

# CURSO .CSS

## Flexbox



Autor: Jon Vadillo  
[www.jonvadillo.com](http://www.jonvadillo.com)

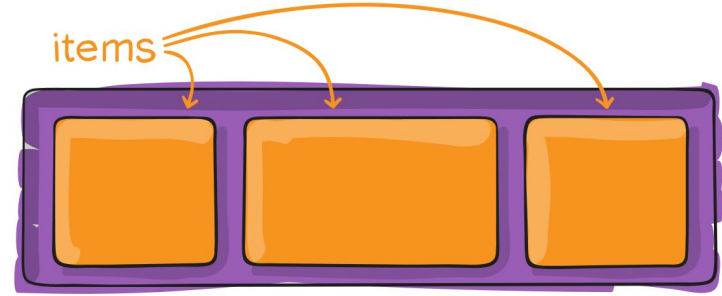
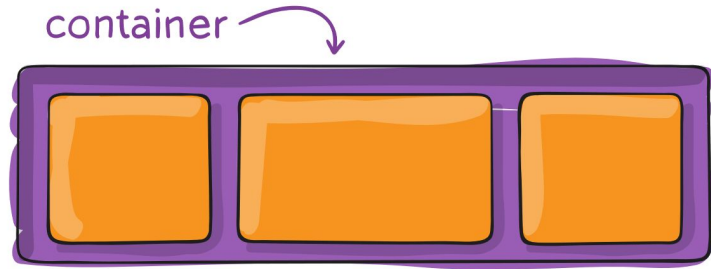
# Contenidos

- ¿Qué es Flexbox?
- Conceptos básicos: contenedor y sus elementos
- Propiedades del contenedor
- Propiedades de los elementos

# Flexbox

Método que pueda ayudar a alinear y distribuir el espacio entre los **elementos** (items) de un **contenedor** (container).

# Container & Items



# Container

Al asignar el valor `flex` o `inline-flex` a la propiedad `display`, los hijos directos de este contenedor se convierten en ítems flex.

## CSS

```
.container {  
  display: flex; /* or inline-flex */  
}
```

# Hands on!

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

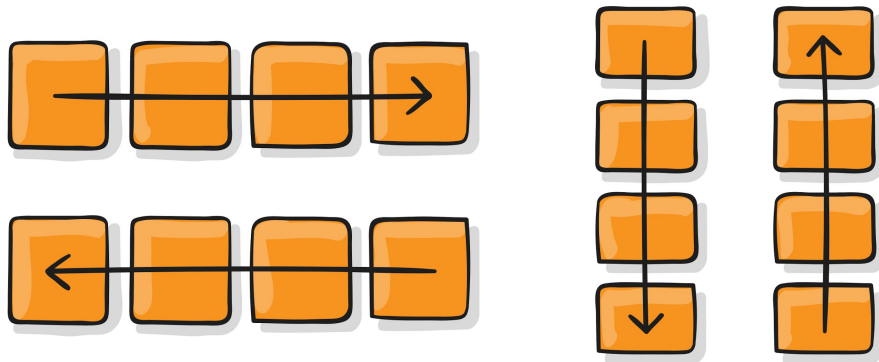
```
<div class="container">  
  <div class="item">1</div>  
  <div class="item">2</div>  
  <div class="item">3</div>  
  <div class="item">4</div>  
  <div class="item">5</div>  
</div>
```

