INCLUYENDO FLEXBOX A NUESTRO PROYECTO

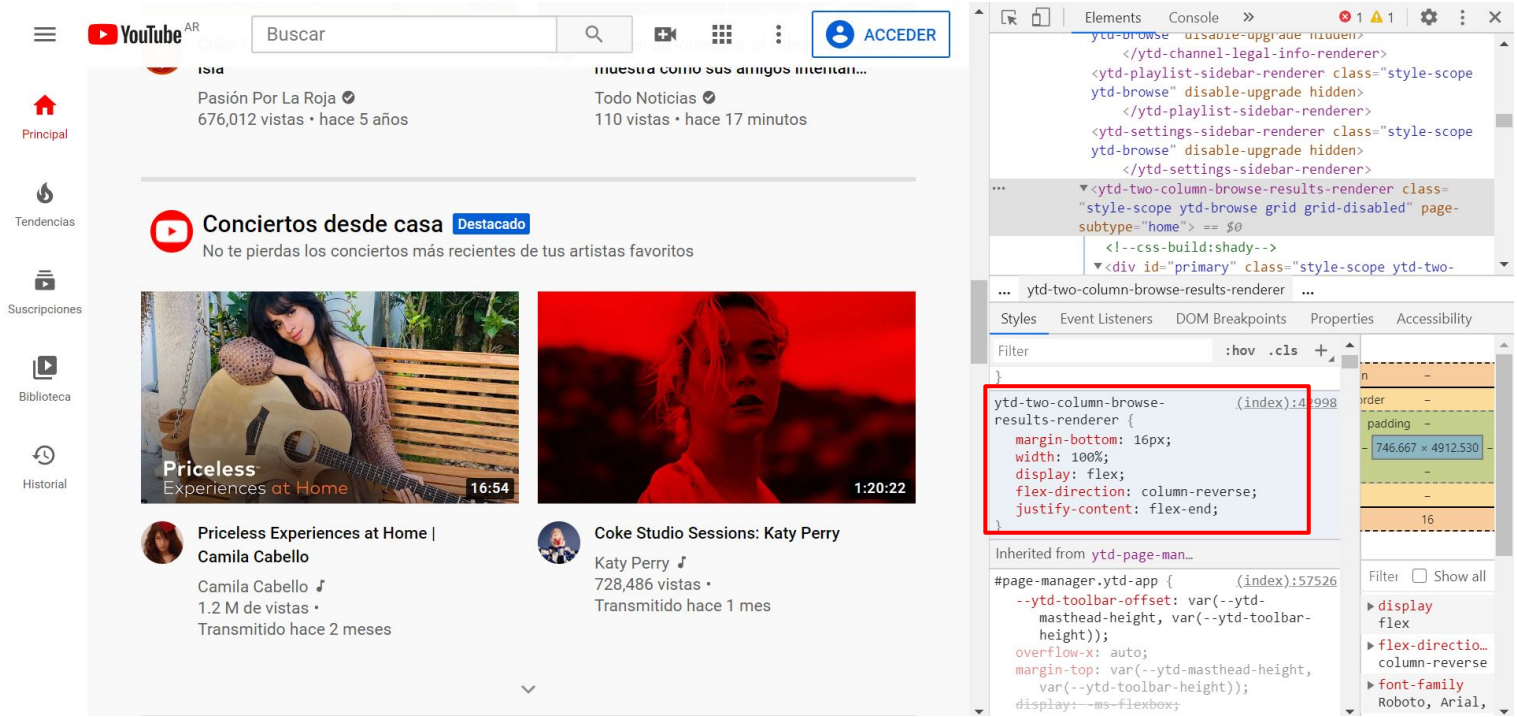
¿QUÉ ES FLEXBOX?

👉 Flexbox **es un modo de diseño** que nos permite crear estructuras para sitios web de una forma más fácil. Como vimos en la clase anterior, utilizando la propiedad float podíamos desplazar contenedores.

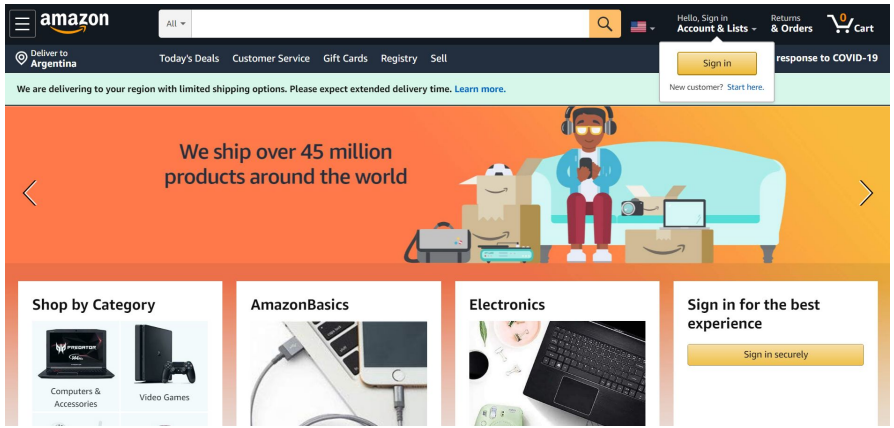
¿QUÉ ES FLEXBOX?

- Es un **conjunto de propiedades** de CSS.
- Se basa sobre un contenedor (padre) para ordenar a sus ítems (hijos).
- No sólo se podrá posicionar elementos vertical y horizontalmente, sino que podrás establecer cómo se distribuirán, el orden que tendrán e incluso el tamaño en proporción a otros elementos.

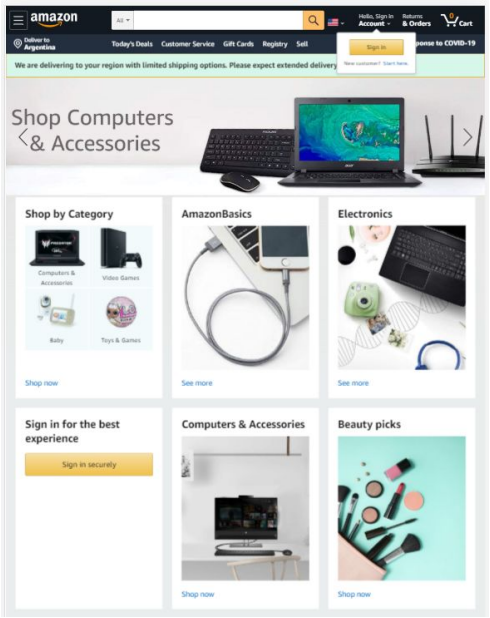
Páginas con Flexbox:



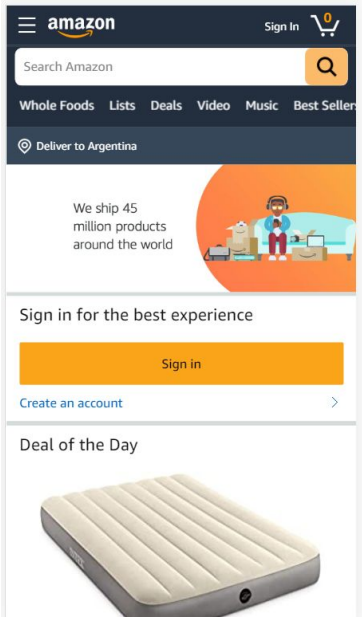
Páginas con Flexbox:



Desktop



Tablet



Mobile

PROPIEDADES DE PADRES E HIJOS

CODER HOUSE

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 limite?
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	Alinear verticalmente a los hijos.
align-content	Alinear verticalmente a los hijos cuando son multilínea.

¿CÓMO EMPEZAMOS?

