

Programador Web Inicial

GRIDS

¿QUÉ ES?

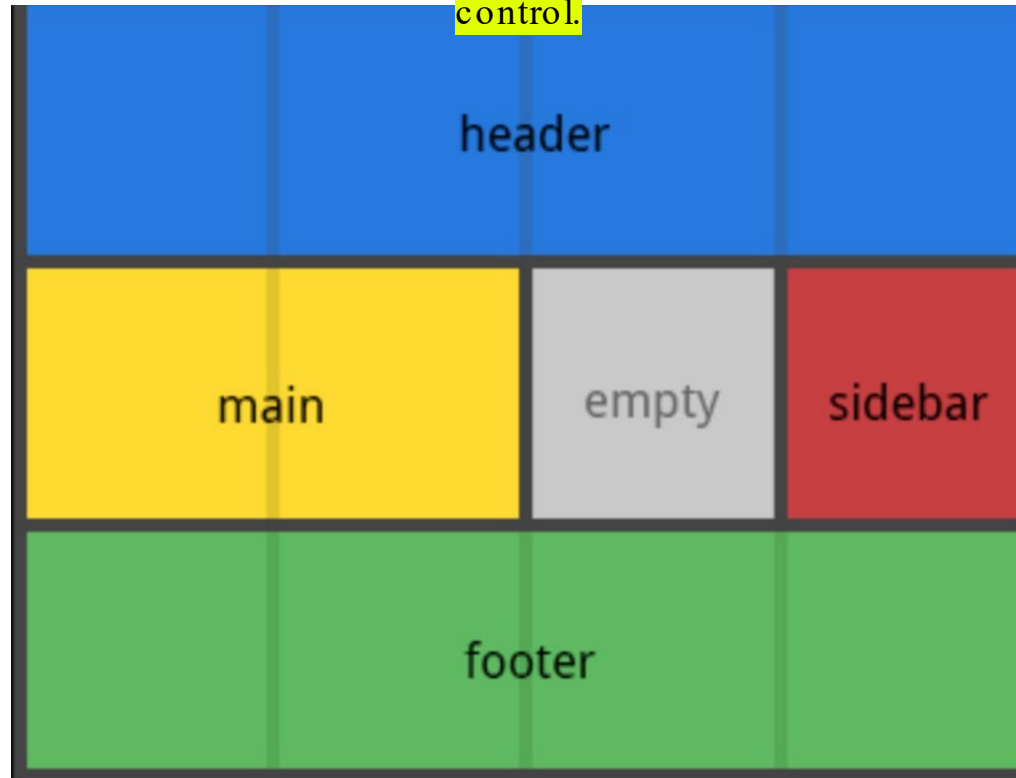
CSS Grid es el sistema de maquetación más potente que hay disponible.

Se trata de un sistema en 2D que permite definir filas y columnas (a diferencia de, por ejemplo, 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.

CSS Grid layout contiene funciones de diseño dirigidas a los desarrolladores de aplicaciones web. 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.

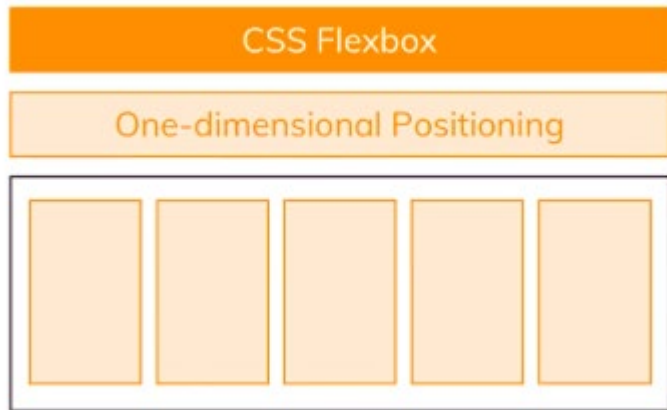


¿POR QUÉ GRIDS?