# Propiedades del container

- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

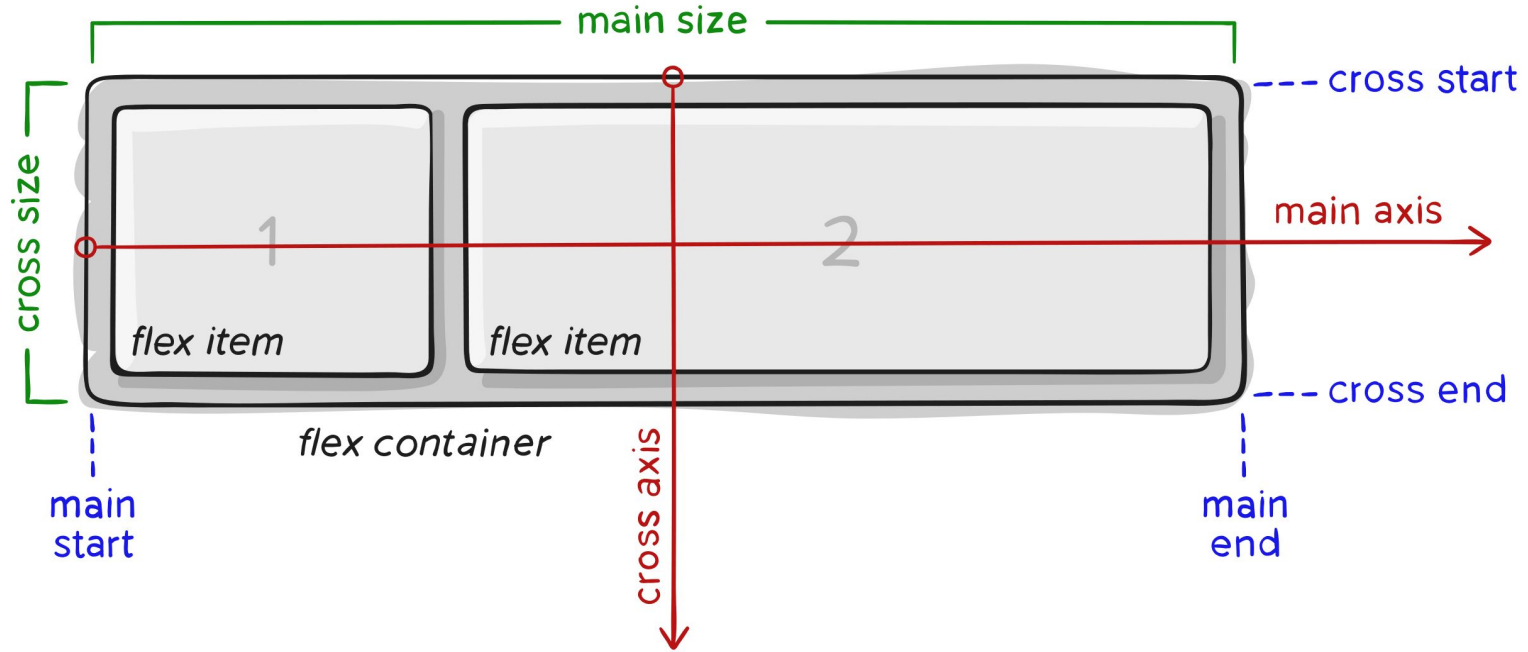
## Eje principal y eje cruzado

Todo lo que hacemos con flexbox está referido a estos dos ejes: el eje principal y el eje cruzado. El eje principal se define con la propiedad `flex-direction` y el cruzado será perpendicular a este.





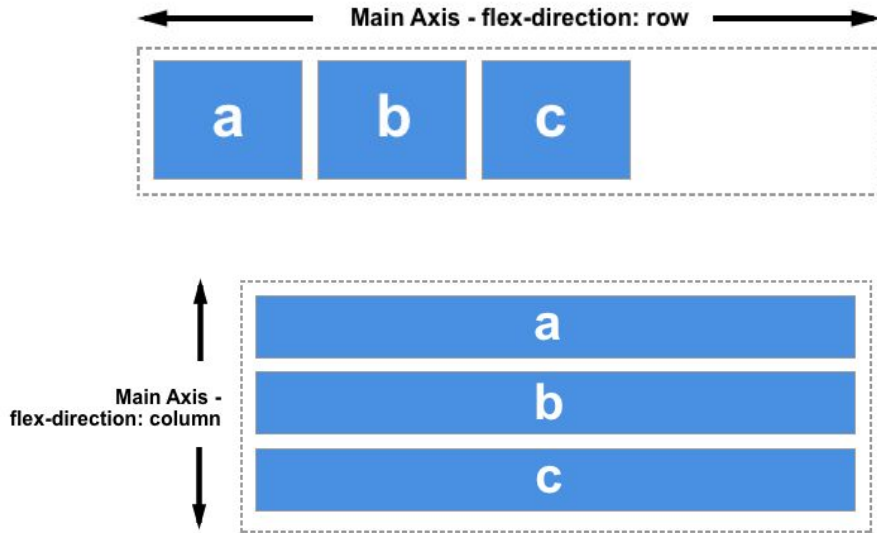
# Eje principal y eje cruzado



# flex-direction



# flex-direction



## Valores

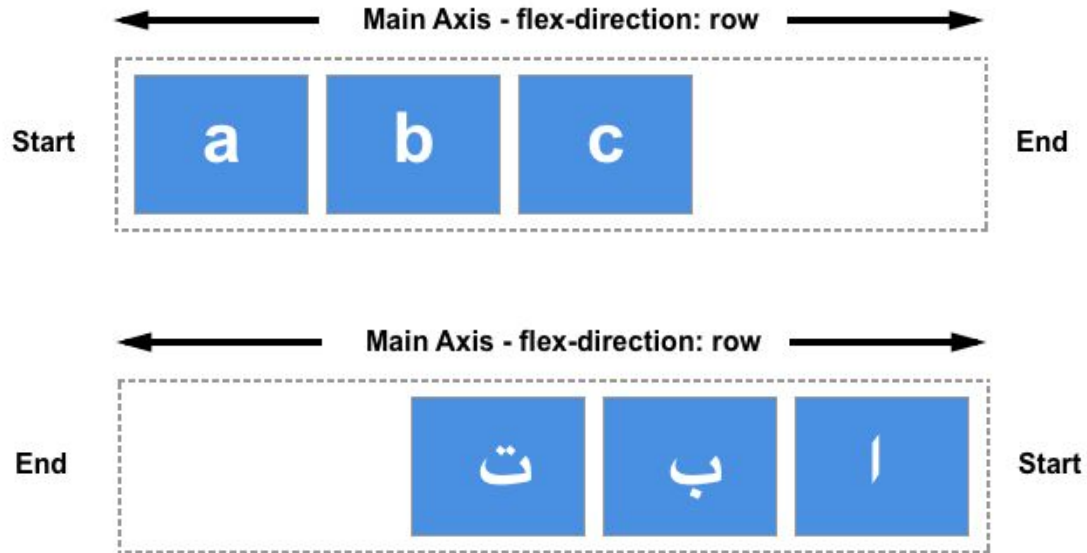
row: de izquierda a derecha.

row-reverse: de derecha a izquierda.

