

FLEXBOX

¿QUÉ ES FLEXBOX?

Flexbox es un modo de diseño que nos permite **crear estructuras para sitios web de una forma más fácil**. Si ya sabes de HTML y CSS probablemente alguna vez habrás visto que los sitios web se realizan utilizando la propiedad float, para desplazar contenedores.

Con Flexbox ya no necesitarás usar float para posicionar tus elementos, al contrario, con Flexbox podrás posicionar y distribuir los elementos como tú quieras.

¿QUÉ ES FLEXBOX?

Entonces...

No se trata de una propiedad de CSS, sino un conjunto de ellas, se basa sobre un contenedor (**padre**) para ordenar a sus items (**hijos**).

No solo puedes posicionar elementos vertical y horizontalmente, sino que **puede establecer cómo se distribuirán, el orden que tendrán e incluso el tamaño que tendrán en proporción a otros elementos**. **Esto es perfecto para crear diseños adaptables a dispositivos móviles (Responsive Design).**

BENEFICIOS


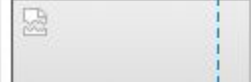
- Distribuir los elementos en sentido vertical u horizontal.
- Facilita la adaptación del contenido en distintos dispositivos
- Redefinir el sentido del flujo de los elementos (hacia arriba, hacia abajo, hacia la izquierda o hacia la derecha).
- Alinear los elementos respecto al padre o respecto a sus hermanos.

ALGO DE HISTORIA

Veamos un poco de historia, brevemente, de cómo ha evolucionado la forma de crear estructuras.

AAANTES

USANDO TABLAS

| | | | |
|--|--|---|---|
| td 130 x 53 |  | Upload, tag and share your videos worldwide! | Sign Up Log In Help |
| | <input type="text"/> | Search Videos | Upload Videos // Browse Videos |
| Home My Videos My Favorites My Profile | <u>Upload</u> | <u>Watch</u> | <u>Share</u> |
| | Quickly upload and tag videos in almost any video format. | Instantly find and watch 1000's of fast streaming videos. | Easily share your videos with family, friends, or co-workers. |
| Today's Featured Videos | See More Videos | <u>Sign up for your free account!</u> | |
|  | <u>Tiffany</u> | Is a total dork | Recent Tags: |
| Tags: // tiffany : dork : funny : random : | guitar : francis : video : xts912 : belgium : man : driving : fall : woman : isls : flash : iran : orange : scary : soccer : homestead : live : crazy : fight : music : food : | | |

Fuente: <https://web.archive.org/web/20050810030357/http://www.youtube.com/>

NO HASTA HACE MUCHO...

USANDO FLOAT

The image shows a screenshot of a YouTube homepage from 2010. The main content area features a grid of video thumbnails. The first row includes a video titled "Interview with Obama" and three smaller video thumbnails with titles like "Why are banks reluctant to modif...", "State of the Union Question", and "President Obama - How will you h...". The right sidebar contains sections for "What's New", "New Product Id", "YouTube Badge", and "Your Questions". A CSS inspector is open on the right, showing the style rules for the element `#homepage-main-con`. The `float: left;` rule is highlighted with a red box.

Interview with Obama

President Obama took your questions in an exclusive interview live from the White House. Watch the full video here as well as three of the questions submitted by YouTube users.

Why are banks reluctant to modif...

5 days ago
443 views
★★★★★
0334231

State of the Union Question

5 days ago
3,337 views
★★★★★
energyaction

President Obama - How will you h...

5 days ago
1,494 views
★★★★★
ENOUGHproject

What's New

New Product Id

YouTube Badge

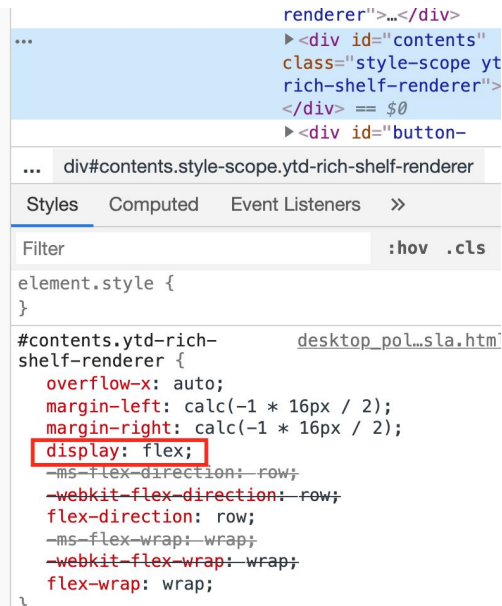
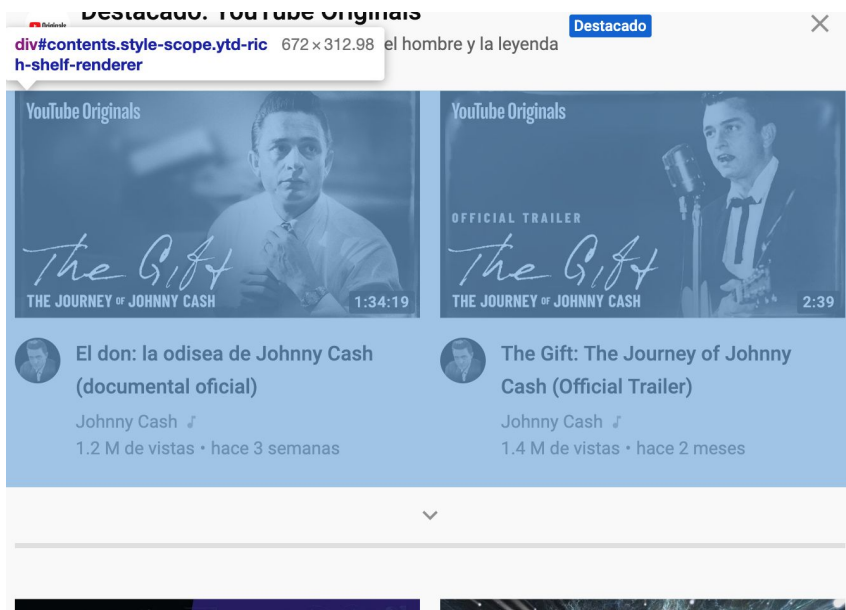
Your Questions

```
aseDiv.date-2010C
Estilos de filtro
elemento {
}
#baseDiv .homepage
skin {
width: 625px;
}
#homepage-main-con
{
float: left;
clear: left;
width: 640px;
margin-top: 5px
}
html, body, div, s
img, applet, embed
canvas, object, if
```