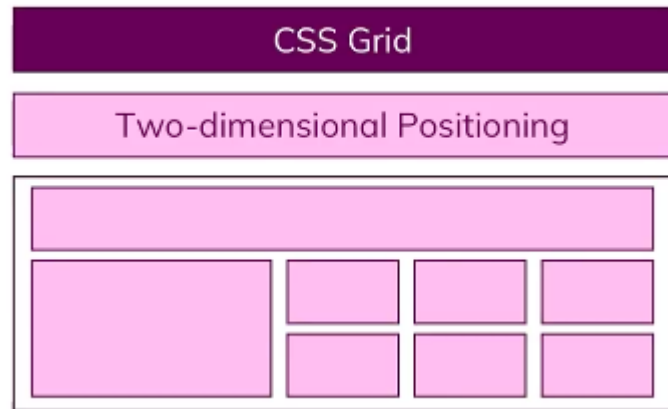
- Grid CSS surge de la necesidad de algo más potente, y toma las ventajas del sistema Flexbox, sumándole muchas mejoras y características que permiten crear muy rápido cuadrículas sencillas y potentes.
- Grid toma la filosofía y la base del sistema Flexbox. Esto no significa que lo reemplaza, sino que pueden convivir.
- Está pensado para estructuras grandes y complejas.

DIFFERENCIA ENTRE FLEXBOX Y GRIDS

Flexbox

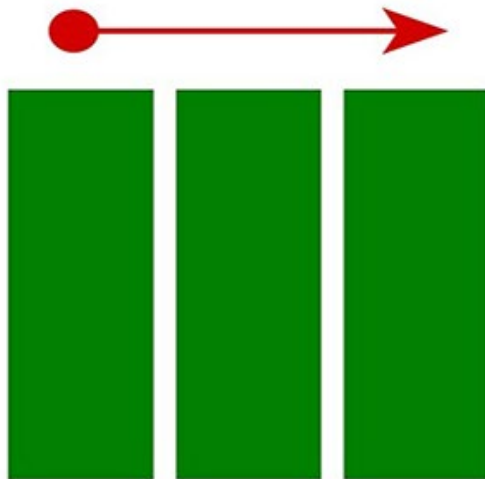


Grids



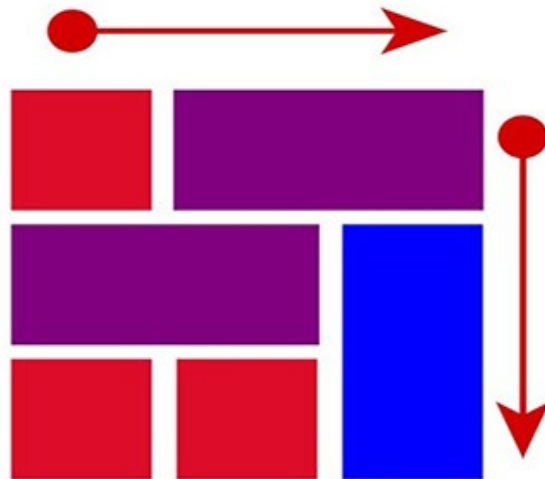
Ambos son mucho más potentes que cualquier técnica que haya existido antes.

