

CSS

Estructura en nuestros estilos



GRID

Sistema de grillas

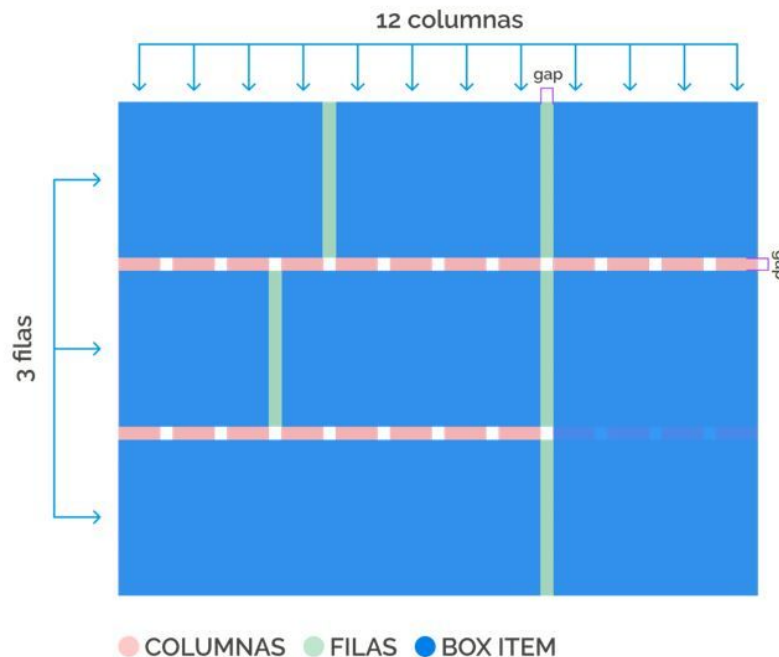
¿Qué es GRID?

A diferencia de **Flexbox**, donde podemos posicionar nuestros elementos en **una sola dimensión**, con **GRID** tenemos la posibilidad de hacerlo de forma horizontal y vertical al mismo tiempo, es decir, **en 2 dimensiones**.

GRID toma todas las ventajas de Flexbox para volcarlo en un **sistema más potente** que nos permite **crear grillas o cuadrículas** de una manera muy sencilla.

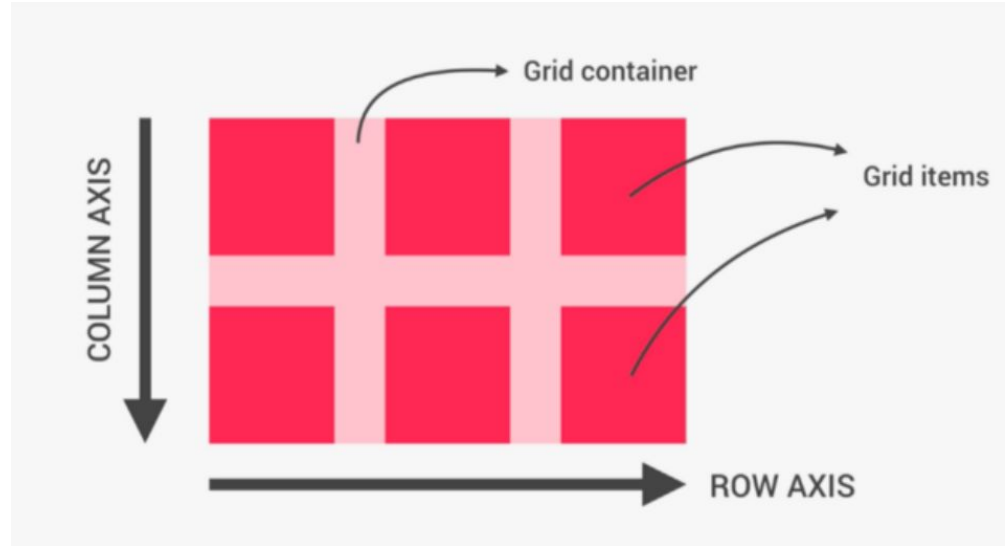
Poder en 2 dimensiones

Como mencionamos anteriormente **GRID** nos permite trabajar en **filas y columnas** simultáneamente, solo es cuestión de crear una grilla o cuadrícula que se **ajuste a nuestras necesidades** y **posicionar** los elementos **hijos** dentro de ella.



Ejes

En el caso de **GRID** se toma como **main axis** al eje **X** (row axis) y como **cross axis** al eje **Y** (column axis), lo que tendrá vital importancia al momento de utilizar las propiedades de **alineación** que veremos en un momento.