Fuente: <https://web.archive.org/web/20100202095628/http://www.youtube.com/>

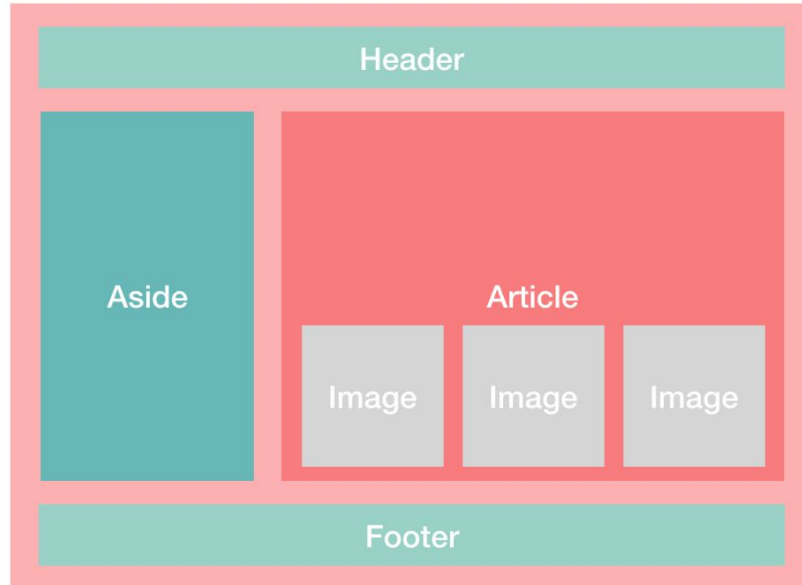
PRESENTE

USANDO FLEXBOX



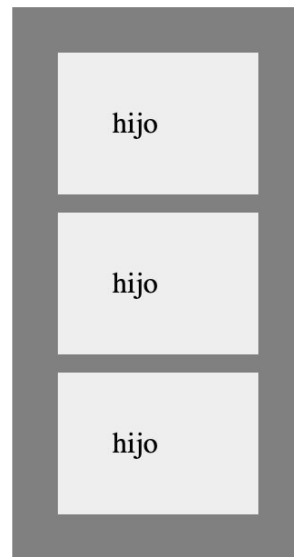
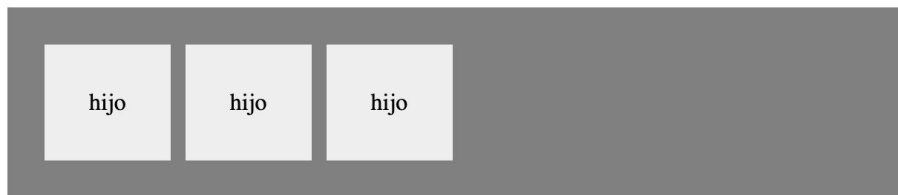
OBJETIVO

Vamos a crear una estructura así

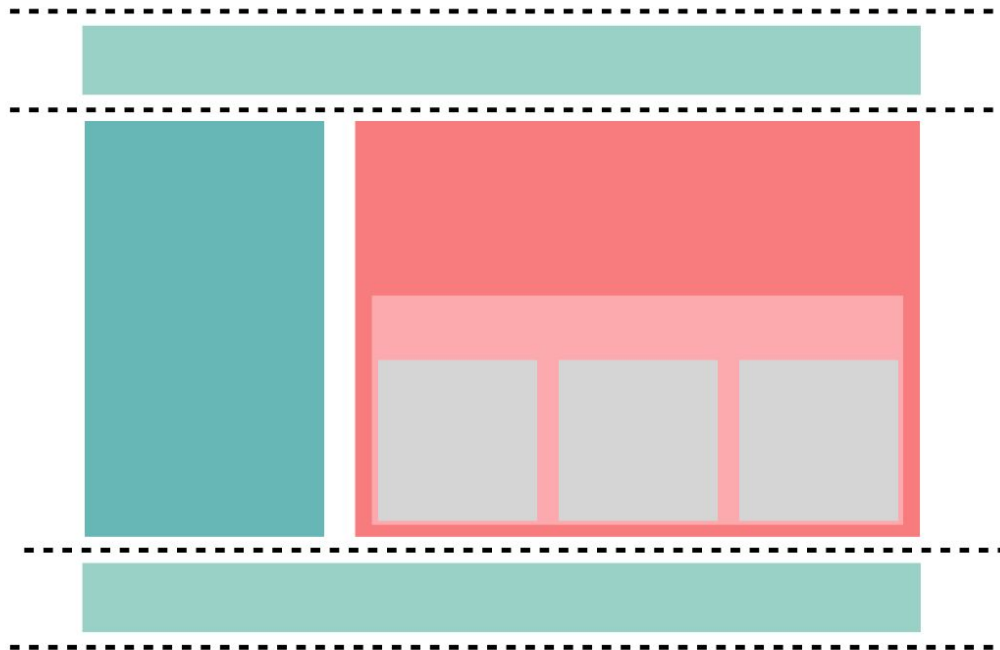


ENTONCES...

Para alcanzar este objetivo, vamos a necesitar **seccionarlo en filas o columnas**, en este caso filas.



ENTONCES...



PROPIEDADES DE PADRES E HIJOS

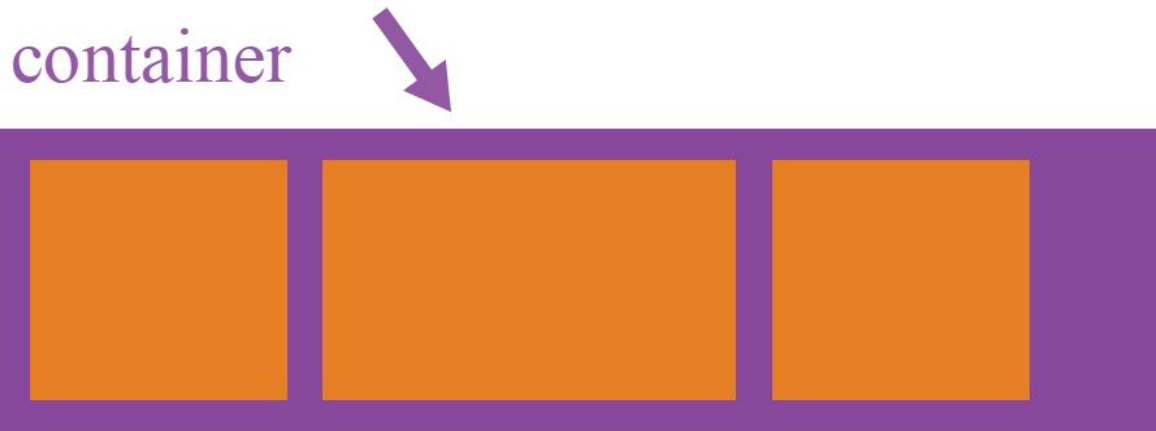


FLEXBOX

Propiedades para aplicar en el contenedor flexible (el padre)

- **display:** *flex* (el que inicia): Indicará que sus hijos serán “flexibles”
- **flex-direction:** elegir dirección vertical u horizontal
- **flex-wrap:** ¿se hará multilínea cuando llegue al límite?
- **flex-flow:** *abreviación de propiedades*
- **justify-content:** alinear horizontalmente a los hijos si el padre es “fila” o verticalmente si el padre es “columna”
- **align-items:** alinea verticalmente a los hijos (si el padre es “columna”)
- **align-content:** alinea verticalmente a los hijos cuando son multilínea

PROPIEDADES DEL PADRE



(Fuente: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>)

¿CÓMO EMPEZAMOS?