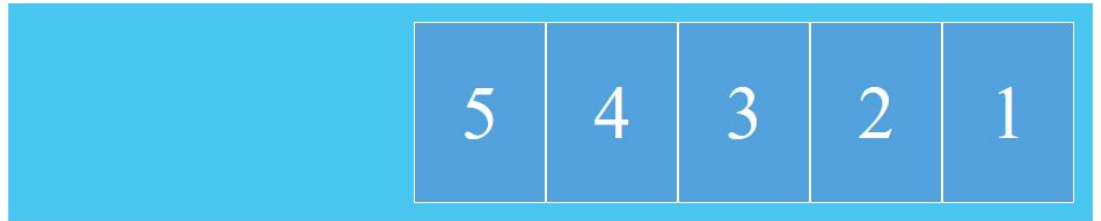
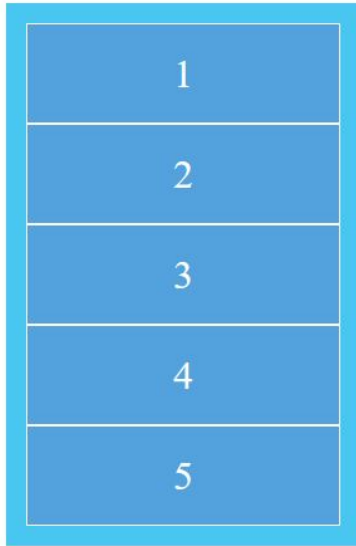
column: de arriba a abajo.

column-reverse: de arriba a abajo.

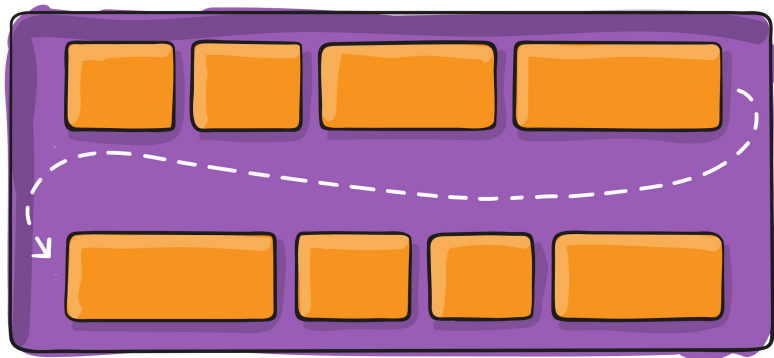
# flex-direction



# Hands on!



## Contenedores multilínea: flex-wrap



Por defecto flexbox alinea todos los elementos en **una** única fila. Podemos cambiar este comportamiento con **flex-wrap**.

# flex-wrap

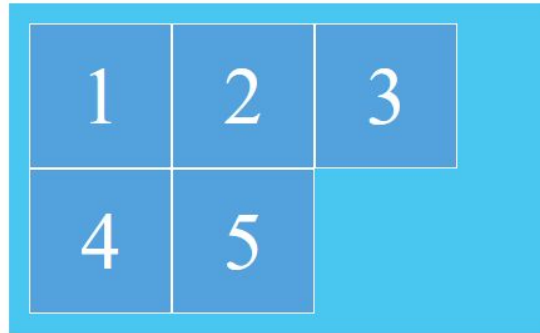
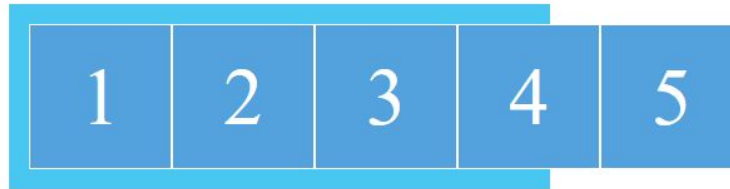
## Valores

**no-wrap:** todos los items en una única línea. Con esta propiedad los ítems podrían salirse del margen si estos no pudieran contraerse, o no contraerse lo suficiente para ser calzados.

**wrap:** cuando el contenedor flex contiene ítems que se les ha asignado un ancho, donde el ancho total de los ítems excede al del contenedor flex, los items se repartirán en las siguientes líneas.

**wrap-reverse:** se comporta igual que wrap pero los items se colocarán de arriba a abajo.

# Hands on!





# flex-flow

Es una simple abreviación que permite indicar las propiedades `flex-wrap` y `flex-direction` en una sola propiedad. El valor por defecto es “row nowrap”

**CSS**

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

# Justificación de los elementos

flex-start



flex-end



center



La propiedad `justify-content` es usada para alinear los ítems en el eje principal.

## Valores

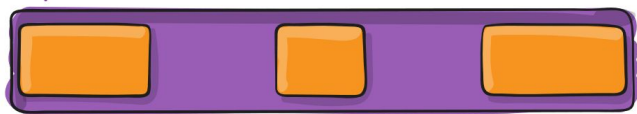
`flex-start`: alinea los ítems al inicio del margen del contenedor.

`flex-end`: alinea los ítems al final

`center`: para alinearlos al centro.