Propiedades

Al igual que **Flexbox**, **Grid Layout** está compuesto por un conjunto de propiedades aplicadas a un **elemento padre** que *definirá* una grilla modelo o **template** con la forma buscada. Sobre este template es que luego posicionaremos los **elementos hijos**.

Cabe destacar que al ser una **plantilla** esta **no forma parte de la estructura**, sino que genera un lienzo cuadriculado sobre el cual **distribuiremos** nuestros **elementos**.

Declaración de una Grilla con GRID

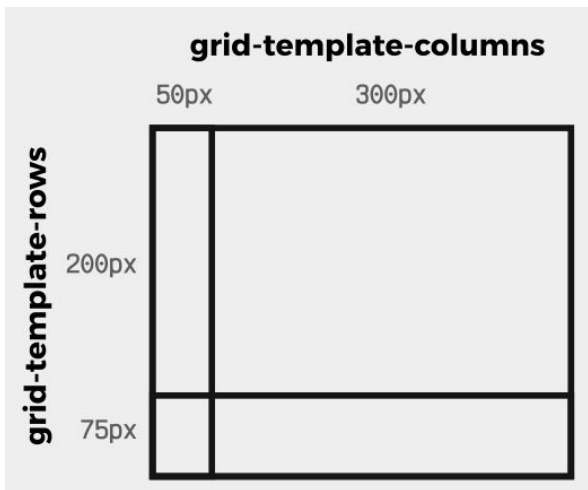
Para comenzar a usar **GRID** es necesario declarar a nuestro contenedor padre con un display en forma de grilla. Nuevamente **usamos** la propiedad **display**, pero ahora con el valor: **grid**;

```
<section id="gallery">
  <picture class="gallery__img-1">
    
  </picture>
  <picture class="gallery__img-2">
    
  </picture>
  <picture class="gallery__img-3">
    
  </picture>
  <picture class="gallery__img-4">
    
  </picture>
  <picture class="gallery__img-5">
    
  </picture>
  <picture class="gallery__img-6">
    
  </picture>
</section>
```

```
#gallery {
  display: grid;
}
```

Filas y Columnas

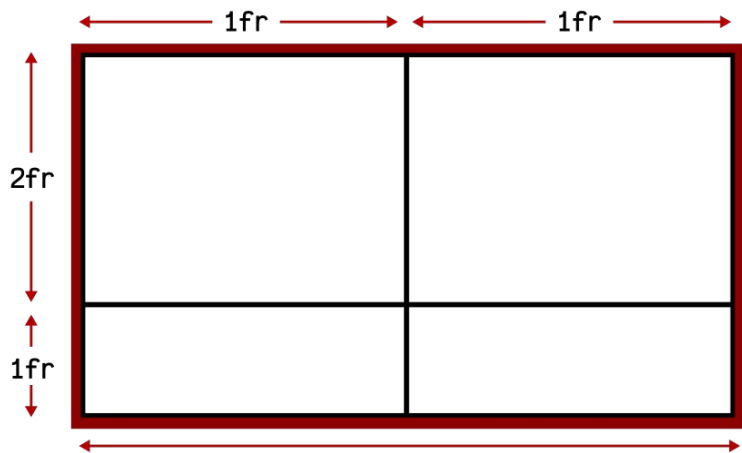
Una vez que **declaramos** nuestro contenedor como una **grilla**, debemos **elegir** la cantidad de **filas y columnas** que vamos a necesitar en nuestro **template**. Para eso vamos a utilizar las propiedades **grid-template-columns** y **grid-template-rows** de la siguiente manera:



```
.grid-container {  
  display: grid;  
  grid-template-columns: 50px 300px;  
  grid-template-rows: 200px 75px;  
}
```


Fractions

Si bien `grid-template-columns` y `grid-template-rows` aceptan las unidades de medida tradicionales, en GRID existe la unidad fr (fraction) que divide el espacio disponible entre la cantidad de fr declarados y los reparte proporcionalmente.



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

En este ejemplo tanto el width cómo el height se dividen por la cantidad de fracciones declaradas y lo distribuye en consecuencia.

Cabe destacar que fr se puede combinar con cualquier otra unidad de medida en la misma declaración.

Repeat

En los casos donde necesitamos crear una **plantilla** con **muchas** filas o columnas del **mismo tamaño**, podemos utilizar el valor `repeat(cantidad, tamaño);`.