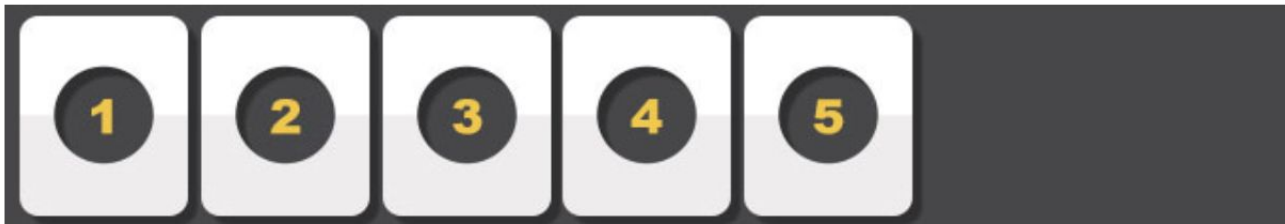
Lo primero que debemos hacer es establecer la propiedad display con el valor flex en el elemento padre.

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

FLEX-DIRECTION: ROW

Esta propiedad nos va a permitir especificar si queremos que los flex items se dispongan en filas o columnas.

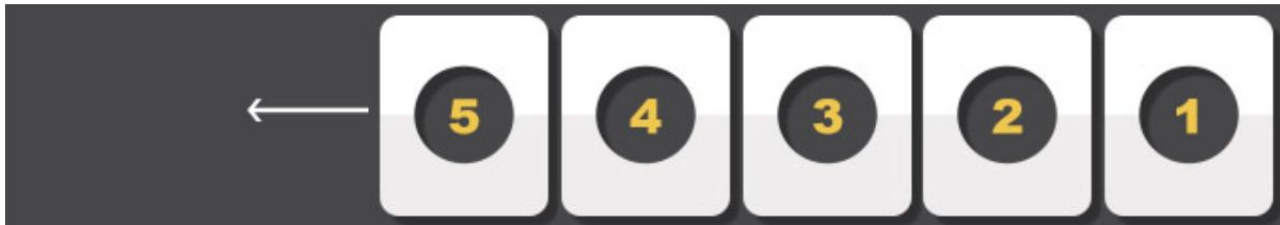
```
.padre-flex {  
  display: flex;  
  flex-direction: row; /* predeterminado */  
}
```



FLEX-DIRECTION: ROW-REVERSE

Con el valor **row-reverse** (fila inversa) los flex items se apilan en una fila de derecha a izquierda.

```
.padre-flex {  
  display: flex;  
  flex-direction: row-reverse;  
}
```



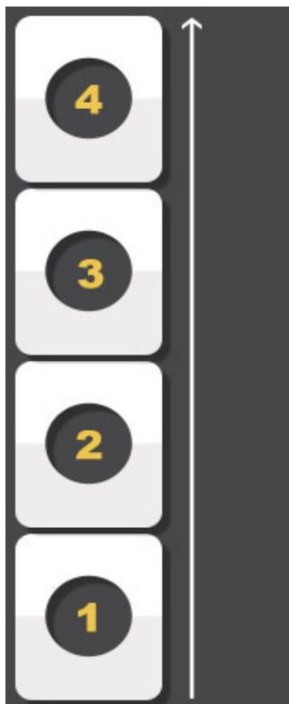
FLEX-DIRECTION: COLUMN



Con el valor **column** los flex items se apilan en una columna **de arriba hacia abajo**.

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
}
```

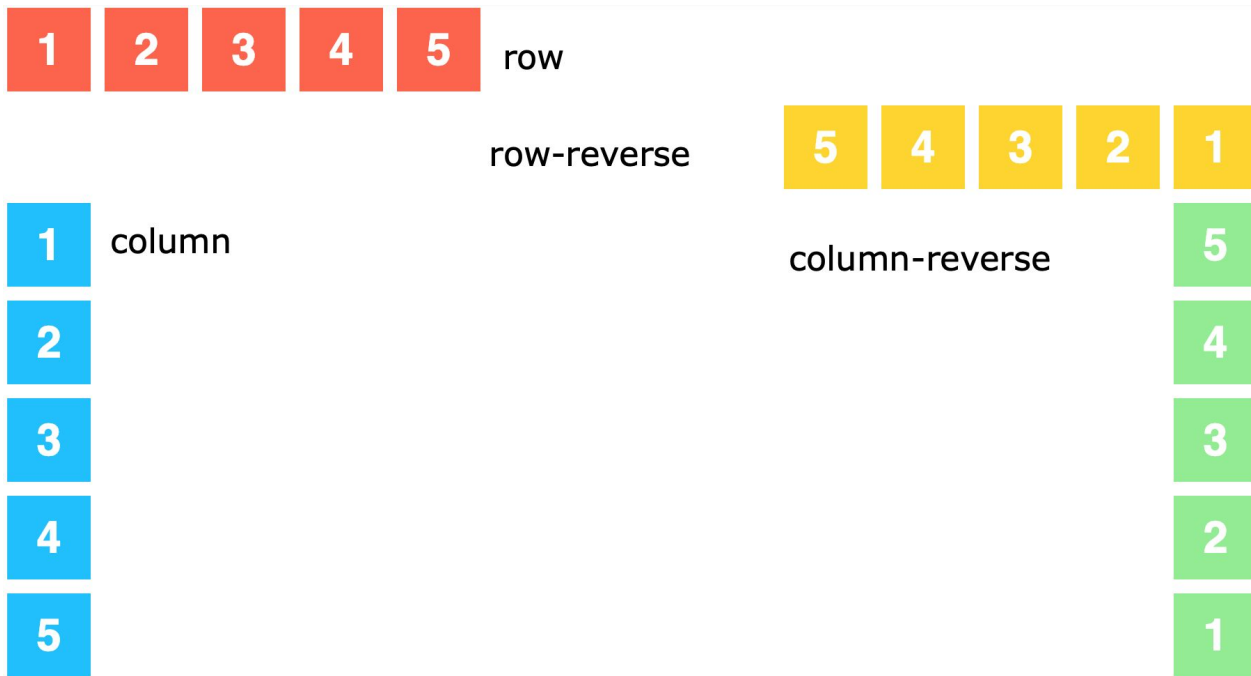
FLEX-DIRECTION: COLUMN-REVERSE



Con el valor **column-reverse** los flex items se apilan en una columna de abajo hacia arriba