# Justificación de los elementos

space-between



space-around



space-evenly



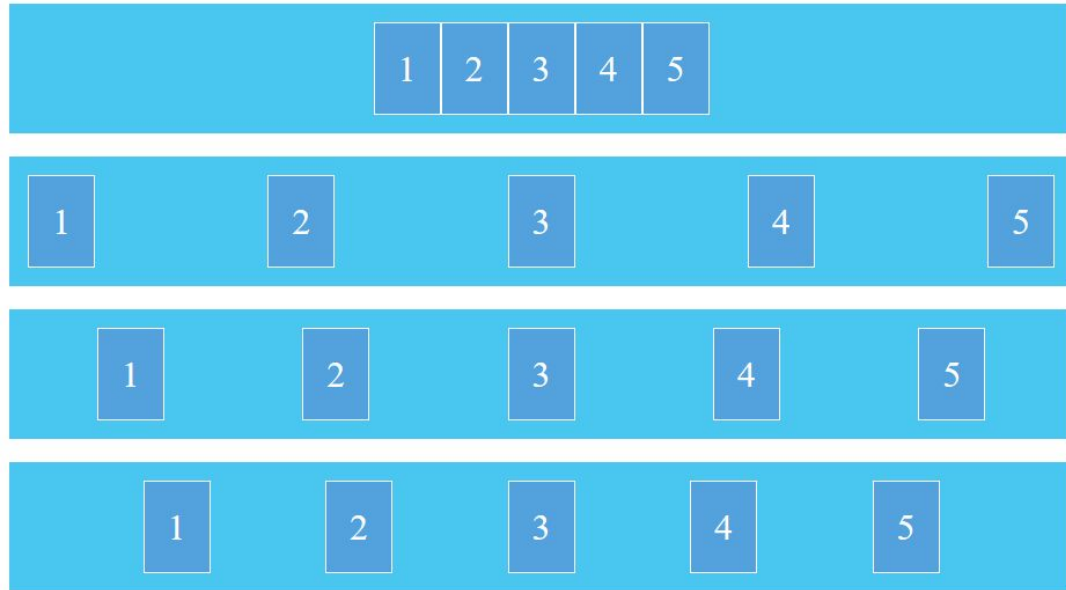
## Valores

space-between: distribuye los ítems para que haya un espacio equitativo entre cada ítem.

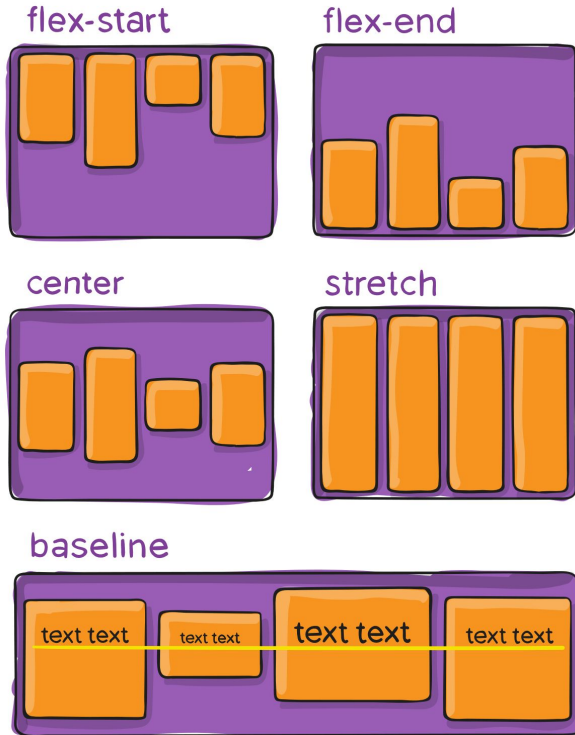
space-around: crea un espacio equitativo a la derecha e izquierda de cada ítem.

space-evenly: el espacio entre elementos y los márgenes del contenedor es equitativo.

# Hands on!



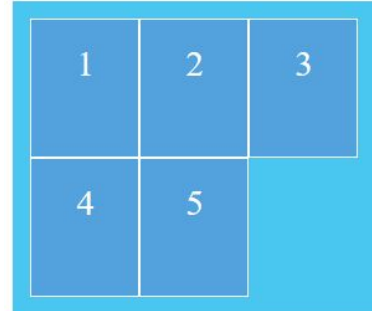
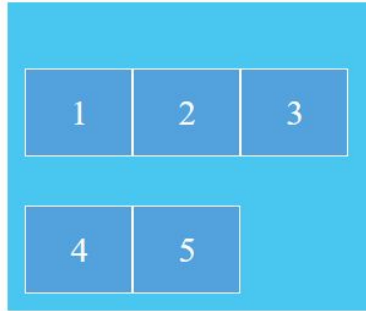
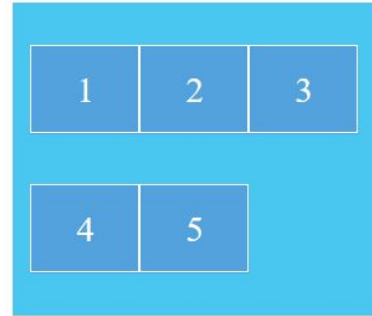
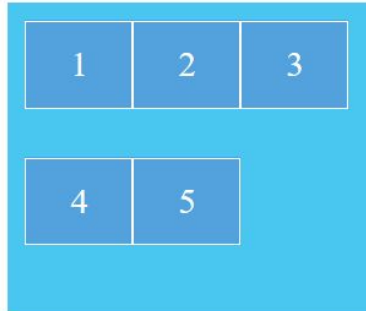
# Alineación de los elementos



La propiedad `align-items` es usada para alinear los ítems en el eje cruzado.

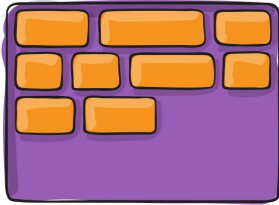
Normalmente se usa cuando hay una única línea (cuándo hay más de una, se debe utilizar `align-content`).