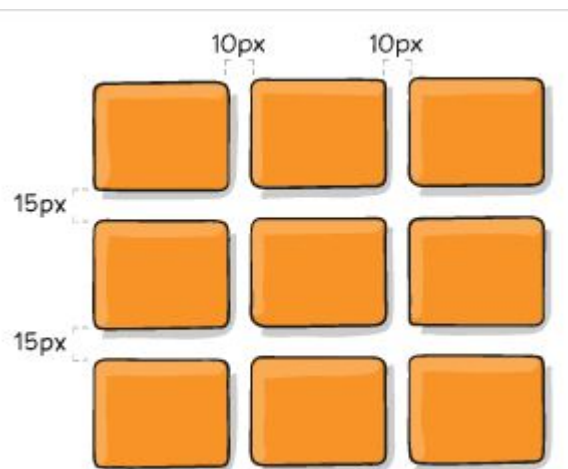
```
.grid-container {  
  display: grid;  
  grid-template-columns: repeat(12, 1fr);  
  grid-template-rows: repeat(3, 250px);  
}
```

cantidad: es el **número** de columnas o filas que necesitamos.

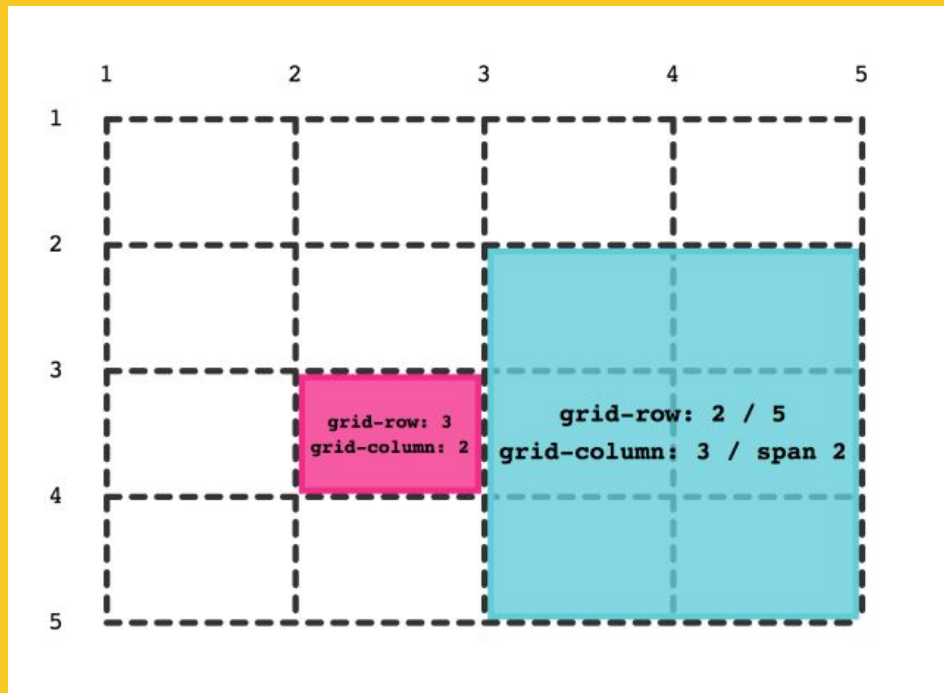
tamaño: el **ancho** o **alto** que *deberán tomar*.

GAP

Esta propiedad **mágica**, funciona igual que en flexbox y se utiliza para **definir** el **espaciado entre los elementos de una grilla**. En este caso **podemos** hacerlo de forma separada mediante las propiedades **column-gap** y **row-gap** o en conjunto a través de la propiedad **gap**.