```
.padre-flex {  
  display: flex;  
  flex-direction: column-reverse;  
}
```

FLEX-DIRECTION - RESUMEN



FLEX-WRAP

El comportamiento inicial del contenedor flexible es poder mantener **los flex items en su eje sin importar que las dimensiones de estos items cambien.**

Con *flex-wrap* vamos a poder especificar si queremos que los items puedan saltar a una nueva línea si el contenedor flexible se queda sin espacio.

FLEX-WRAP: NOWRAP

```
.padre-flex {  
  display: flex;  
  flex-wrap: nowrap;  
}
```



FLEX-WRAP: WRAP

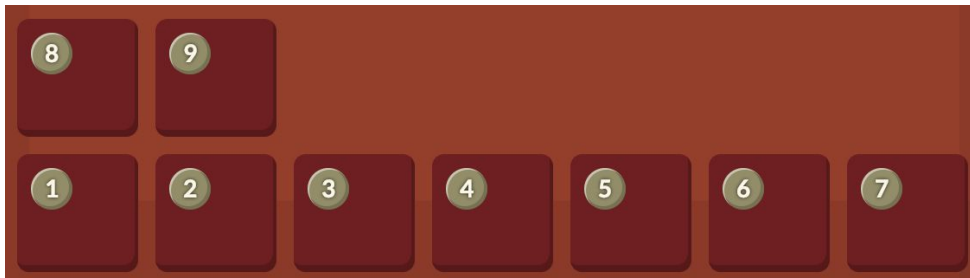
Los *flex items* (*hijos*) pueden romper la línea del eje horizontal si les es necesario **para así conservar las características de sus dimensiones**. Esto es de izquierda a derecha y de arriba a abajo.



```
.padre-flex {  
  display: flex;  
  flex-wrap: wrap;  
}
```

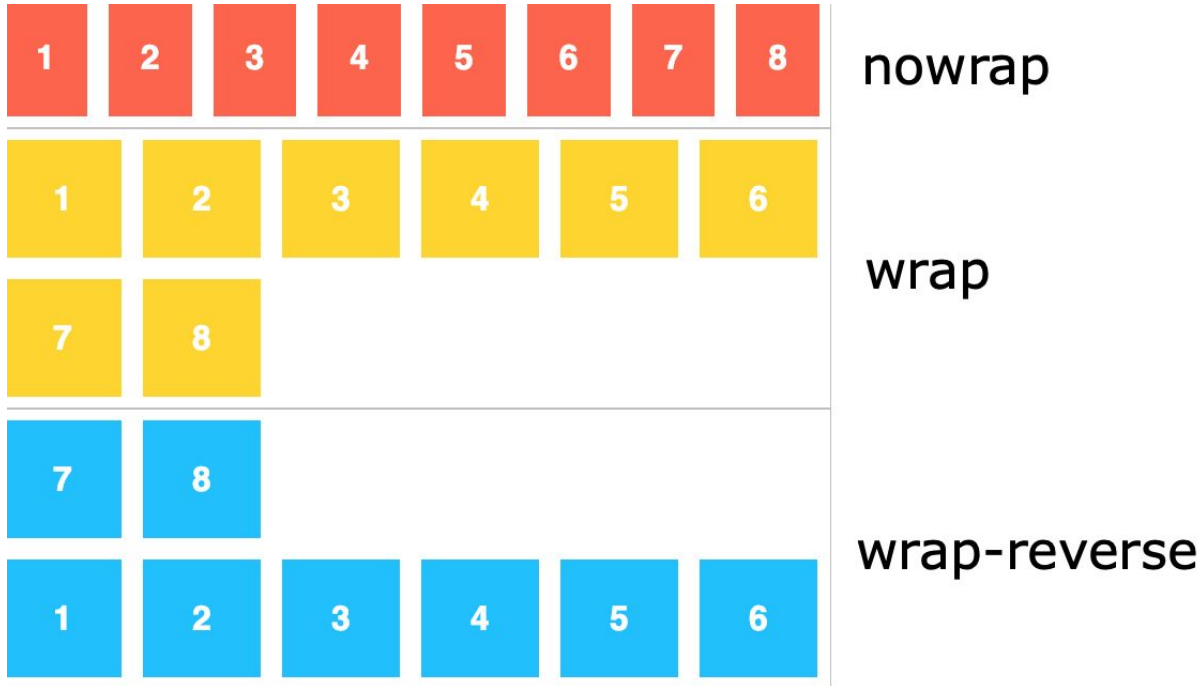
FLEX-WRAP: WRAP-REVERSE

Esta vez el orden es de izquierda a derecha y de abajo a arriba.



```
.padre-flex {  
  display: flex;  
  flex-wrap: wrap-reverse;  
}
```


FLEX-WRAP - RESUMEN



FLEX-FLOW

Es la **forma abreviada** (shorthand) o forma rápida para las propiedades

- *flex-direction*
- *flex-wrap*

Se pone primero la propiedad de *flex-direction* y luego la de *flex-wrap*

```
.padre-flex {  
  display: flex;  
  flex-flow: row nowrap;  
}
```

JUSTIFY-CONTENT

- *Justify-content* nos va a permitir **alinear los elementos**.
- Esto puede ser de forma vertical o horizontal según lo especifiquemos con *flex-direction*
- Nos va a ayudar a distribuir los *flex items (hijos)* en el *contenedor (padre)* cuando los ítems no utilicen todo el espacio disponible en su eje principal actual.
- **Los siguientes ejemplos son a base de que el contenedor está en “row”**

JUSTIFY-CONTENT

Los siguientes ejemplos son a base de
que el contenedor está en “row”

```
.padre-flex {  
  display: flex;  
  flex-direction: row;  
}
```

JUSTIFY-CONTENT: FLEX-START

Es alinear los *flex items* (hijos) al lado izquierdo.

```
.padre-flex {  
  display: flex;    flex-direction: row;  
  justify-content: flex-start; /* predeterminado */  
}
```



JUSTIFY-CONTENT: FLEX-END

Es alinear los *flex items* (hijos) al lado derecho.

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: flex-end;  
}
```



JUSTIFY-CONTENT: CENTER

Es alinear los *flex items* (hijos) al centro.

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: center;  
}
```



JUSTIFY-CONTENT: SPACE-BETWEEN

Es hacer que los *flex items* (*hijos*) se tomen la misma distancia o espaciado entre ellos dentro del contenedor flexible quedando el primer y último elemento pegados con los bordes del contenedor en el eje principal.

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: space-between;  
}
```



JUSTIFY-CONTENT: SPACE-AROUND

Muestra los *flex items (hijos)* con el mismo espacio de separación entre sí, toman el espaciado entre los bordes del contenedor padre.