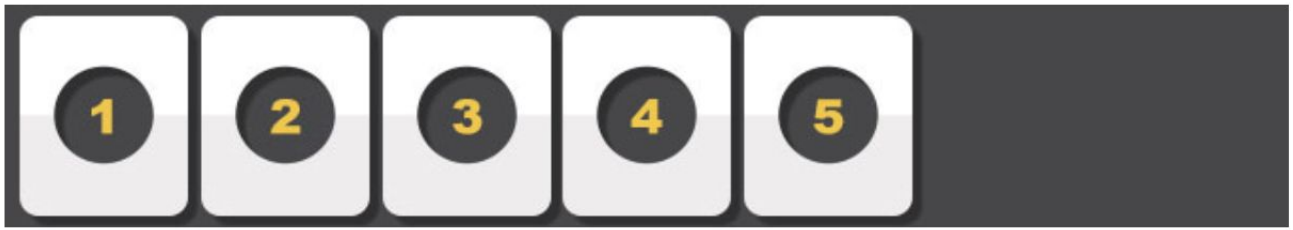
👉 Lo primero que debemos hacer es establecer la propiedad display con el valor flex en el elemento padre.

```
.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 */
}
```

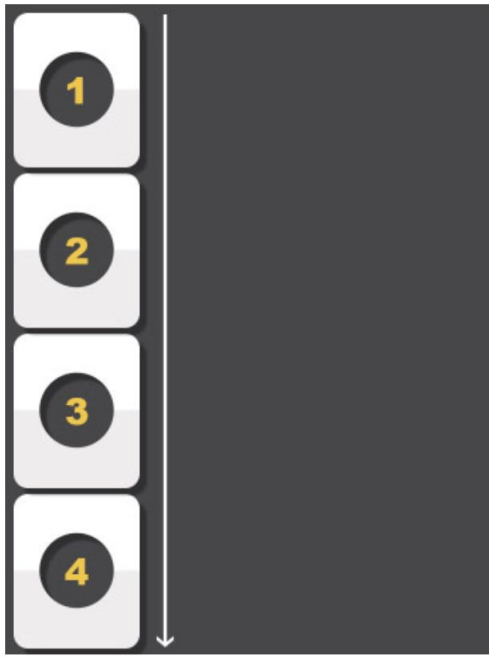


FLEX-DIRECTION: ROW-REVERSE

👉 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;
}
```

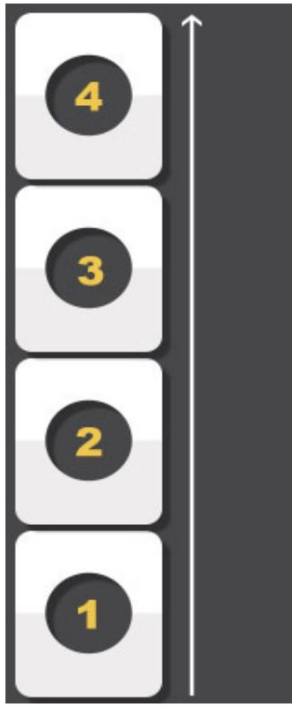
FLEX-DIRECTION: COLUMN



👉 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-WRAP

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

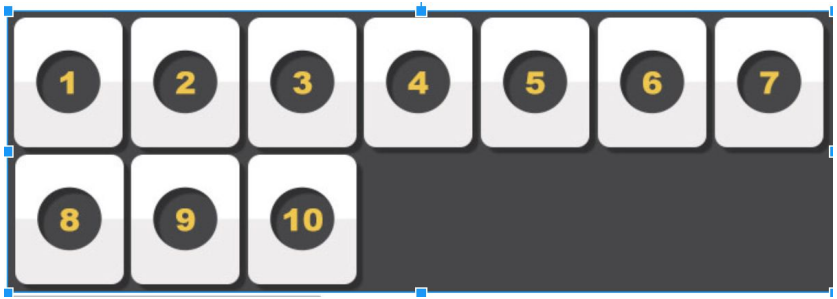
👉 Con *flex-wrap* vamos a poder **especificar** si queremos que los ítems puedan saltar a una nueva línea, cuando el contenedor flexible se quede sin espacio.

FLEX-WRAP: NOWRAP

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

FLEX-WRAP: WRAP

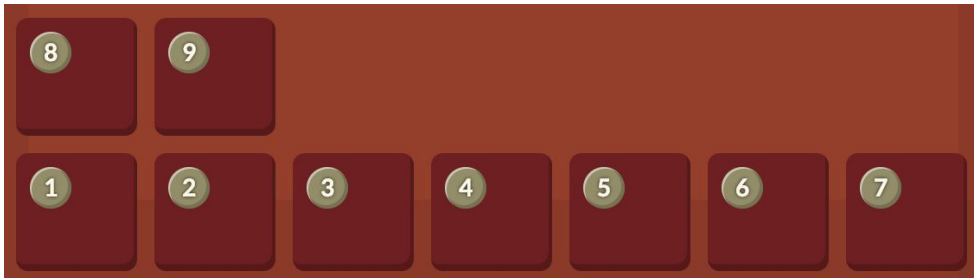
Los *flex items (hijos)* pueden romper la línea del eje horizontal, si les es necesario para conservar las características de sus dimensiones. Esto es de izquierda a derecha, y de arriba a abajo.




```
.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;
}
```

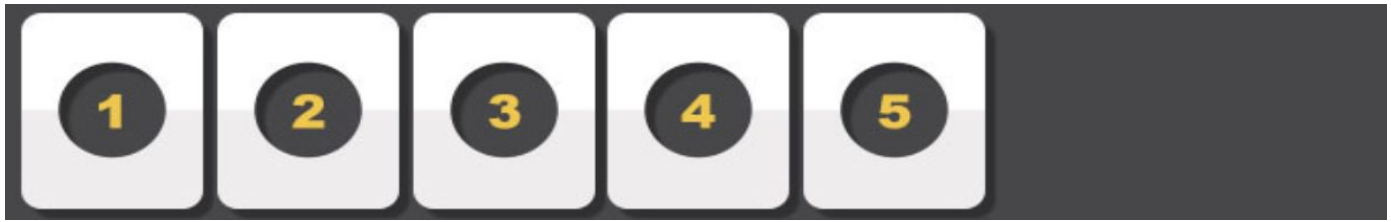
JUSTIFY-CONTENT

- *Justify-content* nos va a permitir alinear los elementos.
- Esto puede ser de forma vertical u 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 parten de la base del contenedor en “row”.**

JUSTIFY-CONTENT: FLEX-START

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

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



JUSTIFY-CONTENT: FLEX-END

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

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

JUSTIFY-CONTENT: CENTER

Consiste en alinear los *flex items (hijos)* al centro.

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

JUSTIFY-CONTENT: SPACE-BETWEEN

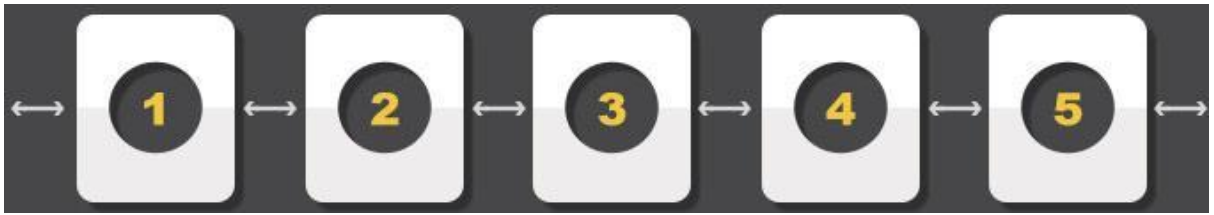
Es hacer que los *flex items (hijos)* tomen la misma distancia o espaciado entre ellos dentro del **contenedor flexible**, quedando el primer y último elemento pegados con los bordes del contenedor en el eje principal.

```
.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.
El espaciado entre los bordes lo toman del contenedor padre.