GRID POR VALORES



grid-column

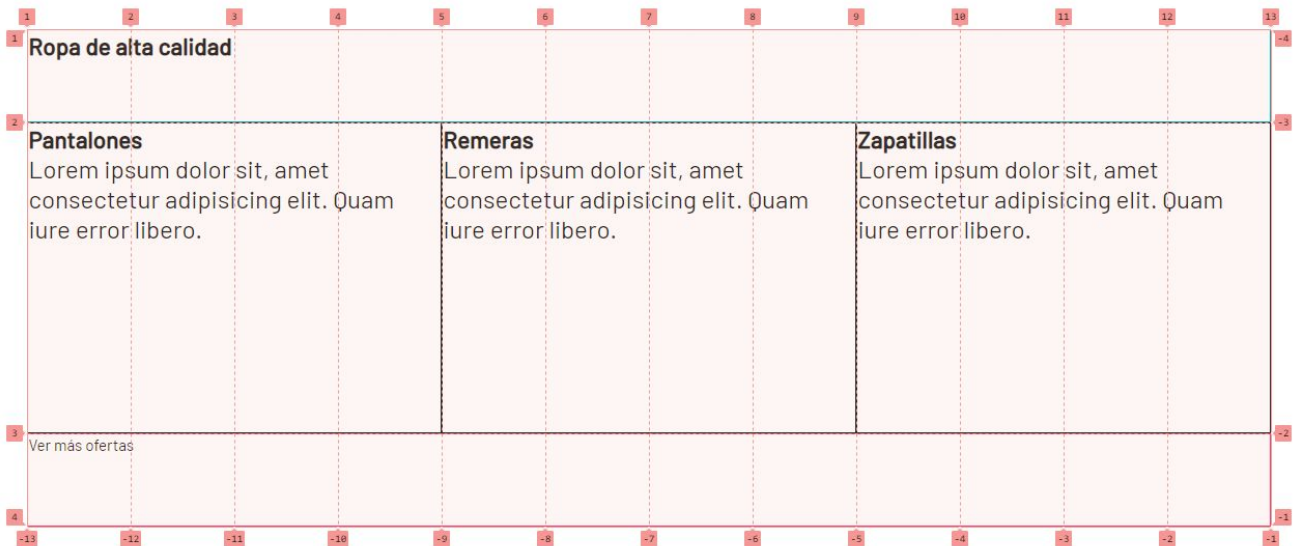
Shorthand de `grid-column-start` y `grid-column-end` nos va a permitir indicarle a un grid-item en qué columna debe empezar y en cuál finalizar `start / end;`.

```
.grid-container {  
  display: grid;  
  width: 800px;  
  grid-template-columns: repeat(12, 1fr);  
  grid-template-rows: 60px 200px 60px;  
}  
  
.grid-container h1,  
.grid-container a {  
  grid-column: 1 / 13;  
}  
  
.grid-item-1 {  
  grid-column: 1 / 5;  
}  
  
.grid-item-2 {  
  grid-column: 5 / 9;  
}  
  
.grid-item-3 {  
  grid-column: 9 / 13;  
}
```

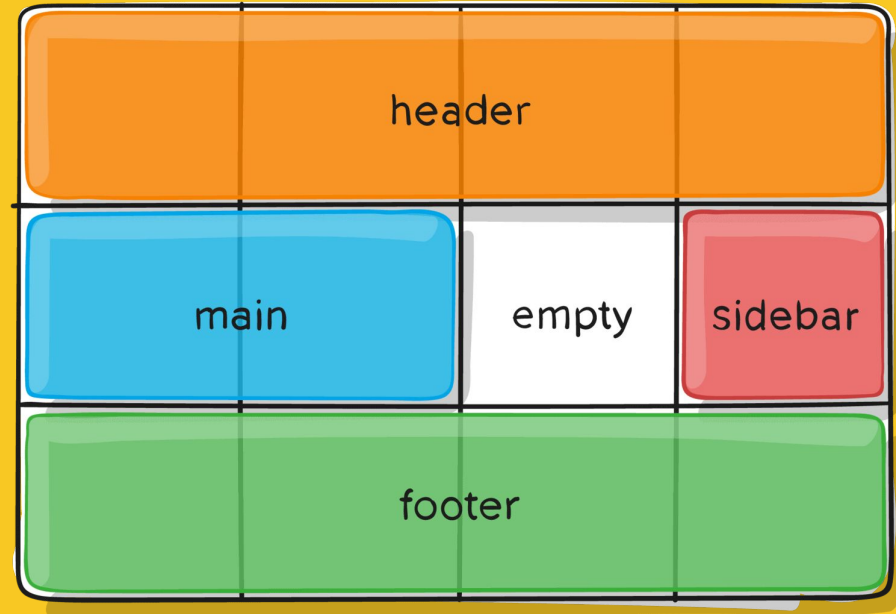
```
<section class="grid-container">  
  <h1>Ropa de alta calidad</h1>  
  <article class="grid-item-1">  
    <h2>Pantalones</h2>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
    Quam iure error libero.</p>  
  </article>  
  <article class="grid-item-2">  
    <h2>Remeras</h2>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
    Quam iure error libero.</p>  
  </article>  
  <article class="grid-item-3">  
    <h2>Zapatillas</h2>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
    Quam iure error libero.</p>  
  </article>  
  <a href="/ofertas.html">Ver más ofertas</a>  
</section>
```

Igual que el anterior, solo que en este caso nos va a permitir indicarle a un grid-item en qué **fila** debe **empezar y en cuál finalizar**.