Nota: Para Microsoft Edge16+, no funciona con flexbox, si con Grid

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: space-around;  
}
```



JUSTIFY-CONTENT: SPACE-EVENLY

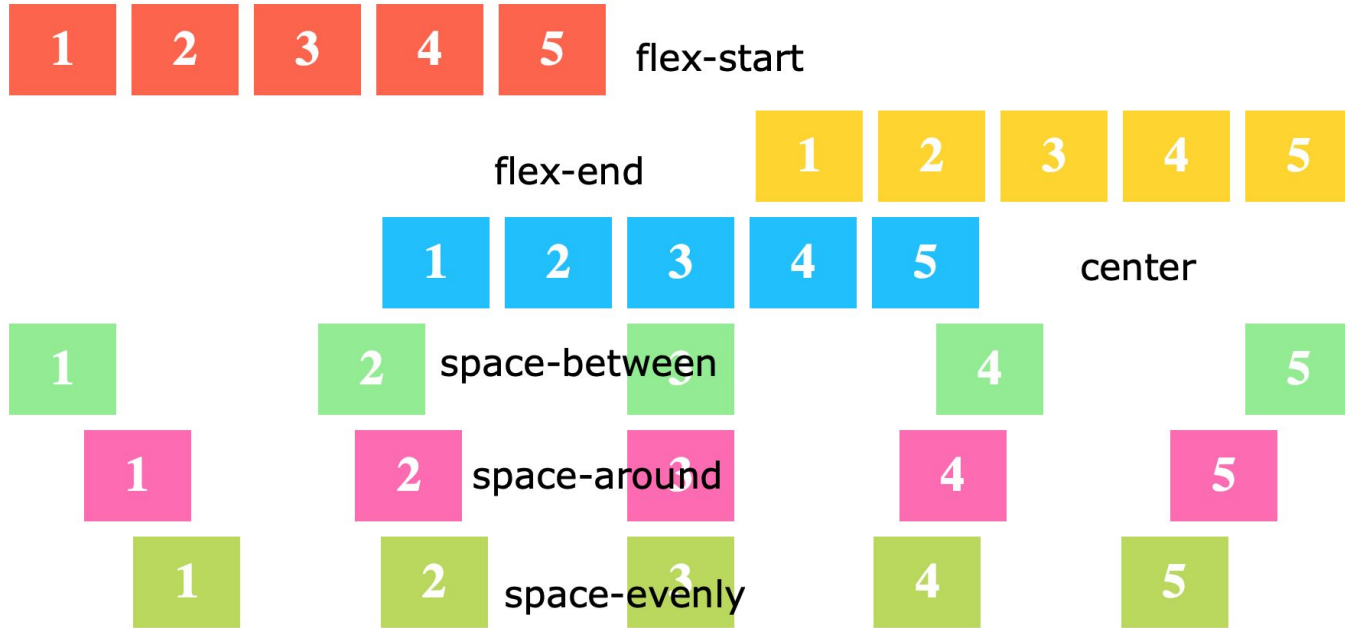
Hace los *flex items (hijos)* que su espacio sea igual, no es lo mismo que space-around.

***Nota:** Para Microsoft Edge16+, no funciona con flexbox, si con Grid*

```
.padre-flex {  
  display: flex;  flex-direction: row;  
  justify-content: space-evenly;  
}
```



JUSTIFY-CONTENT - RESUMEN



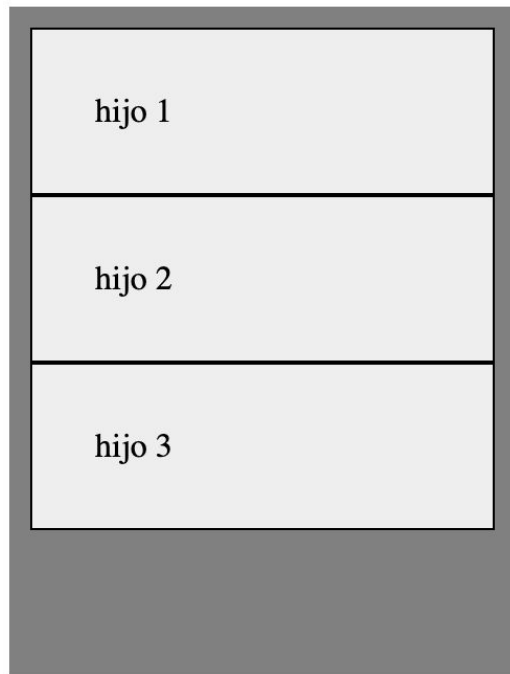
JUSTIFY-CONTENT

Los siguientes ejemplos son a base de que el contenedor está en “column” y tiene alto definido

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
}
```

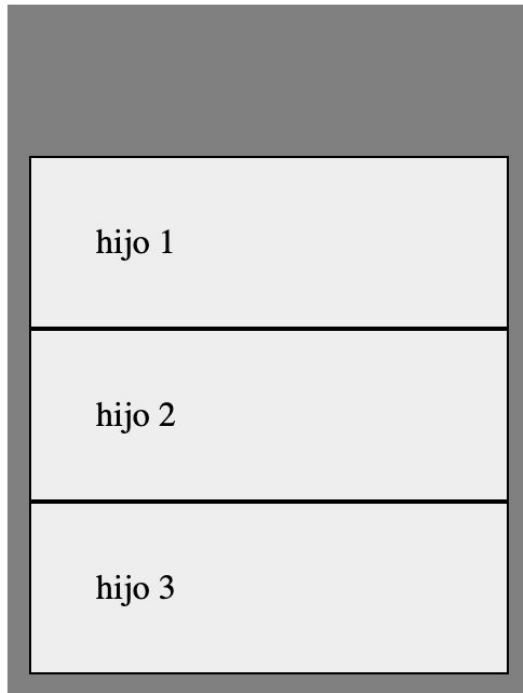
JUSTIFY-CONTENT: FLEX-START

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-start;  
  height: 300px;  
}
```



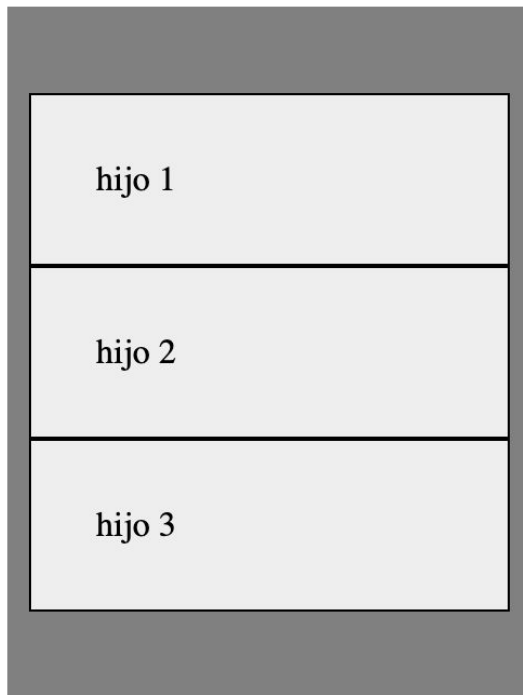
JUSTIFY-CONTENT: FLEX-END

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-end;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: CENTER

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: SPACE-BETWEEN

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-between;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: SPACE-AROUND