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



JUSTIFY-CONTENT: SPACE-EVENLY

Hace que el espacio entre los *flex items (hijos)* sea igual. **No es lo mismo que space-around.**

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

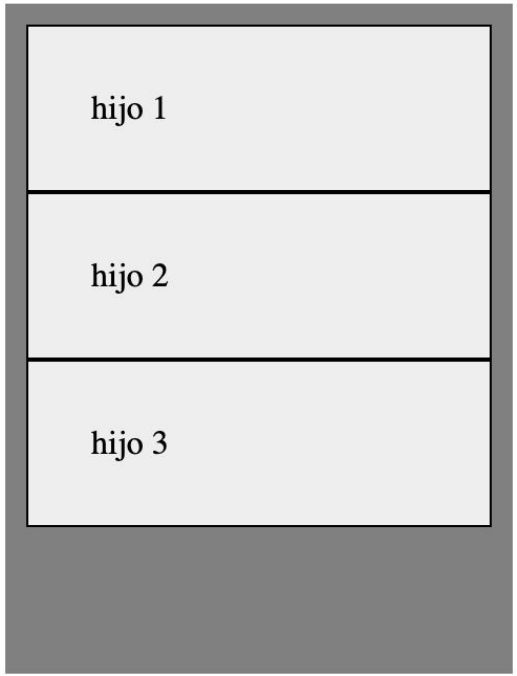
JUSTIFY-CONTENT

Los siguientes **ejemplos** parten de la base del contenedor en “column”, con altura definida:

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

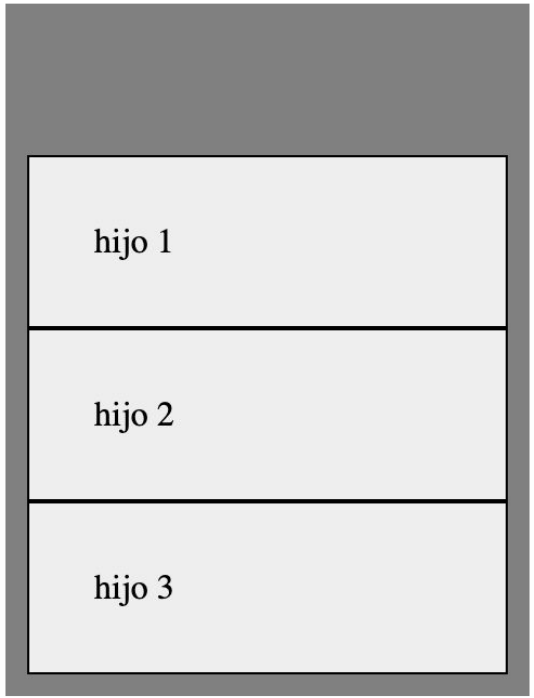
JUSTIFY-CONTENT: FLEX-START

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



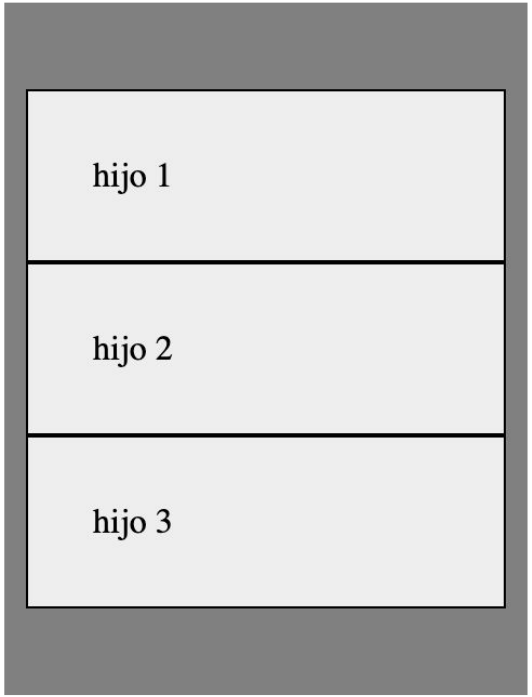
JUSTIFY-CONTENT: FLEX-END

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



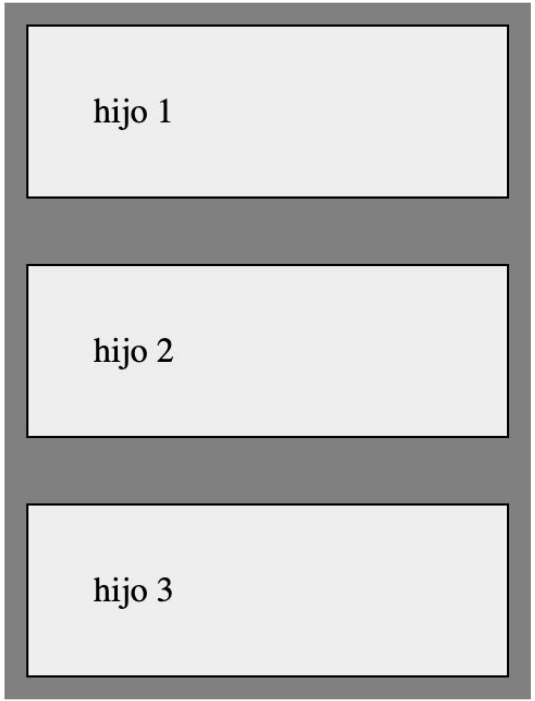
JUSTIFY-CONTENT: CENTER

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



JUSTIFY-CONTENT: SPACE-BETWEEN

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

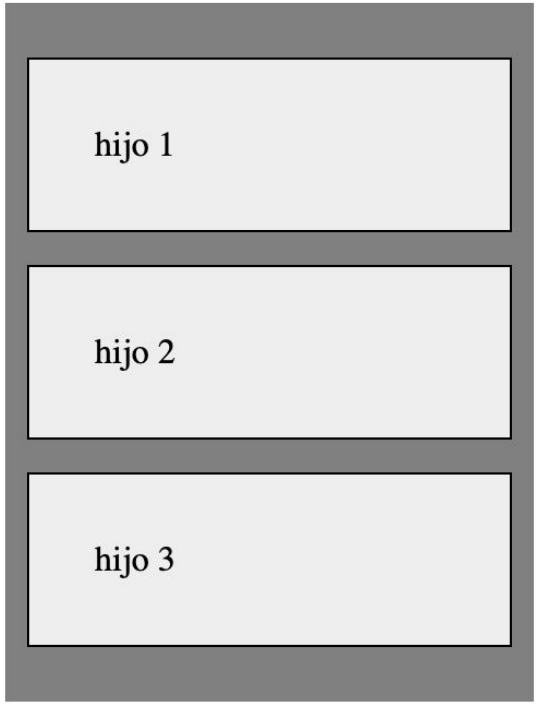


JUSTIFY-CONTENT: SPACE-AROUND

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

JUSTIFY-CONTENT: SPACE-EVENLY

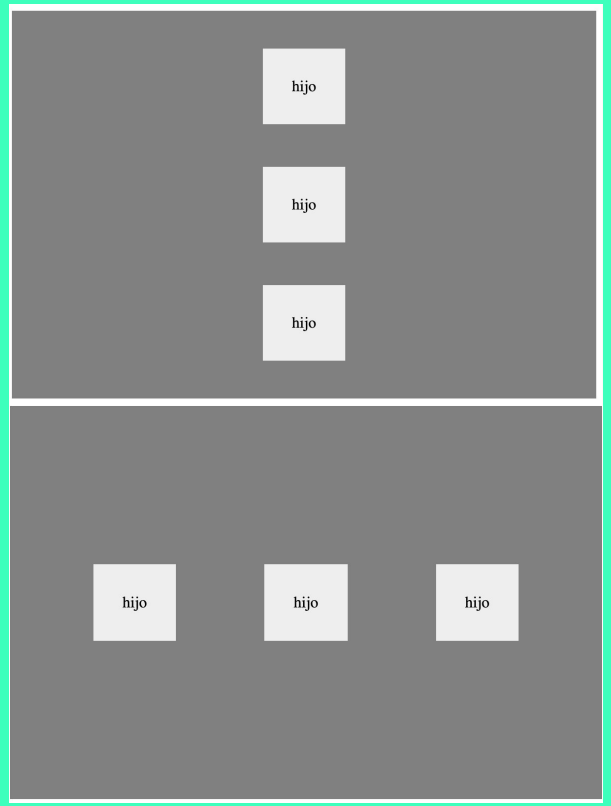
```css
.padre-flex {
    display: flex;
    flex-direction: column;
    justify-content: space-evenly;
    height: 300px;
}
```



ALIGN-ITEMS

👉 Alinear los elementos verticales de forma horizontal.
(flex-direction: column)