Así nos queda nuestros elementos distribuidos en la grilla creada:

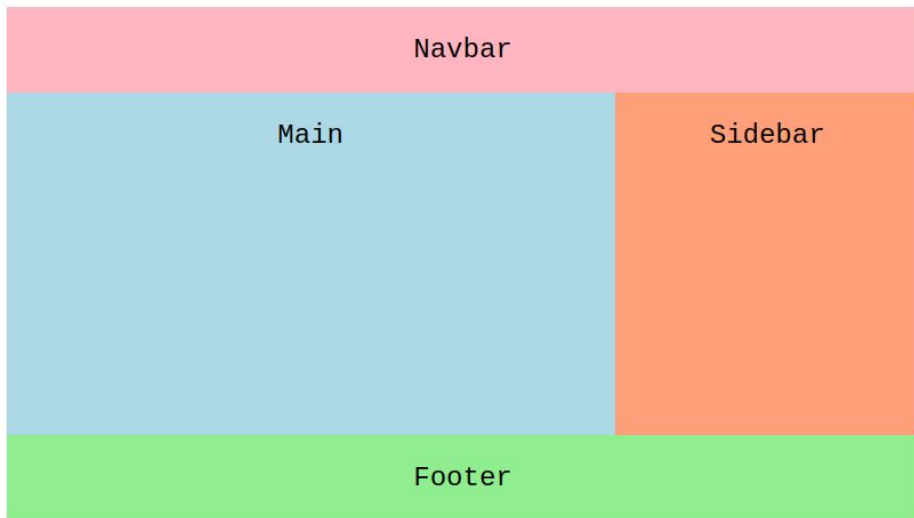


GRID AREAS



Areas

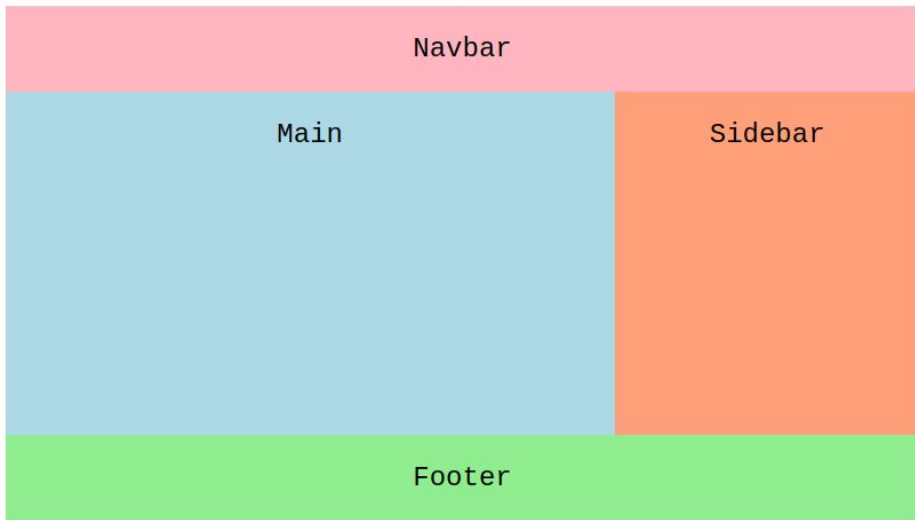
grid-template-areas: es una propiedad de GRID que nos permite “dibujar” nuestra plantilla CSS para luego posicionar a los elementos hijos en esa representación.



```
.grid-container {  
  display: grid;  
  grid-template-columns: 2fr 1fr;  
  grid-template-rows: 120px 1fr 120px;  
  grid-template-areas:  
    "navbar navbar"  
    "main sidebar"  
    "footer footer";  
}
```


Areas

Estas areas establecidas desde el `.grid-container` son las que aplicaremos luego a nuestros `.grid-items` para especificar qué espacio ocupará cada uno.



```
header {  
  grid-area: navbar;  
}  
  
main {  
  grid-area: main;  
}  
  
aside {  
  grid-area: sidebar;  
}  
  
footer {  
  grid-area: footer;  
}
```

Alineaciones

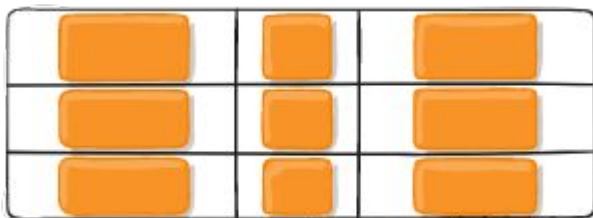
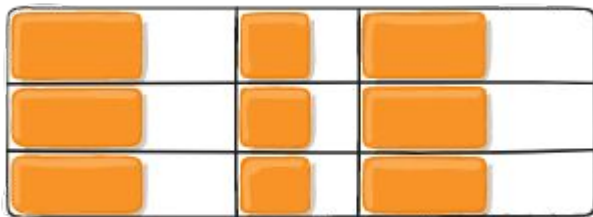
Al tomar las *mismas ventajas de Flexbox* para crear un **sistema** más potente, veremos que en **GRID** las variantes para **alinear nuestros elementos hijos** son muy **similares** a las que ya conocemos.