```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-around;  
  height: 300px;  
}
```



JUSTIFY-CONTENT: SPACE-EVENLY

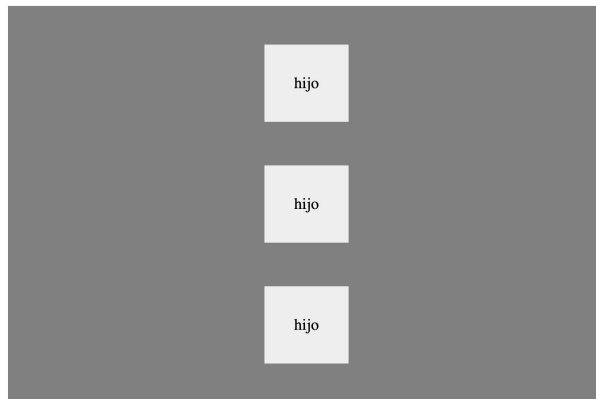
```
.padre-flex {  
  display: flex;  
  flex-direction: column;  
  justify-content: space-evenly;  
  height: 300px;  
}
```



ALIGN-ITEMS

1. Alinear los elementos verticales de forma horizontal.

(flex-direction: column)



2. Alinear los elementos horizontales de forma vertical.

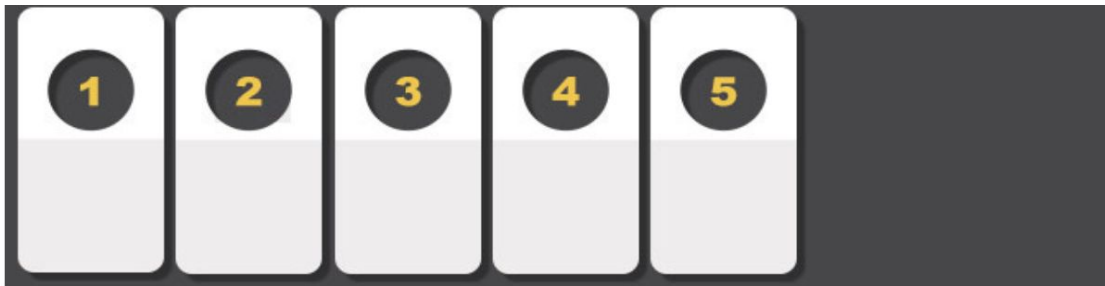
(flex-direction: row)



ALIGN-ITEMS: STRETCH

Va a tratar de **llenar toda la altura** (o anchura) del contenedor, siempre y cuando los *hijos* no tengan propiedades de dimensión definidas.

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: stretch;  
}
```



ALIGN-ITEMS: FLEX-START

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: flex-start; /* predeterminado */  
}
```



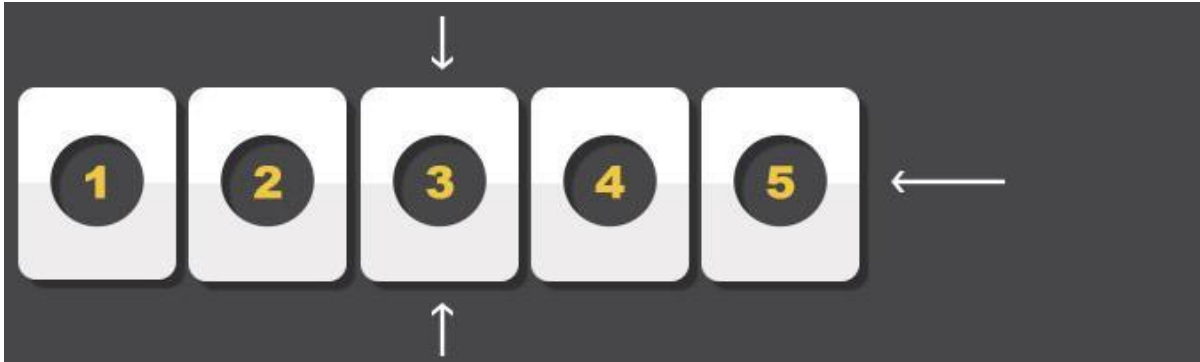
ALIGN-ITEMS: FLEX-END

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: flex-end; /* predeterminado */  
}
```



ALIGN-ITEMS: CENTER

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: center; /* predeterminado */  
}
```

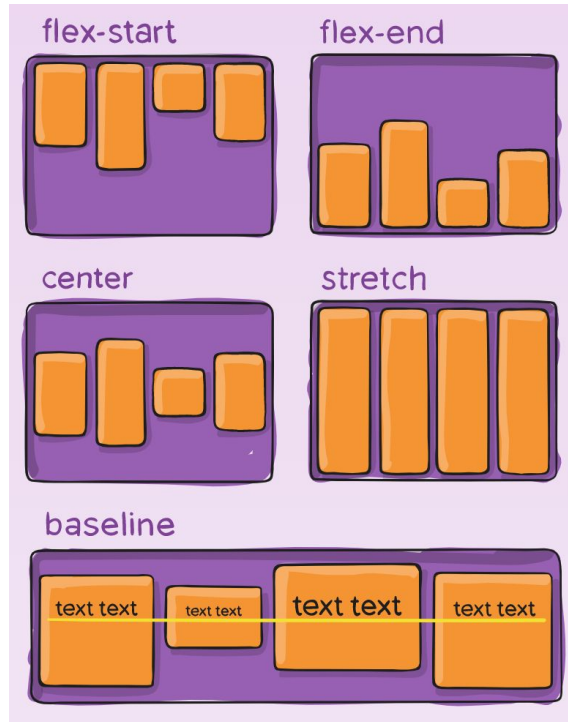


ALIGN-ITEMS: BASELINE

