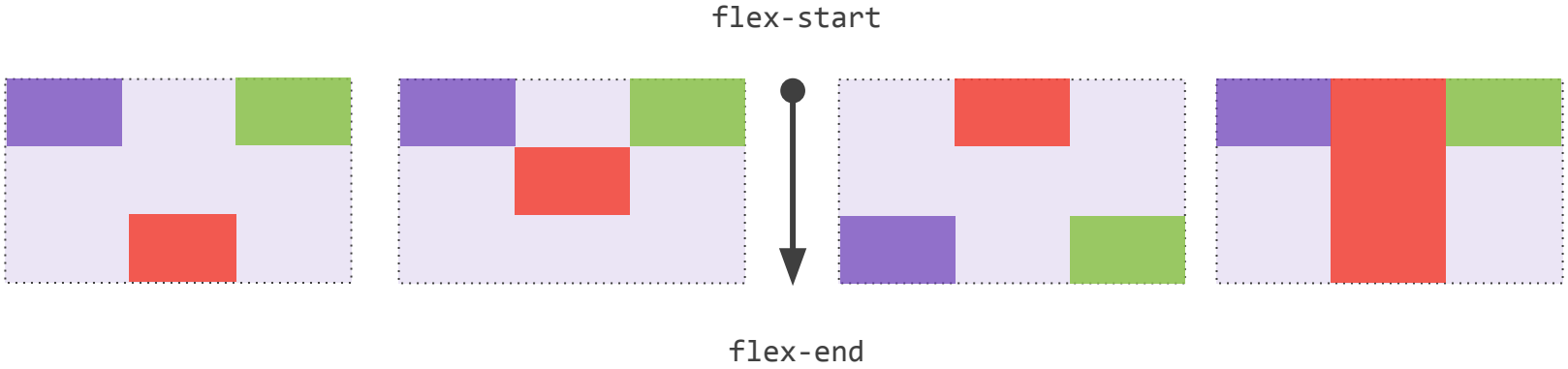
`1` equivale al 100% del espacio disponible, y `0` al 0%. Podemos usar cualquier valor en el medio, como `0.25` para el 25%.



3 | align-self

align-self

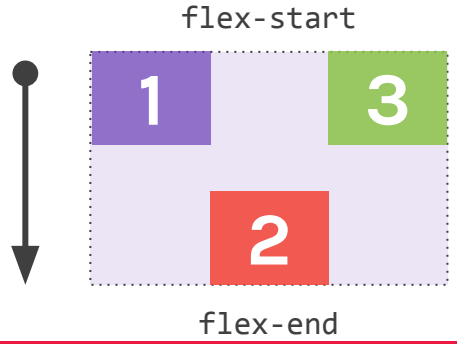
Nos permite **alinear**, sobre el **cross axis**, a cada ítem al que le **apliquemos** esta propiedad, independientemente de la **alineación** que se haya definido en el **contenedor flex** con **align-items**.



align-self: flex-end

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

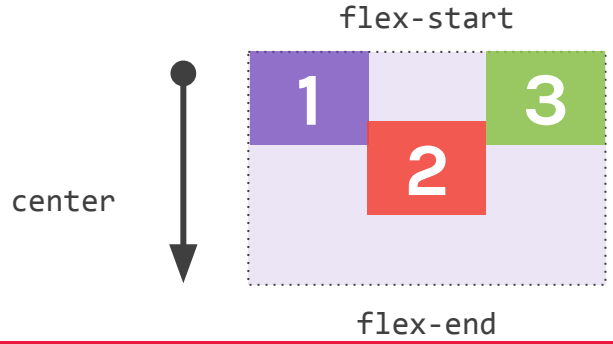
```css
.contenedor-padre {
    align-items: flex-start;
}
.caja-dos {
    align-self: flex-end;
}
```



align-self: center

Con `center`, el **ítem** se alínea al **centro** del eje transversal.

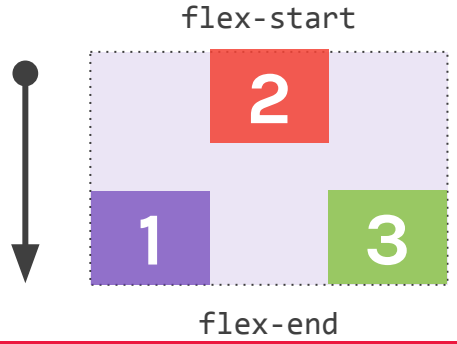
```css
.contenedor-padre {
    align-items: flex-start;
}
.caja-dos {
    align-self: center;
}
```



align-self: flex-start

Con `flex-start` el **ítem** se alínea al **inicio** del eje transversal.

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

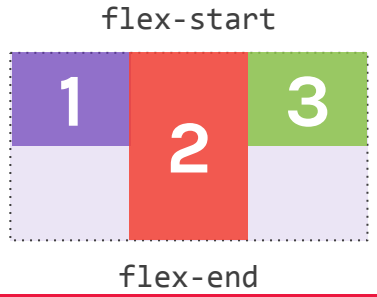


align-self: stretch

Con stretch, el ítem se **ajusta** hasta abarcar **todo** el **cross axis**, es el comportamiento por defecto.

Funciona siempre que el elemento no tenga definida una altura.

```css
.caja-uno, .caja-tres {
    align-self: flex-start;
}
.caja-dos {
    align-self: stretch;
}
```



“

Estas **propiedades aplicarán** para los flex-items siempre y cuando el contenedor **padre** sea un **flex-container**.

”


DigitalHouse>
Coding School