ITEMS

justify-items

`start | end | center | stretch;`

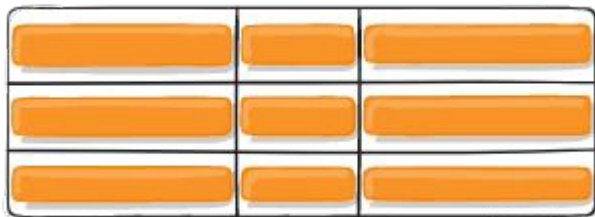
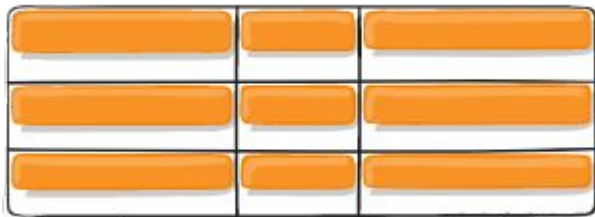
Se utiliza para alinear los ítems en el **eje horizontal** o **row axis**.



align-items

`start | end | center | stretch | baseline;`

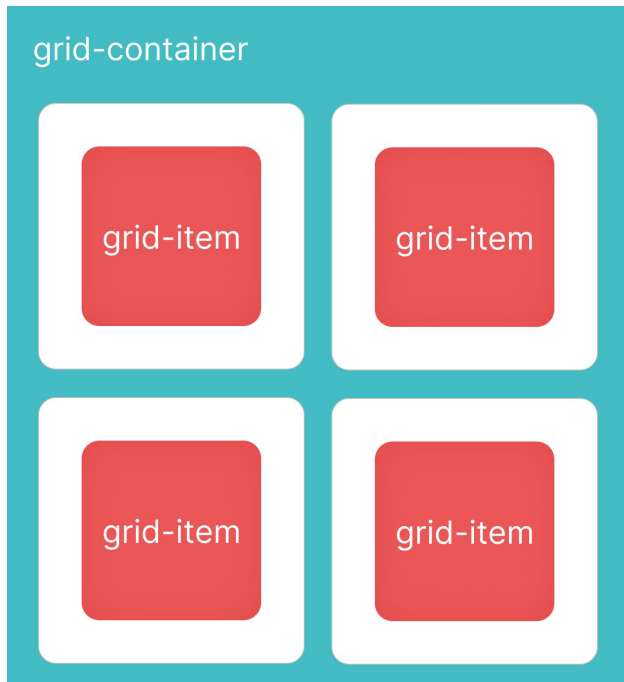
Se utiliza para alinear los ítems en el **eje vertical** o **column axis**.



place-items

start | end | center | stretch | baseline;

Nos permite definir `justify-items` y `align-items` con la misma propiedad cuando poseen el mismo valor.



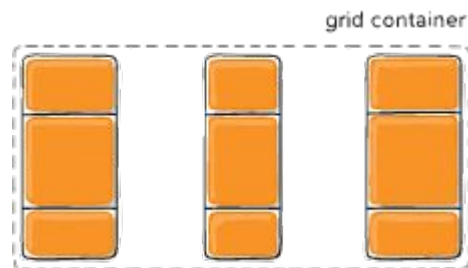
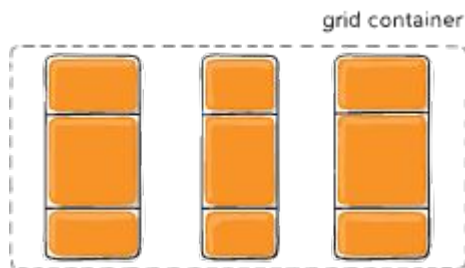
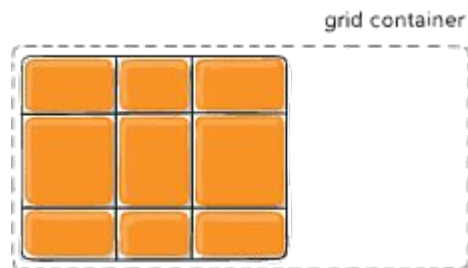
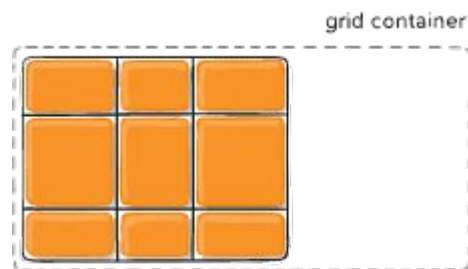
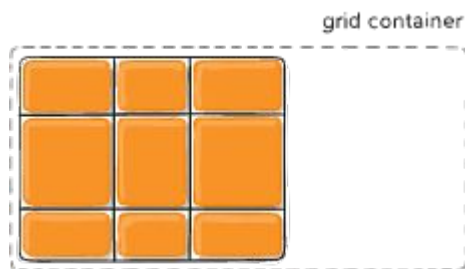
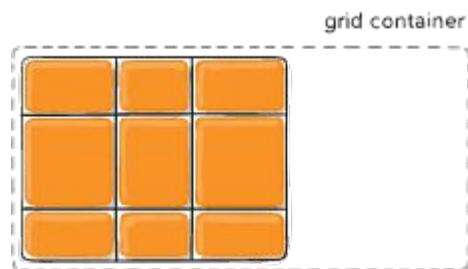
```
.grid-container {  
  display: grid;  
  place-items: center;  
}
```