DIFERENCIA ENTRE FLEXBOX Y GRIDS



Flexbox

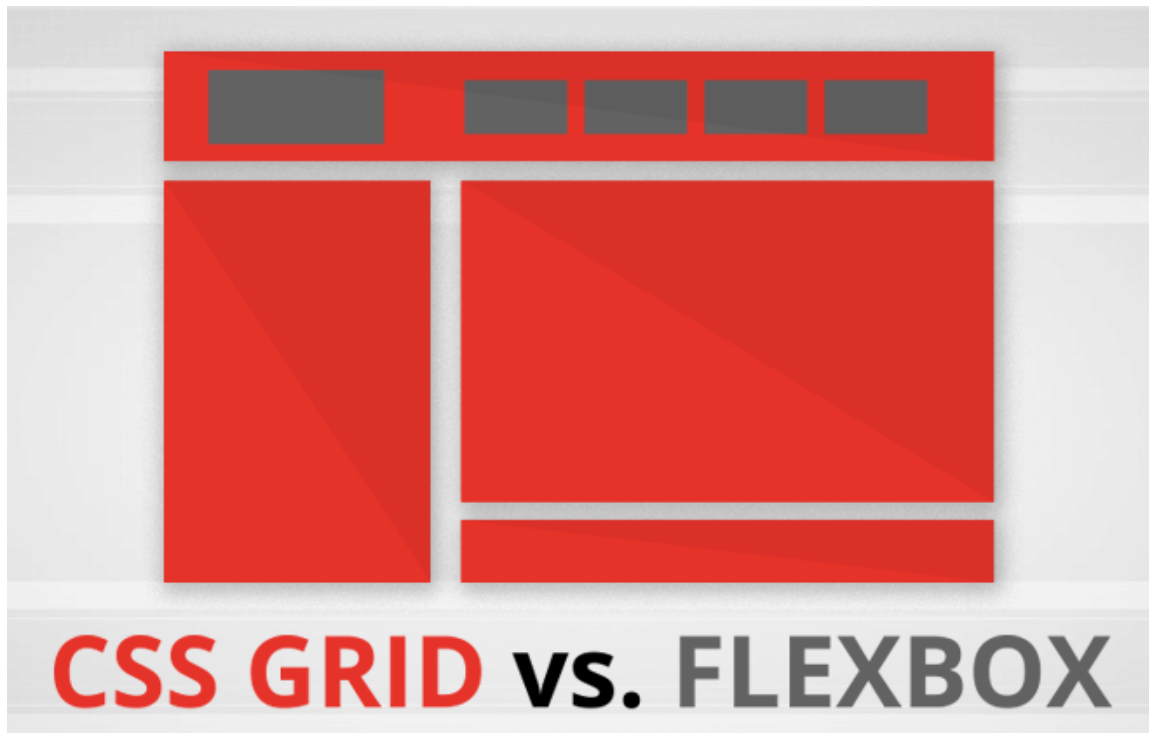
Unidimensional



CSS Grids

Bidimensional

PUEDEN CONVIVIR



PROPIEDADES DEL PADRE

Activamos la cuadrícula Grid utilizando, sobre el elemento contenedor, la propiedad display con el valor grid o inline-grid. Esto influye en cómo se comportará la cuadrícula con el exterior. El primero de ellos permite que la aquella aparezca encima/debajo del contenido exterior (en bloque), mientras que el segundo permite que la cuadrícula se vea a la izquierda/derecha (en línea) del contenido exterior.