# Hands on!

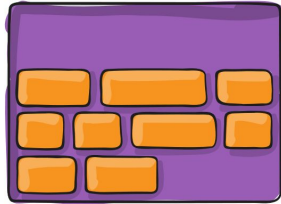


# Alineación de los elementos

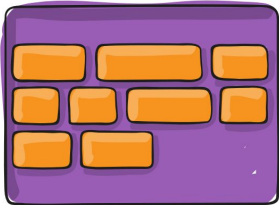
flex-start



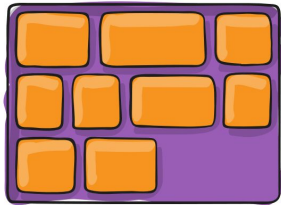
flex-end



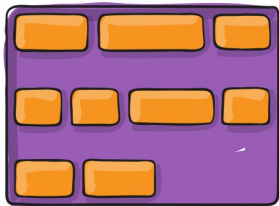
center



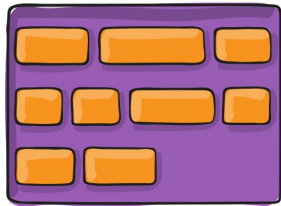
stretch



space-between



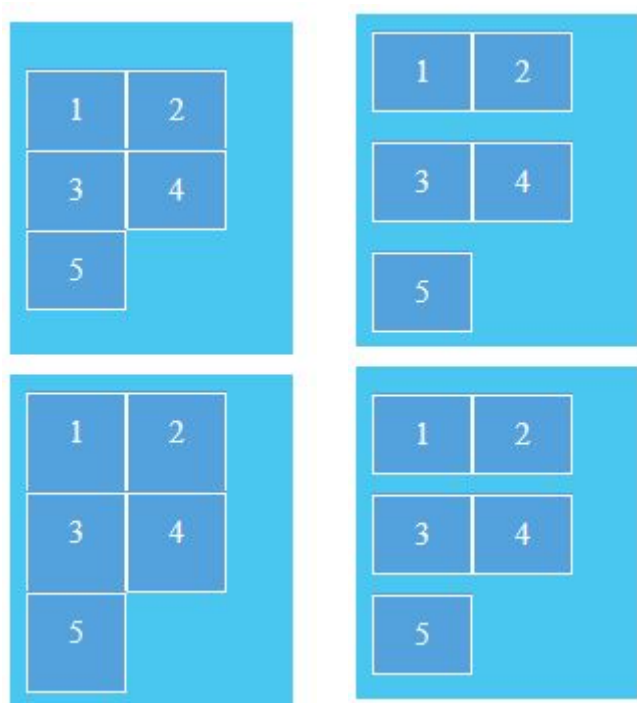
space-around



La propiedad **align-content** ajusta las líneas dentro de un contenedor flex cuando hay espacio extra en el eje transversal.

Esta propiedad no tiene efecto en cajas flexibles de una sola línea.

# Hands on!

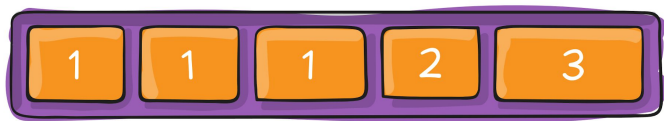




# Propiedades de los items

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

## order



Por defecto los elementos se colocan en el orden en el que se definen.

La propiedad `order` define el orden en el que los elementos se colocan en el contenedor.

# order

CSS

```
.item {  
  order: <integer>; /* default is 0 */  
}
```

Los elementos se colocarán en **orden ascendente** según el valor de order.

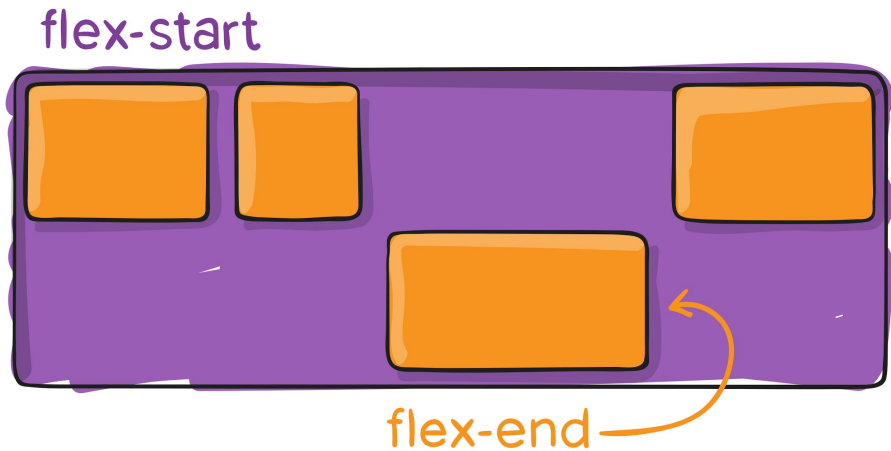
Los items con el mismo valor de order se colocarán en el orden en el cual aparecen en el código fuente.