```
.padre-flex {  
  display: flex; flex-direction: row;  
  align-items: baseline; /* predeterminado */  
}
```



ALIGN-ITEMS - RESUMEN



ALIGN-CONTENT

Esta propiedad sólo tiene efecto cuando el contenedor flexible tiene varias líneas de *flex items* (hijos). Si se colocan en una sola línea esta propiedad no tiene ningún efecto sobre el diseño.

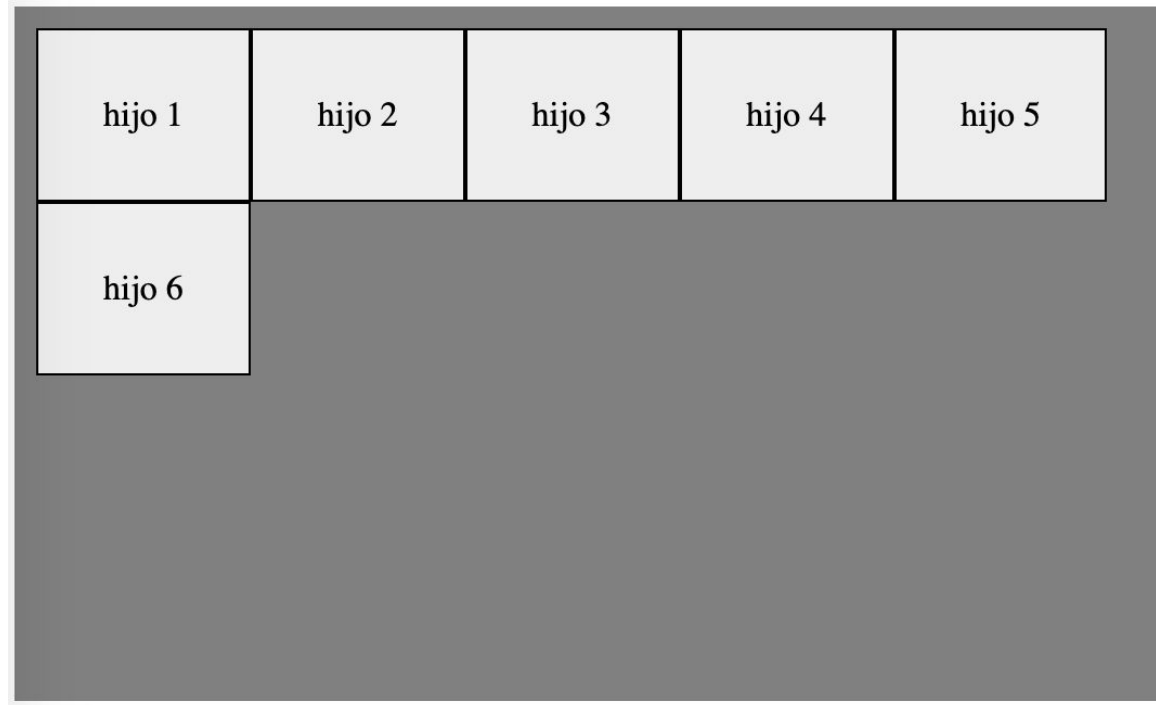
Para poderlo aplicar se necesita tener el atributo *flex-wrap* que me permita verificar los ejes horizontales.

```
.padre-flex {  
  display: flex; flex-wrap: wrap; flex-direction: row;  
  align-content: stretch; /* predeterminada */  
}
```

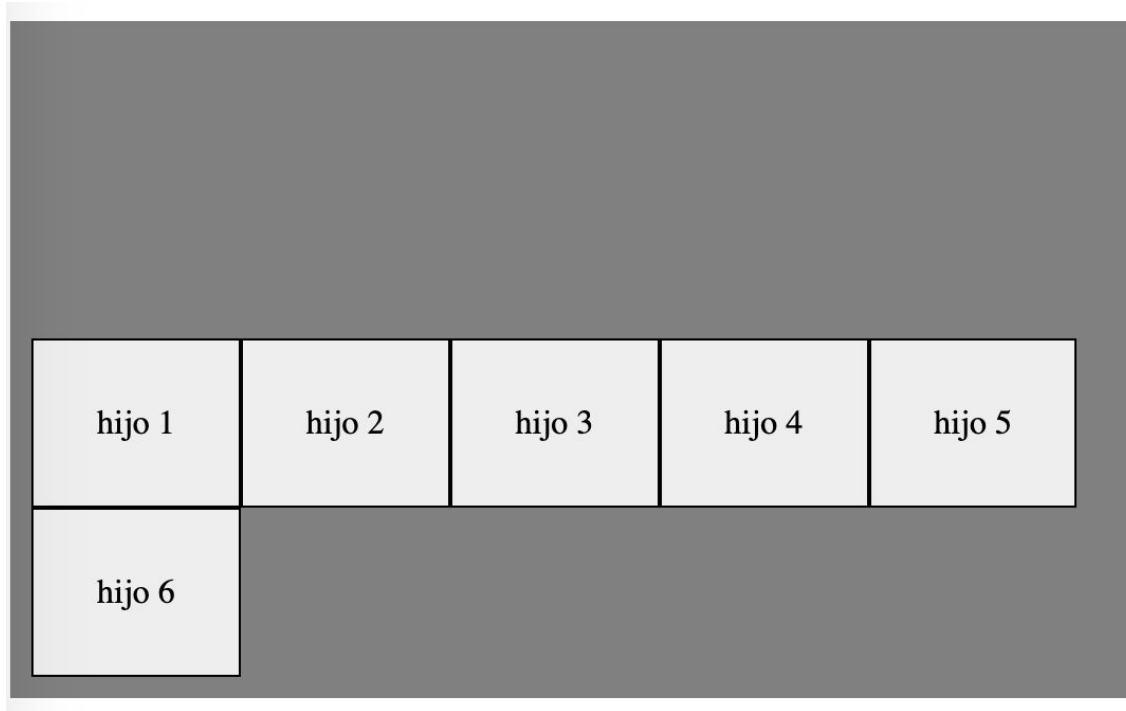
ALIGN-CONTENT: STRETCH

| | | | | |
|--------|--------|--------|--------|--------|
| hijo 1 | hijo 2 | hijo 3 | hijo 4 | hijo 5 |
| hijo 6 | | | | |

ALIGN-CONTENT: FLEX-START



ALIGN-CONTENT: FLEX-END



ALIGN-CONTENT: CENTER

| | | | | |
|--------|--------|--------|--------|--------|
| hijo 1 | hijo 2 | hijo 3 | hijo 4 | hijo 5 |
| hijo 6 | | | | |

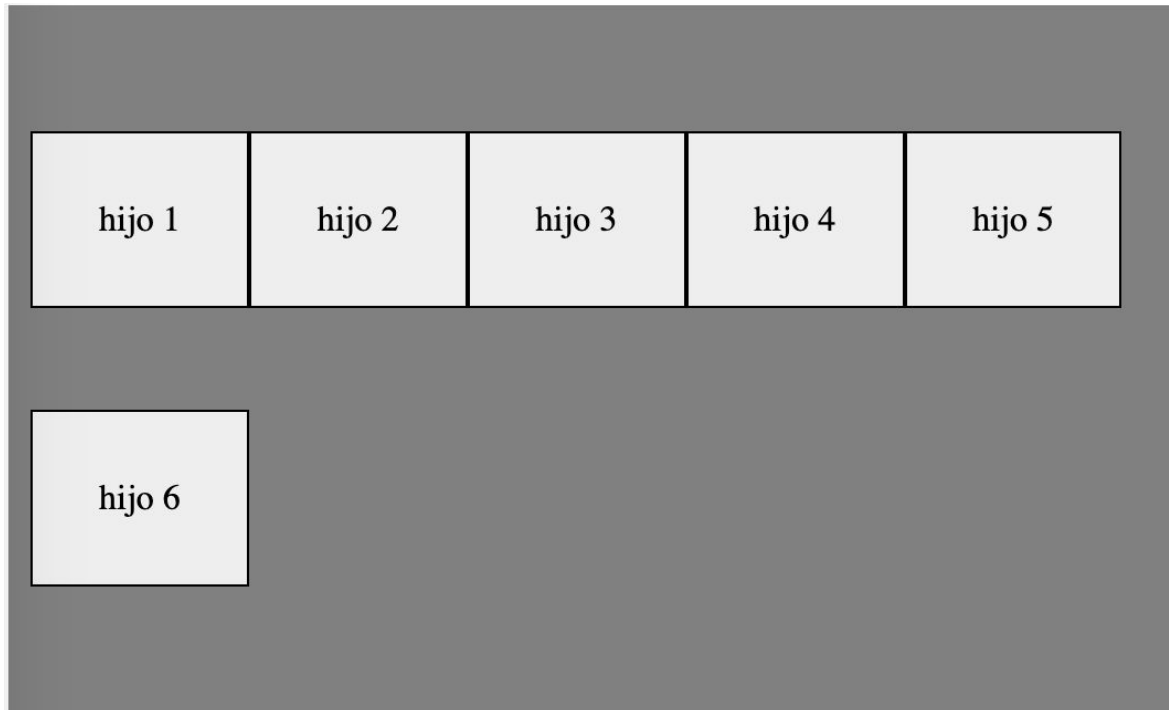
ALIGN-CONTENT: SPACE-BETWEEN



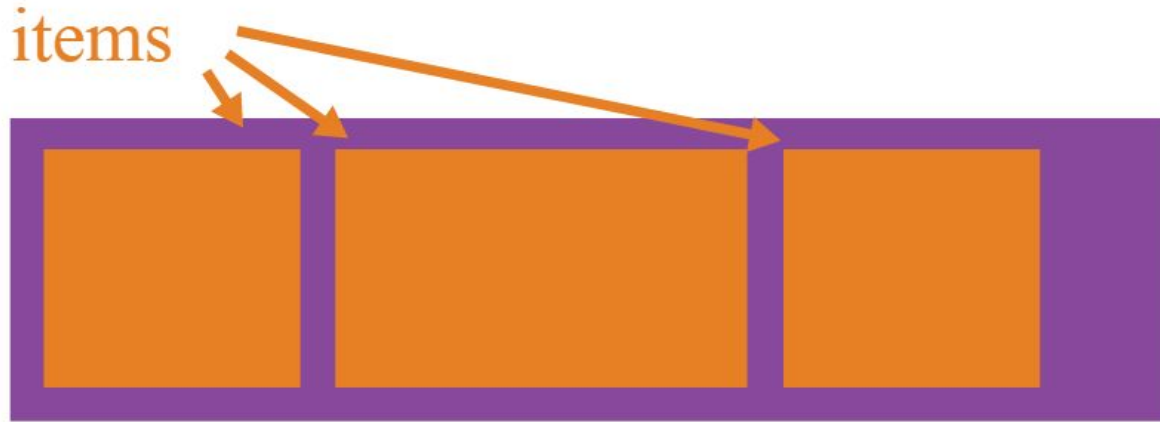
ALIGN-CONTENT: SPACE-AROUND



ALIGN-CONTENT: SPACE-EVENLY



PROPIEDADES PARA LOS FLEX ITEMS (hijos)



ORDER

Esta propiedad permite **modificar el orden** de aparición de un elemento. Recibe como valor numeros enteros.