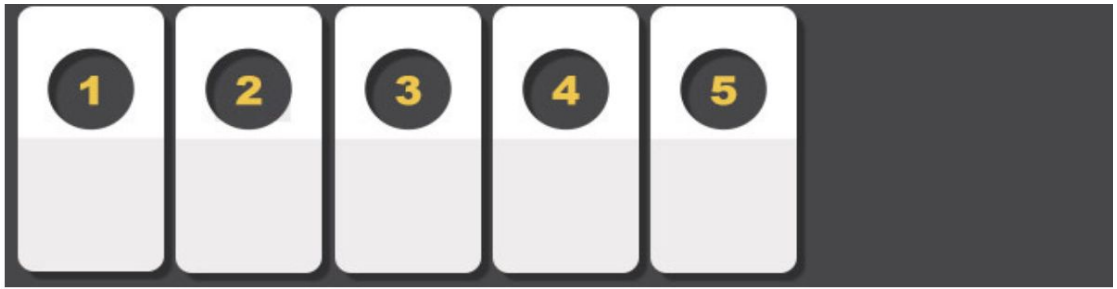
👉Alinear los elementos horizontales de forma vertical.
(flex-direction: row)



ALIGN-ITEMS: STRETCH

Tratará 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;
}
```

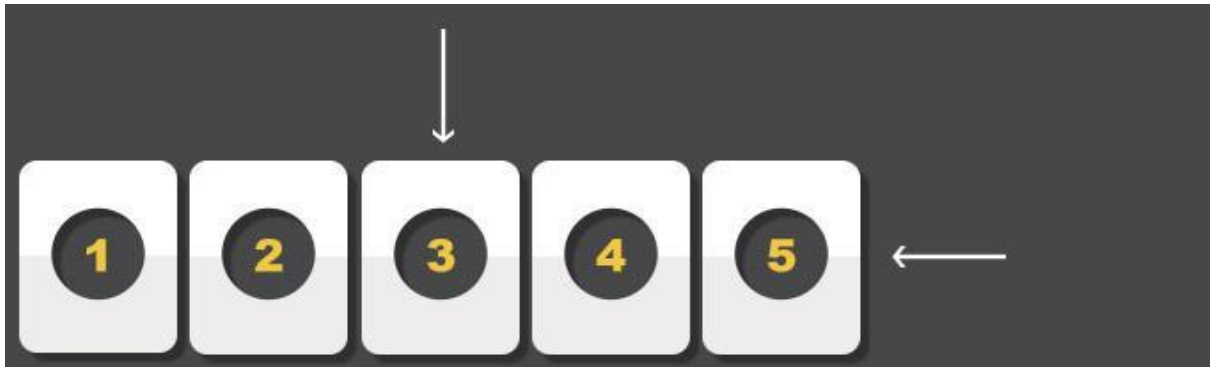


ALIGN-ITEMS: FLEX-START

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

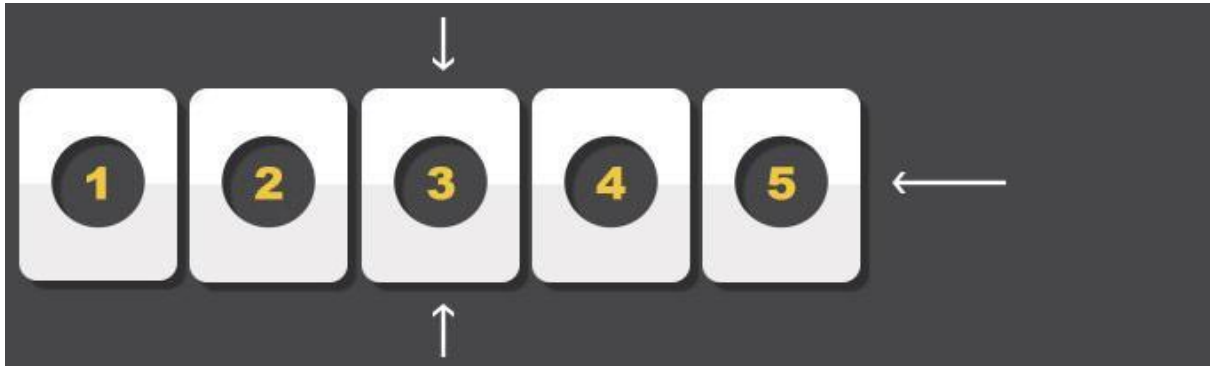
ALIGN-ITEMS: FLEX-END

```
.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-CONTENT

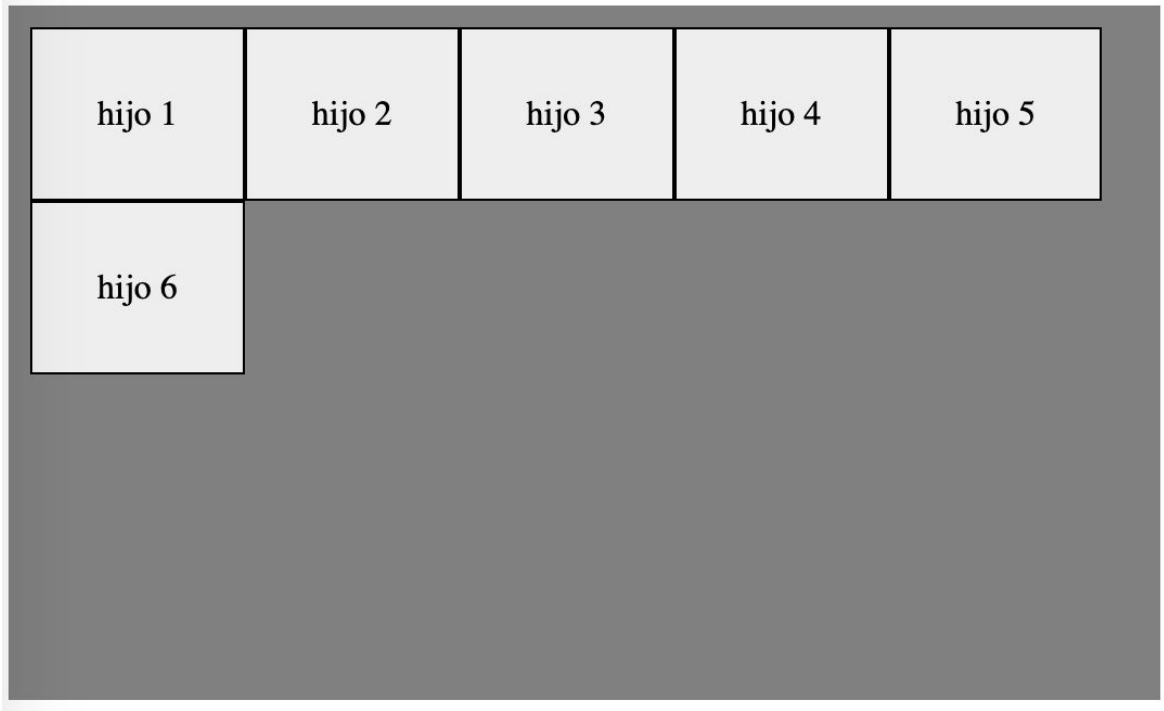
👉Esta propiedad sólo tiene efecto cuando el contenedor flexible tiene varias líneas de *flex items (hijos)*.
👉Si se colocan en una sola línea, esta propiedad no tiene ningún efecto sobre el diseño.

👉Para poder aplicarlo se necesita tener el atributo *flex-wrap,* que permita verificar los ejes horizontales.