CONTENT

justify-content

start | end | center | space-between | space-around | space-evenly | stretch;

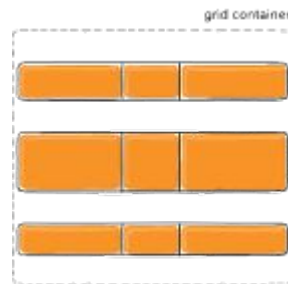
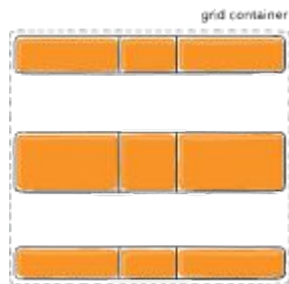
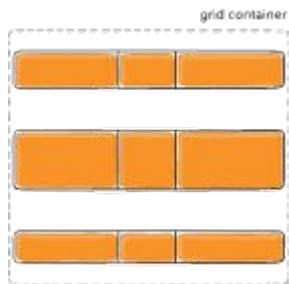
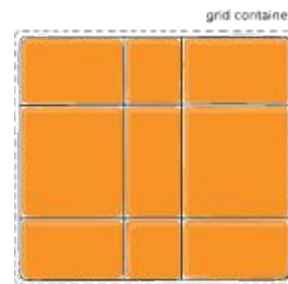
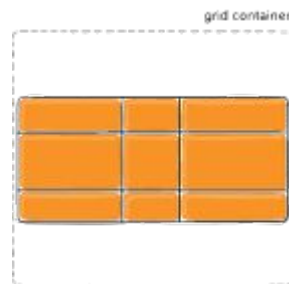
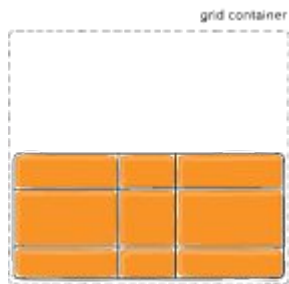
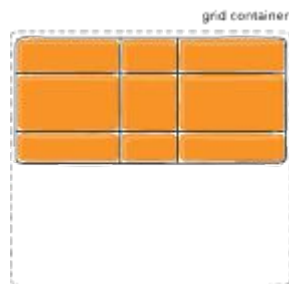
Servirá para **alignar** cada columna (contenido) dentro del contenedor.



align-content

start | end | center | space-between | space-around | space-evenly | stretch;

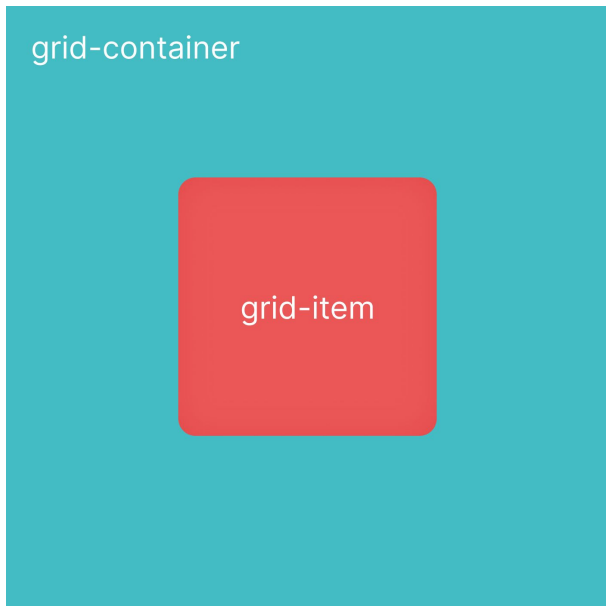
Servirá para **alinear** cada fila (contenido) dentro del contenedor.



place-content

start | end | center | stretch | baseline;

Nos permite definir `justify-content` y `align-content` con la misma propiedad cuando poseen el mismo valor.