# Hands on!

-2	-1	0	3	6
----	----	---	---	---

-2
-1
0
3
6

# align-self

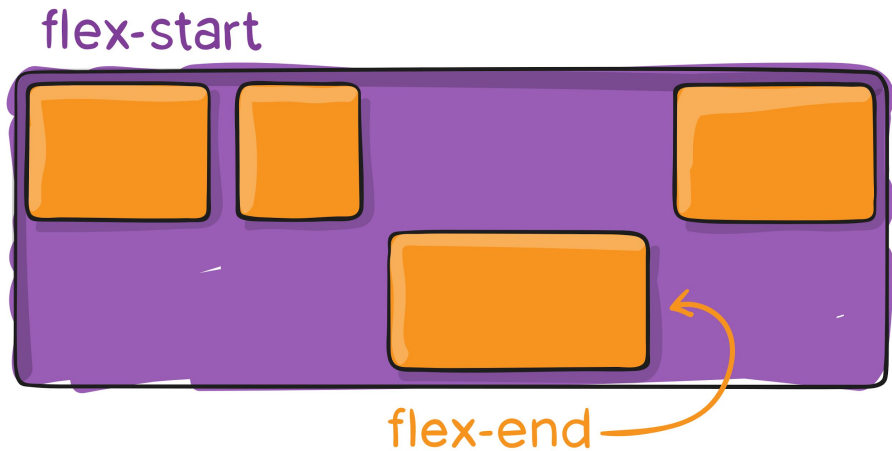


La propiedad `align-self` permite alinear un elemento de la forma deseada y sobre-escribir lo indicado mediante la propiedad `align-items`.

Posibles valores:

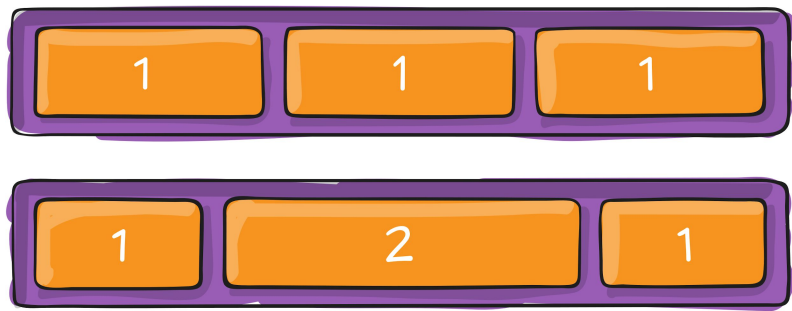
`auto` | `flex-start` | `flex-end` | `center` |  
`baseline` | `stretch`;

# ¡Importante!



El valor por defecto de la propiedad `align-self` es `stretch`, es decir, los elementos cogen el ancho máximo. Por

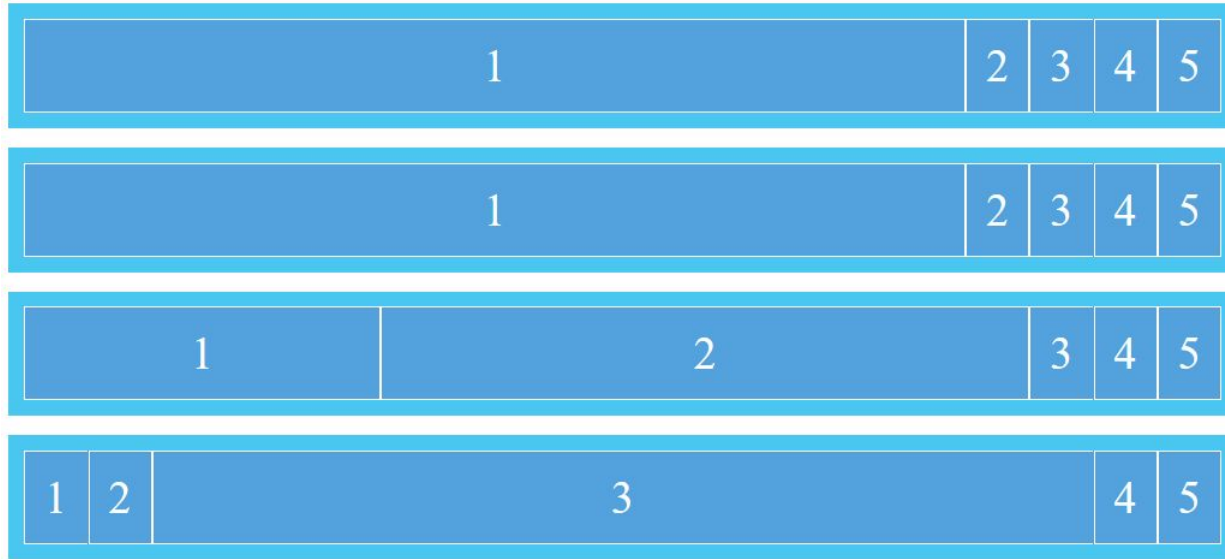
# flex-grow



Por defecto los elementos se colocan en el orden en el que se definen.

La propiedad **flex-grow** especifica qué cantidad de espacio debe ocupar el elemento dentro del contenedor flexible.

# Hands on!





# flex-basis

La propiedad de CSS `flex-basis` especifica cual es el tamaño inicial de un elemento, antes de comenzar a distribuir el espacio mediante otras propiedades.

Si su valor es `auto`, fijará su ancho en función de la propiedad `width`.

## CSS

```
flex-basis: <width>
```

```
.flex-item {  
  flex-basis: 100px;  
}
```

# Hands on!

120px	220px	3	4	5	
-------	-------	---	---	---	--

# Columnas con misma anchura

Un layout típico es el de dos (o más) columnas de la misma anchura.