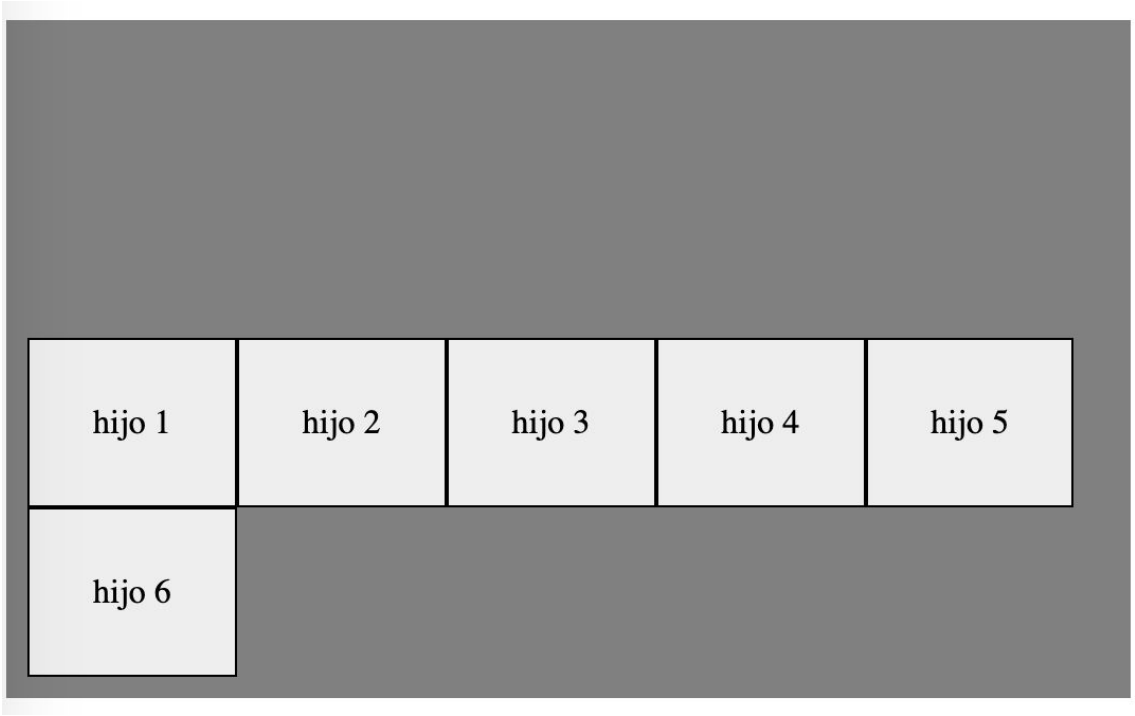
```
.padre-flex {
   display: flex; flex-wrap: wrap; flex-direction: row;
   align-content:stretch; /* predeterminada */
}
```