```
.hijo-flex {  
  order: -1;  
}
```



FLEX-BASIS (parecido al *width* o *height*)

Define el **ancho** de un elemento **inicial**. Este valor por defecto viene configurado en “auto”. Actua como “*máximo*” o “*mínimo*” al usar “flex-grow” o “flex-shrink”

```
.hijo-flex {  
  /* si esta en “row” es como el “width”, si está en  
  “column” es cómo “height” */  
  flex-basis: 100px;  
}
```

FLEX-GROW

- Esta propiedad define la **capacidad de un elemento de crecer cuando en el contenedor todavía hay espacio sobrante.**
- Esta propiedad se configura con un valor numérico entero natural (no acepta negativos).
- Por defecto el valor viene configurado en "0" por lo tanto el elemento no crecerá de manera horizontal.
- Si este valor es configurado en 1 para todos los items, todos ellos crecerán de igual manera por lo que ocuparan la misma cantidad de espacio dentro del contenedor.

FLEX-GROW - HACIENDO CÁLCULOS...

Ok, esto no es fácil

Veamos cómo hacer cálculos para usar esta propiedad.

FLEX-GROW - HACIENDO CÁLCULOS...

Imaginemos que:

- El “padre” tiene un ancho de **400px**
- Tiene **3 hijos**, cada uno **100px**

¿Cuántos tienen la propiedad flex-grow definida?

Imaginemos **que sólo un hijo**, y tiene **como valor 1**.

El cálculo:

Total entre los 3 hijos: **300px**

Espacio disponible: **100px (400-300)**

$100 / 3 = 33\text{px}$ cada fracción (el hijo con valor 1, se lleva $1 \times 33\text{px} = 33\text{px}$)

Hijo con valor 1: **$33 \times 3 = 100\text{px}$**

FLEX-GROW - HACIENDO CÁLCULOS...

```
.hijo-flex {  
  flex-grow: 0;  
}  
  
.hijo-flex.mayor {  
  flex-grow: 1;  
}
```

100px

200px

100px

FLEX-GROW - OTRO EJEMPLO

Imaginemos que:

- El “padre” tiene un ancho de **500p**
- Tiene **2 hijos**, cada uno **100px**

¿Cuántos tienen la propiedad flex-grow definida?

Imaginemos **que:**

1 hijo tiene “flex-grow: 2” y el otro “flex-grow: 1”

El cálculo (vital calcular cuantos “grow” hay):

Total entre los 2 hijos: **200px**

Espacio disponible: **300px (500-200)**

$300 / 3 = 100\text{px}$ cada fracción (el hijo con valor 2, se lleva $2 \times 100\text{px} = 200\text{px}$)

Hijo con valor 2: **$2/3 = 200\text{px}$**

Hijo con valor 1: **$1/3 = 100\text{px}$**

FLEX-GROW - HACIENDO CÁLCULOS...

```
.hijo-flex {  
  flex-grow: 1;  
}  
.hijo-flex.mayor {  
  flex-grow: 2;  
}
```

100px

200px

FLEX-SHRINK

Básicamente es lo mismo que flex-grow, PERO
con el espacio faltante.

FLEX-SHRINK

Básicamente es lo mismo que flex-grow, PERO con el espacio faltante.

Usemos la siguiente herramienta para calcular

<https://www.madebymike.com.au/demos/flexbox-tester/>

Links útiles

- [Demo de flexbox](#)
- [Herramienta para probar propiedades de flexbox](#)
- [A complete guide to Flexbox](#)
- [How Flexbox works – explained with big, colorful, animated gifs](#)
- [Aprendiendo jugando](#)