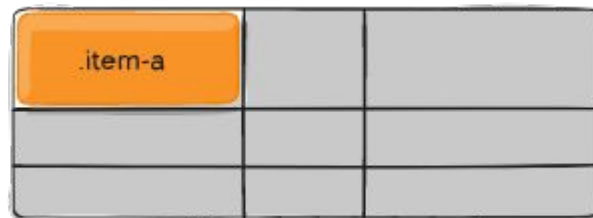
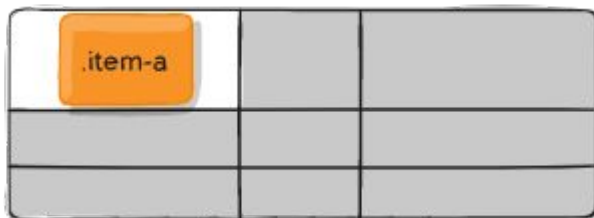
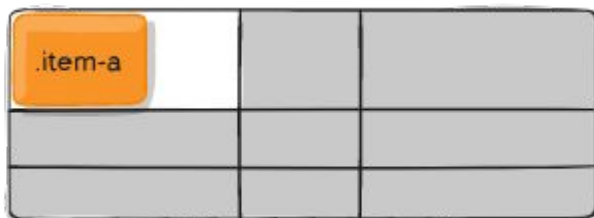
```
.grid-container {  
  display: grid;  
  place-content: center;  
}
```

SELF

justify-self

`start | end | center | stretch;`

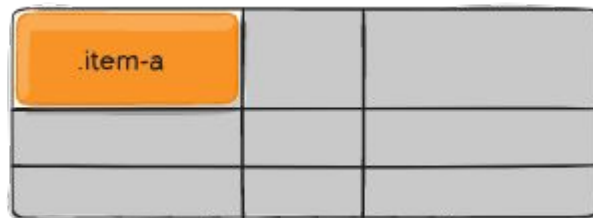
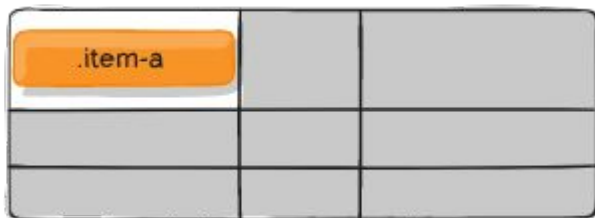
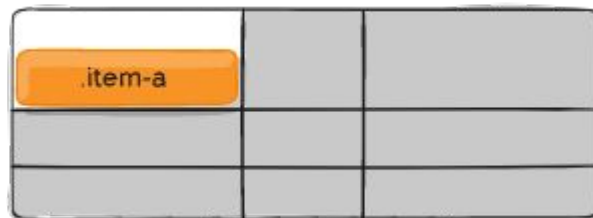
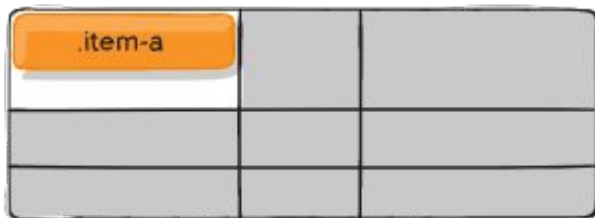
Servirá para **alinear horizontalmente** cada item hijo de forma individual.



align-self

`start | end | center | stretch;`

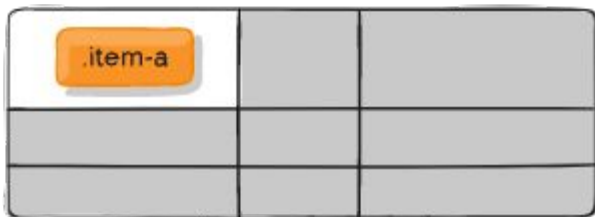
Servirá para **alinear verticalmente** cada item hijo de forma individual.



place-self

`start | end | center | stretch;`

Servirá para **align** ambos ejes con la misma propiedad.



```
.grid-container {  
  display: grid;  
  place-self: center;  
}
```

Fuentes

Las imágenes de esta presentación fueron tomadas de:
<https://css-tricks.com/snippets/css/complete-guide-grid/>