Esto podemos conseguirlo fácilmente combinando las propiedades `flex-grow` y `flex:basis`

```
.columna {  
  flex-grow: 1;  
  flex-basis: 0;  
}
```

# flex-shrink

La propiedad `flex-shrink` indica cuánto encoger un item cuando sea necesario (cuando haya overflow).

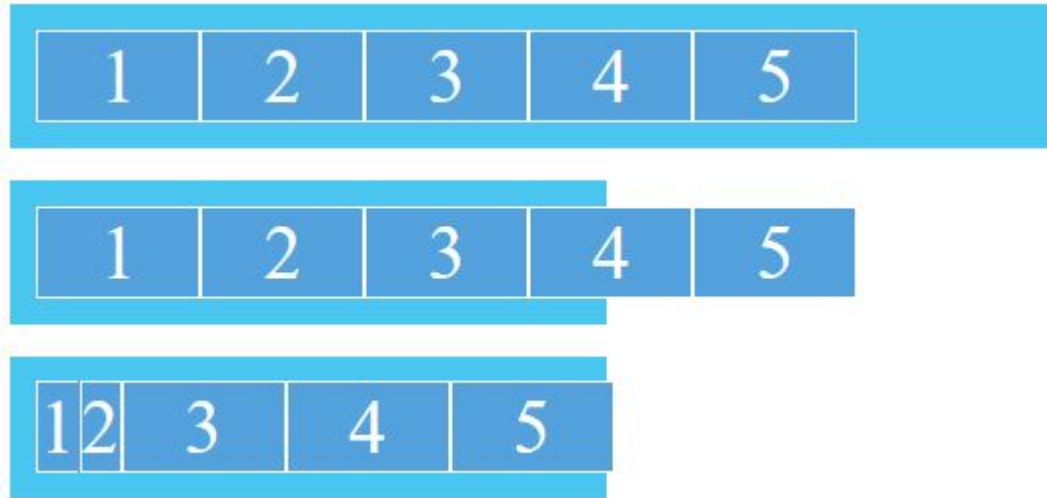
Solo tiene sentido si se ha indicado la propiedad `flex-basis`.

CSS

```
.item {  
    flex-shrink: <number>; /* default 1 */  
}
```

\* El valor 0 indica que no se contraerá, por lo que puede ocurrir overflow.

# Hands on!



# Play, Fun, Learn!

<https://flexboxfroggy.com/#es>

## FLEXBOX FROGGY

Level 7 of 24

The frogs need to cross the pond again, this time for some lilypads with plenty of space around them. Using a combination of `justify-content` and `align-items`.

```
1 #pond {
2   display: flex;
3
4
5 }
6
7
8
9
10
```

Next

Flexbox Froggy is created by Thomas Park • GitHub • Twitter

English • Español • Français • Deutsch • Português • Italiano • Svenska • Polski • Česky • Magyar • Lietuvių • Русский • Türkçe • العربية • 简体中文 • 繁體中文 • 日本語 • 한국어 • Tiếng Việt • Esperanto



<http://www.flexboxdefense.com/>

## Flexbox Defense

Wave 1 of 12


Use the `justify-content` property to move these two towers into position. Click the `?` button in the stylesheet for a reminder on how the property works.

```
1 .tower-group-1 {
2   display: flex;
3   justify-content:center;
4 }
```

☒ hide tower inputs Cancel Wave

Made by Charming Allen

Points: 0



# Ejercicio 1: Menú horizontal



**THRONE**

INICIO

ABOUT

SERVICIOS

CONTACTO

EN

EUS

- Crea un menú como el de la imagen, compuesto por un logotipo, una lista de elementos de navegación y dos enlaces para cambio de idioma.
- La distancia entre bloques tiene que ser la misma.
- Logotipo: <http://throne.stonedthemes.com/wp-content/uploads/2015/08/logo.png>

## Ejercicio 2: Menú vertical

- Crea un menú como el de la imagen, compuesto lista de elementos, los cuales contienen un texto y un enlace.

INICIO	ENTRAR
ABOUT	ENTRAR
SERVICIOS	ENTRAR
CONTACTO	ENTRAR



## Ejercicio 3: Centrar en página

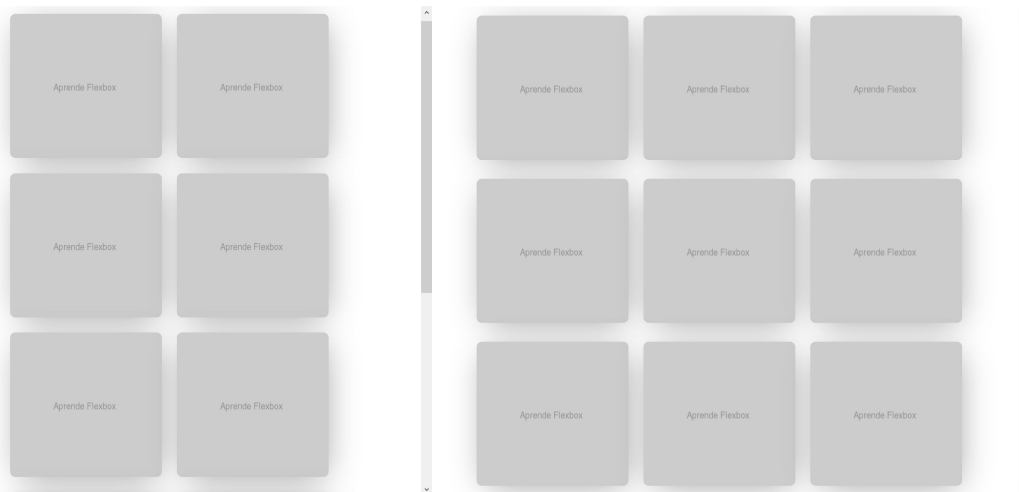
- Crea una página que contenga un div centrado vertical y horizontalmente en la página.