ALIGN-CONTENT: STRETCH

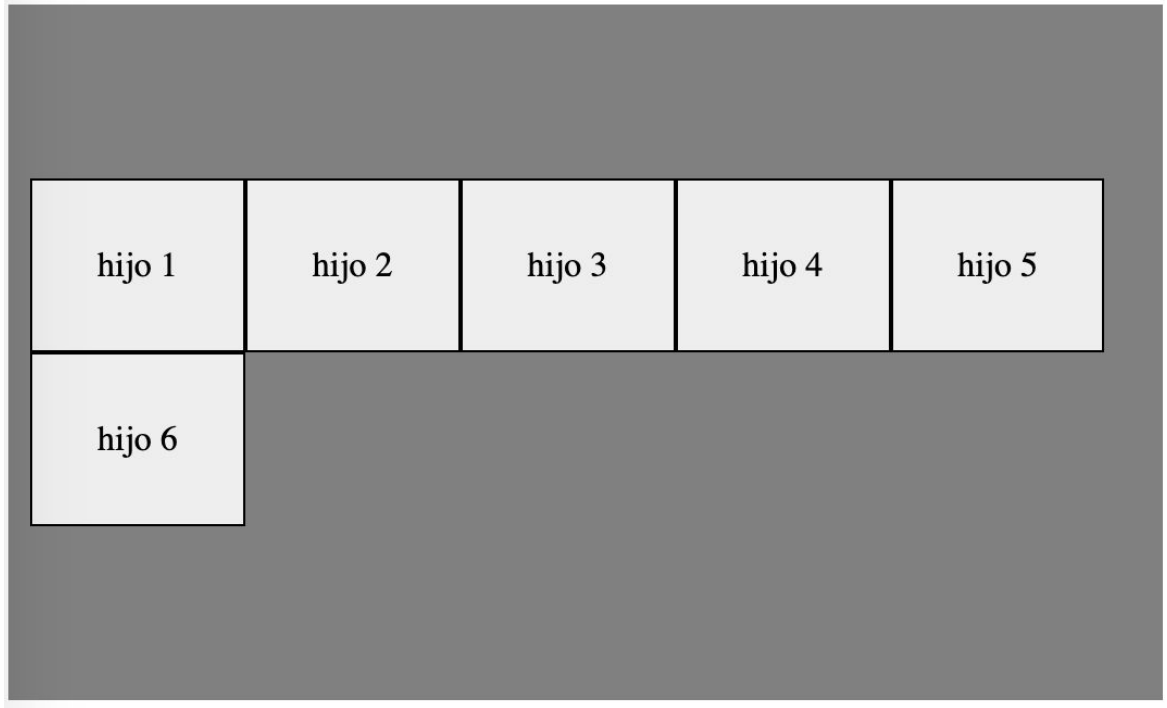
ALIGN-CONTENT: FLEX-START



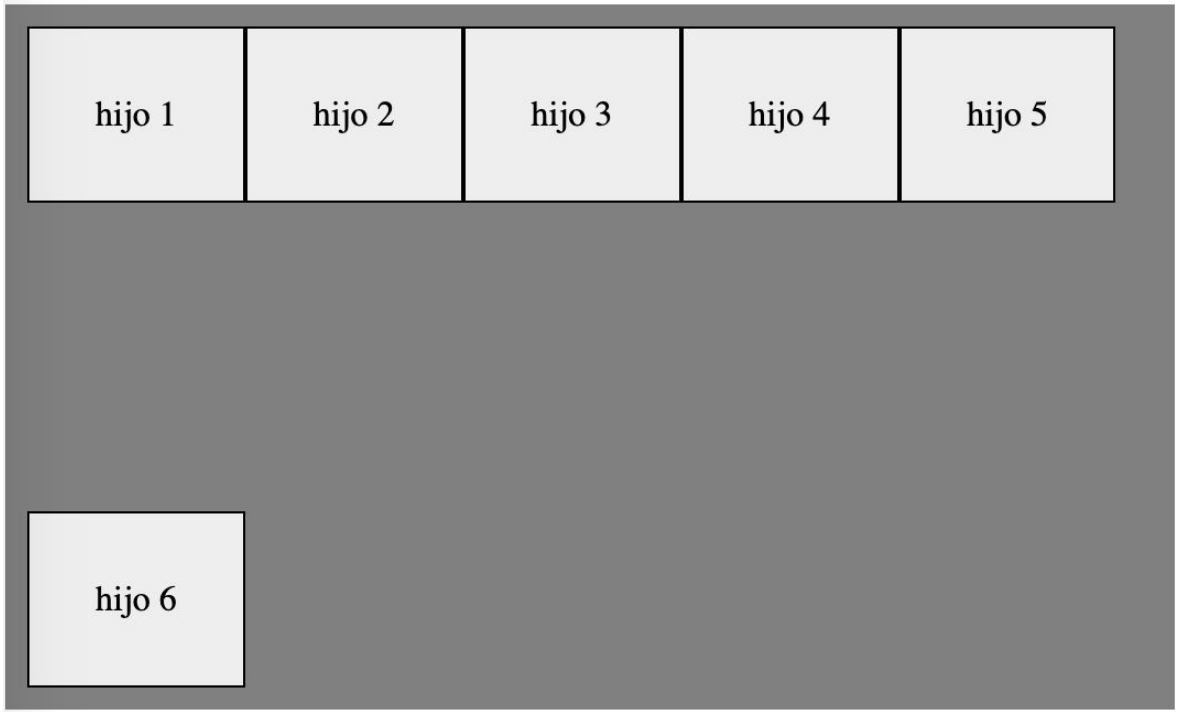
ALIGN-CONTENT: FLEX-END



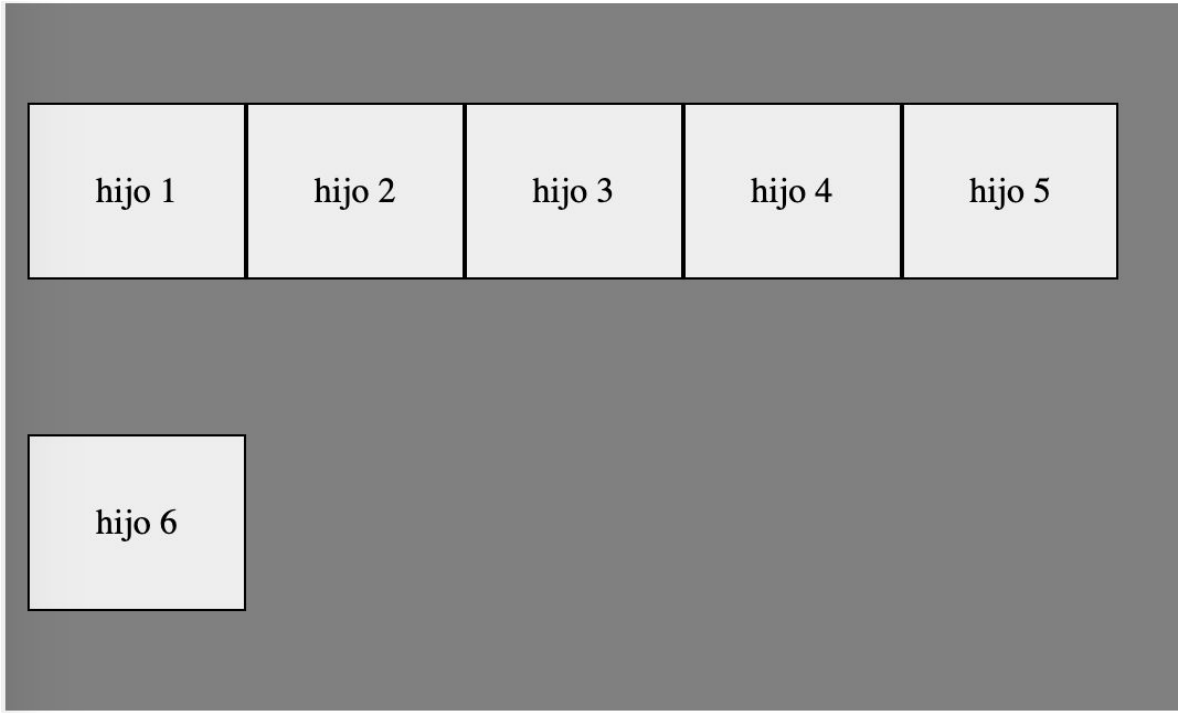
ALIGN-CONTENT: CENTER



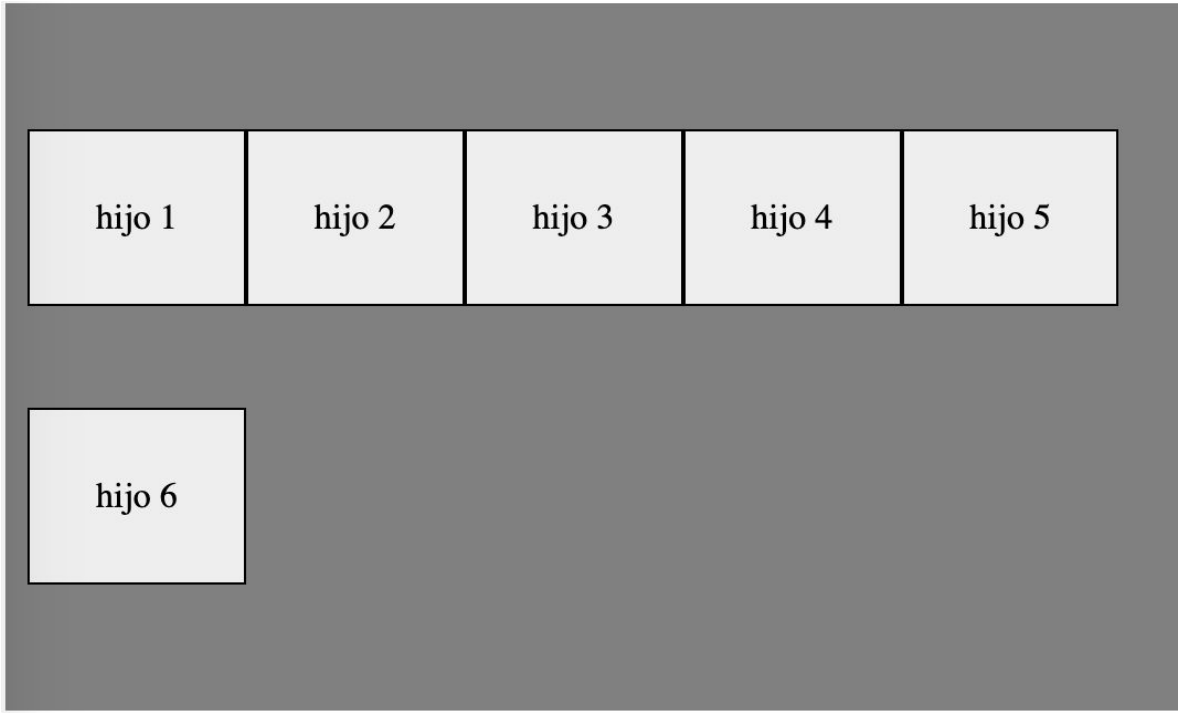
ALIGN-CONTENT: SPACE-BETWEEN



ALIGN-CONTENT: SPACE-AROUND



ALIGN-CONTENT: SPACE-EVENLY



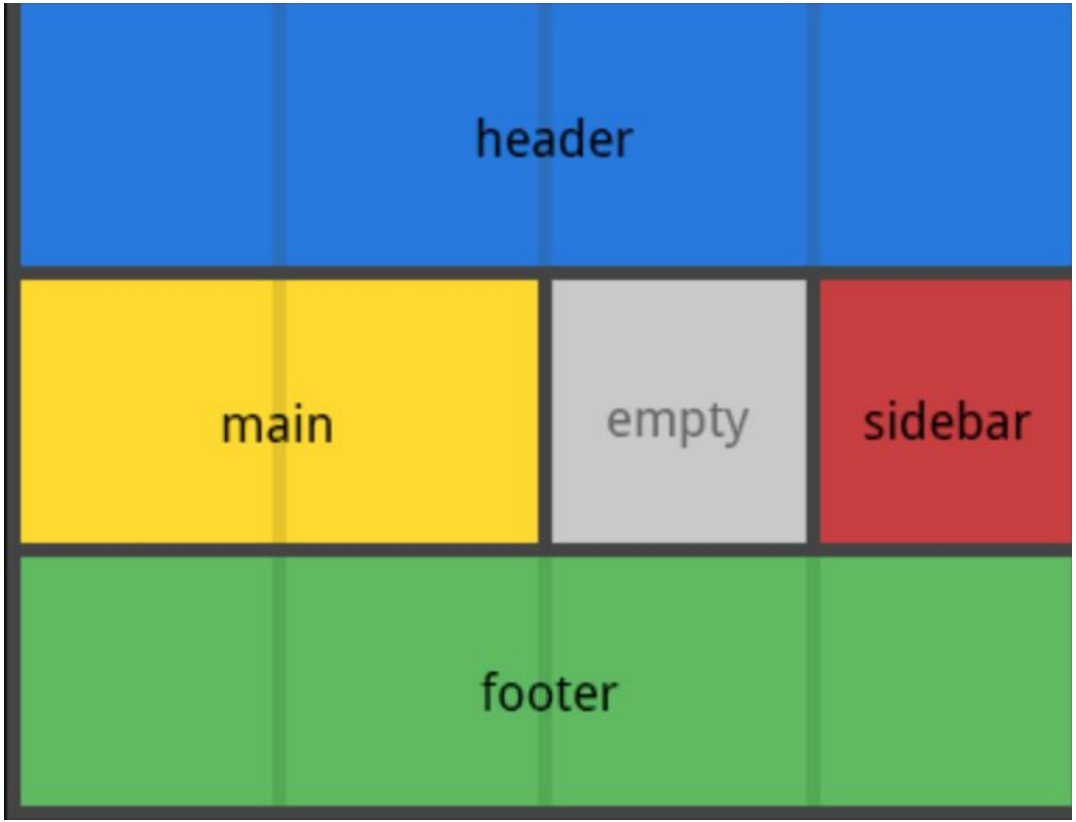
GRIDS

¿QUÉ ES GRIDS?

👉CSS Grid es el sistema de maquetación más potente que hay disponible. Se trata de un sistema en 2D que permite definir filas y columnas (a diferencia de Flexbox, el cual funciona en una única dimensión).