### I'm Centered!

This box is both vertically and horizontally centered. Even if the text in this box changes to make it wider or taller, the box will still be centered. Go ahead, give it a try. Just click to edit the text.

## Ejercicio 4: Galería de imágenes adaptable

- Crea una galería de imágenes que se adapte en función del espacio disponible.



``

# Ejercicio 5: Layout de 3 columnas

```
<body>
  <header>Header</header>
  <main>
    <article>Article</article>
    <nav>Nav</nav>
    <aside>Aside</aside>
  </main>
  <footer>Footer</footer>
</body>
```

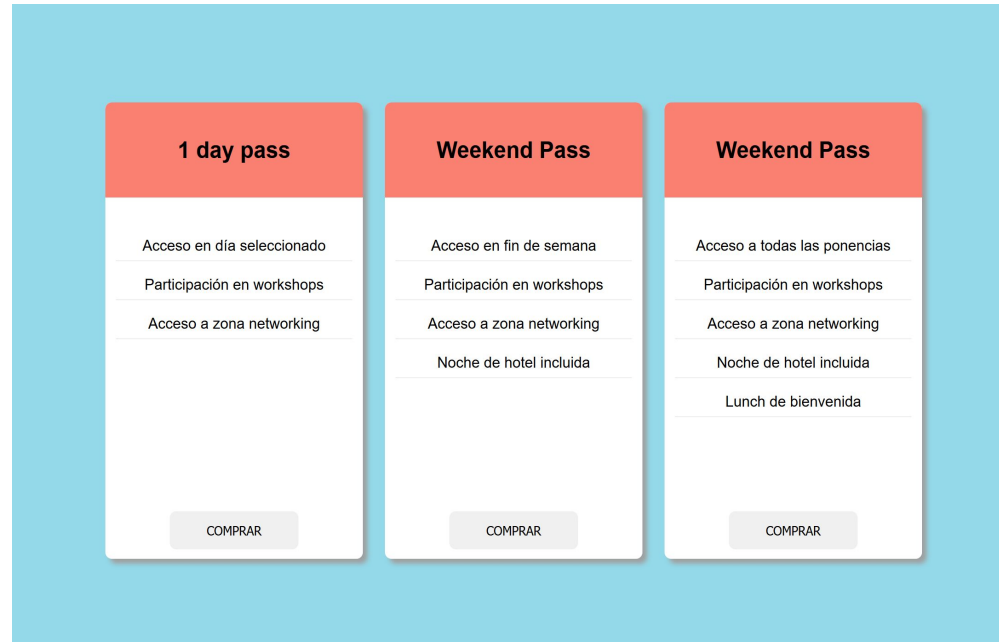


Doble de ancho que el resto de columnas

**NOTA:** Utiliza la unidad “vh” para que la página ocupe toda la pantalla

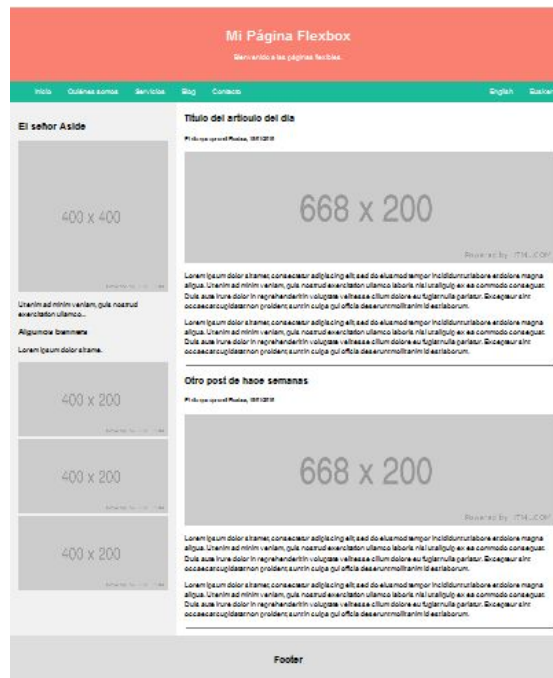
## Ejercicio 6: Pricing box

- Totalmente centrado en la página, cajas de mismo tamaño, con sombra y bordes redondeados.
- Altura mínima de cada caja: 480px
- Colores:
  - Cabecera: salmon
  - Fondo: #93d9e9



# Ejercicio 7: Layout completo

- Crea una página como la de la imagen, compuesta por: header, menú, barra lateral, contenido tipo blog y footer.
- El ancho máximo de la página completa será de 1000px, y la proporción de las columnas es 30%-70%.
- El footer tiene que aparecer siempre al borde inferior de la página.



# Ejercicio 7: Layout completo



## Sources

- [CSS Tricks:](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)  
<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- [Mozilla MDN Web Docs:](https://developer.mozilla.org/) <https://developer.mozilla.org/>
- [Codepen:](https://codepen.io/enxaneta/full/adLPww/) <https://codepen.io/enxaneta/full/adLPww/>
- [CSSReference.io:](https://cssreference.io/flexbox/) <https://cssreference.io/flexbox/>