👉El grid layout permite alinear elementos en columnas y filas. Sin embargo, son posibles más diseños con CSS grid que como lo eran con las tablas.

👉Por ejemplo, los elementos secundarios de un contenedor de cuadrícula podrían posicionarse de manera que se solapen y se superpongan, similar a los elementos posicionados en CSS.



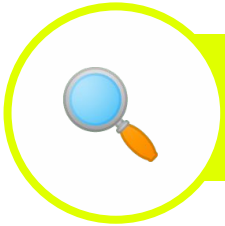
El CSS grid se puede utilizar para lograr muchos diseños diferentes. **Se destaca por dividir una página en regiones principales, o definir la relación en términos de tamaño, posición y capas, entre partes de un control.**

IMPLEMENTAR GRIDS

Los complementos vistos anteriormente suelen ser insuficientes, o a veces un poco complejos para crear un layout/estructuras para páginas web actuales.

Flexbox fue una gran mejora, pero está orientado a estructuras de una sola dimensión.

Muchos frameworks y librerías utilizan un sistema grid, donde definen una cuadrícula determinada, y cambiando los nombres de las clases de los elementos HTML es posible trabajar muchos atributos.



¿POR QUÉ GRIDS?

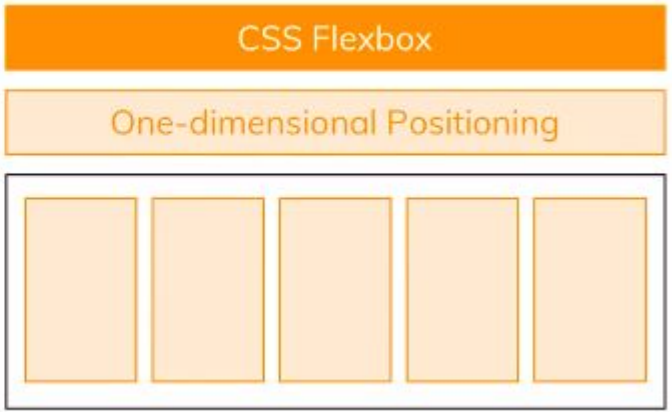
👉 **Grid CSS surge de la necesidad de algo más potente**, y toma las ventajas del sistema Flexbox, sumándole muchas mejoras y características que permiten crear muy **rápido** cuadrículas sencillas y potentes.

👉 Grid toma la filosofía y **la base del sistema Flexbox**. Esto no significa que lo reemplaza, sino que **pueden convivir**.

Está pensado para estructuras grandes y complejas.

DIFERENCIA ENTRE FLEXBOX Y GRIDS

Flexbox

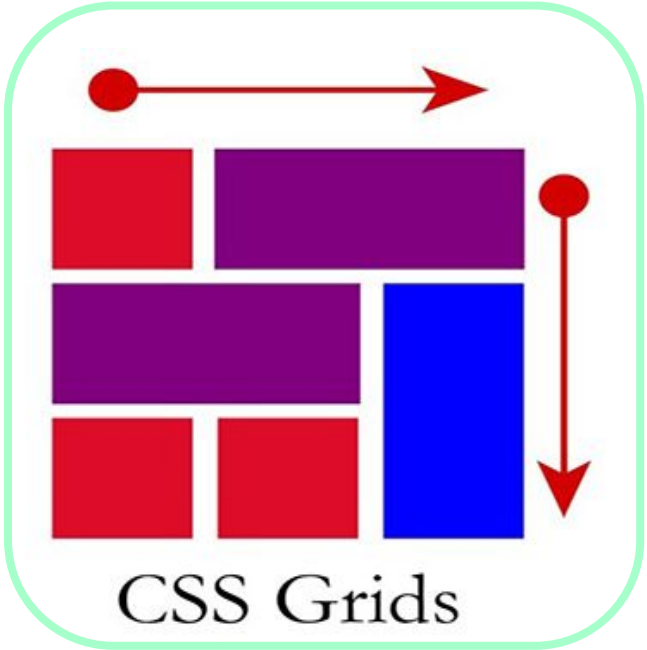


Grids

Ambos son mucho más potentes que cualquier técnica que haya existido antes.

DIFERENCIA ENTRE FLEXBOX Y GRIDS

UNIDIMENSIONAL

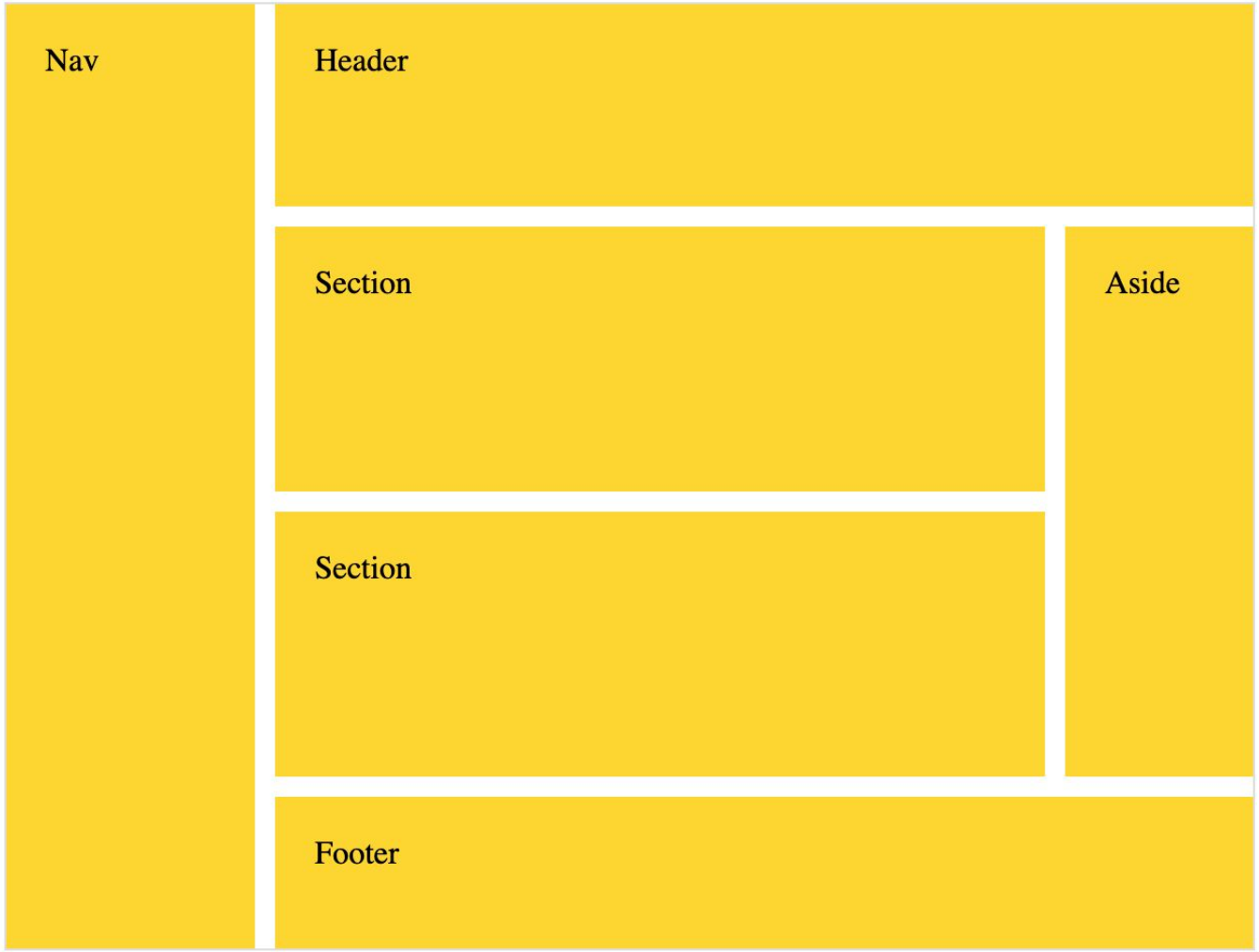


BIDIMENSIONAL

PROPIEDADES DEL PADRE

En primer lugar, aplicas la propiedad “**display: grid**” al elemento padre.
Luego puedes usar lo siguiente para crear la estructura principal:

grid-template-columns	Establece el **TAMAÑO** de cada **columna** (*col 1, col 2...*).
grid-template-rows	Establece el **TAMAÑO** de cada **fila** (*fila 1, fila 2...*).
grid-template-areas	Indica la **disposición** de las áreas en el grid. Cada texto entre comillas simboliza una fila.
grid-column-gap	Establece el **TAMAÑO** de los **huecos** entre columnas (*líneas verticales*).
grid-row-gap	Establece el **TAMAÑO** de los **huecos** entre filas (*líneas horizontales*).



EL OBJETIVO

Este tipo de estructura no es posible con Flexbox, por eso podemos pensar en usar Grid

FILAS Y COLUMNAS EXPLÍCITAS

Es posible crear cuadrículas con un tamaño **definido**. Para ello, sólo tienes que usar las propiedades CSS *grid-template-columns* y *grid-template-rows*, las cuales sirven para indicar las dimensiones de cada celda de la cuadrícula, diferenciando entre columnas y filas.

Propiedad	Descripción
grid-template-columns	Establece el tamaño de las columnas (eje horizontal).
grid-template-rows	Establece el tamaño de las filas (eje vertical).

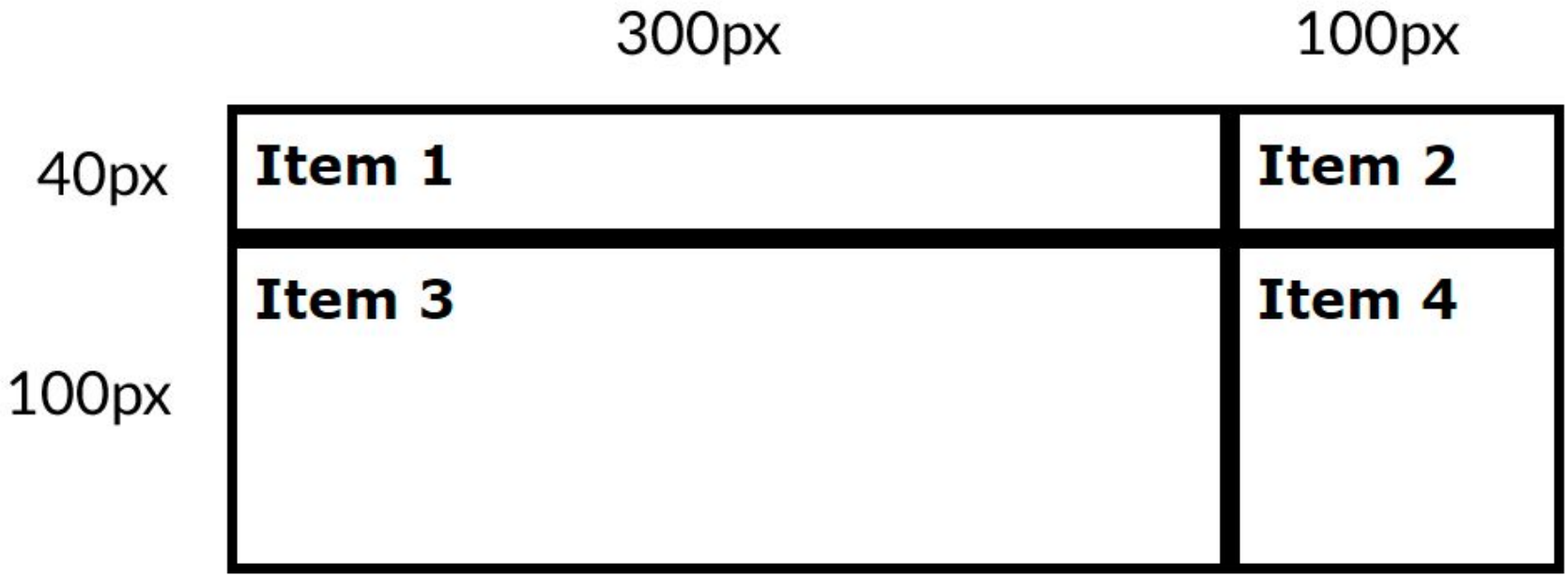
FILAS Y COLUMNAS EXPLÍCITAS

👀 Veamos la forma más simple de crear una grilla, especificando cuántas columnas y filas queremos.

```css
.grid {
   display: grid;
                        /* 2 columnas */
   grid-template-columns: 300px 100px;
                        /* 2 filas */
   grid-template-rows: 40px 100px;
}
```

```html
<section class="grid">
   <div>Item 1</div>
   <div>Item 2</div>
   <div>Item 3</div>
   <div>Item 4</div>
</section>
```

FILAS Y COLUMNAS EXPLÍCITAS



FILAS Y COLUMNAS EXPLÍCITAS

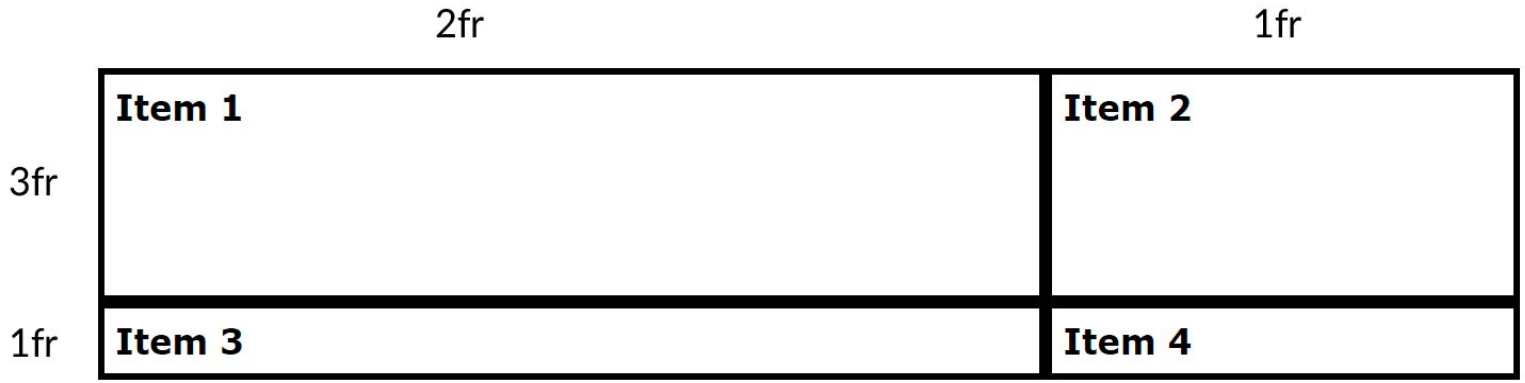
Unidad creada para ser usada en grid (fr (fraction))

```
.grid {
     display: grid;
     grid-template-columns: 2fr 1fr;
     grid-template-rows: 3fr 1fr;
}
```

Nota: también es posible utilizar otras unidades y combinarlas, como porcentajes o la palabra clave auto (que obtiene el tamaño restante).

FILAS Y COLUMNAS EXPLÍCITAS

Cuadrícula de *2x2*, donde el tamaño de ancho de la cuadrícula se divide en **dos columnas** (una el doble de tamaño que la siguiente), y el tamaño de alto de la cuadrícula se divide en dos filas, donde la primera ocupará el triple (3 fr) que la segunda (1 fr):



FILAS Y COLUMNAS REPETITIVAS

Si necesitas hacer muchas columnas y filas iguales, puedes usar lo siguiente:

```
.grid {
     display: grid;
     grid-template-columns: repeat(12, 1fr);
     grid-template-rows: repeat(12, 1fr)
}
```

repeat(*[núm de veces]*, *[valor o valores]*)

GRID ESPACIOS

La cuadrícula tiene todas sus celdas **una a continuación de la otra.** Aunque sería posible darle un margen a las celdas dentro del contenedor, existe una forma más apropiada, evitando los problemas clásicos de los modelos de caja: los **huecos (gutters)**.

```
.grid {
   grid-column-gap: 10px;
   grid-row-gap: 15px;
}
```

GRID ESPACIOS