Elemento padre

Elemento hijo

```
<section>
```

Elemento padre

```
<div>
```

Elemento hijo

```
</div>
```

```
</section>
```

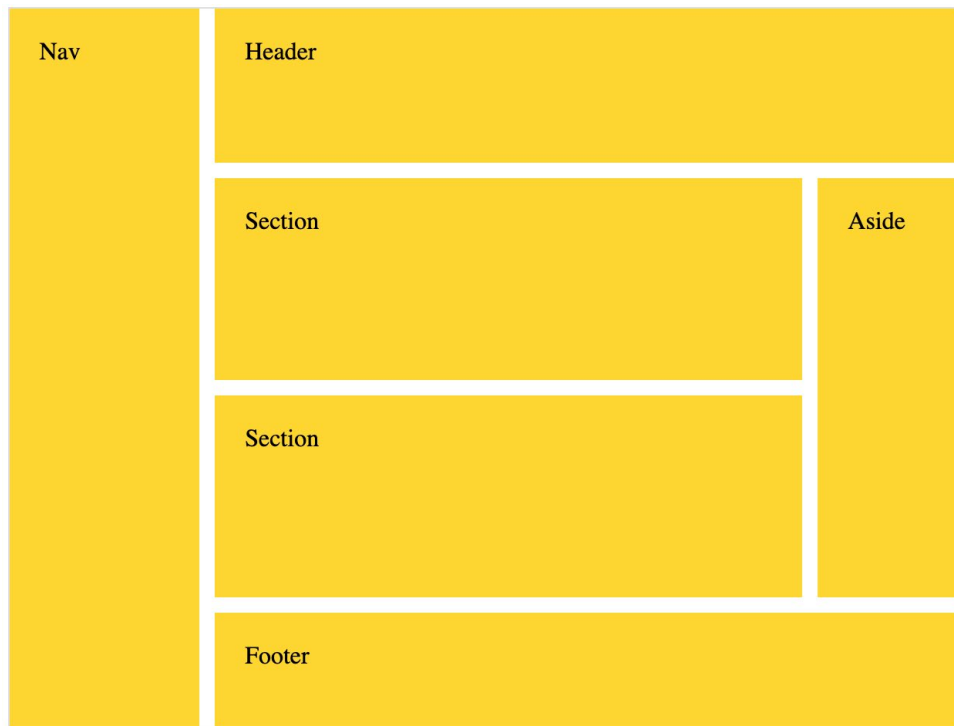

PROPIEDADES DEL PADRE

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 (<i>col 1, col 2...</i>).
grid-template-rows	Establece el TAMAÑO de cada fila (<i>fila 1, fila 2...</i>).
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 (<i>líneas verticales</i>).
grid-row-gap	Establece el TAMAÑO de los huecos entre filas (<i>líneas horizontales</i>).

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
<i>grid-template-columns</i>	Establece el tamaño de las columnas (eje horizontal).
<i>grid-template-rows</i>	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.

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

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

FILAS Y COLUMNAS EXPLÍCITAS

	300px	100px
40px	Item 1	Item 2
100px	Item 3	Item 4

FILAS Y COLUMNAS EXPLÍCITAS

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

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

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

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):**

	2fr	1fr
3fr	Item 1	Item 2
1fr	Item 3	Item 4

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([número de veces][valor o valores])

100

Deberíamos hacer los divs necesarios, pero la grilla está lista para acomodar a sus

[illegible]

section.grid 626 × 240

Así “se ve” la pantalla dividida en grillas.

GRID POR ÁREAS

Ahora veremos otra forma de crear grillas, de una forma más flexible.

Es posible indicar el nombre y la posición concreta de cada área de la cuadrícula. Utiliza la propiedad `grid-template-areas`, donde debes especificar el orden de las áreas. Luego, en cada *ítem hijo*, usas la propiedad `grid-area` para indicar el nombre del área en cuestión.

De esta forma, es muy sencillo crear una cuadrícula altamente personalizada en apenas unas cuantas líneas de CSS.

Propiedad	Descripción
<code>grid-template-areas</code>	Indica la disposición de las áreas en el grid. Cada texto entre comillas simboliza una fila.
<code>grid-area</code>	Indica el nombre del área. Se usa sobre ítems hijos del grid.



GRID POR ÁREAS

Entonces, siguiendo la línea del objetivo que tenemos podemos hacer lo siguiente:

GRID POR ÁREAS

HTML base para todos los ejemplos. Es muy importante marcar la estructura HTML.

```
<div id="grilla">
  <header class="border">Header</header>
  <section id="productos" class="border">Section</section>
  <section id="servicios" class="border">Section</section>
  <nav class="border">Navegacion</nav>
  <aside class="border">Aside</aside>
  <footer class="border">Pie de pagina</footer>
</div>
```

GRID POR ÁREAS

CSS a usar:

```
#grilla {  
  display: grid;  
  grid-template-areas:  
    "nav header header"  
    "nav productos publicidad"  
    "nav servicios publicidad"  
    "nav footer footer";  
  grid-template-rows: 100px 1fr 1fr 75px;  
  grid-template-columns: 20% auto 15%;  
}  
.border {  
  border: 1px solid black;  
}
```

```
header {  
  grid-area: header;  
}  
footer {  
  grid-area: footer;  
}  
section#productos {  
  grid-area: productos;  
}  
section#servicios {  
  grid-area: servicios;  
}  
nav {  
  grid-area: nav;  
}  
aside {  
  grid-area: publicidad;  
}
```

GRID POR ÁREAS

De esta forma, se aproxima a lo que queremos inicialmente, sólo nos falta darle unas decoraciones:



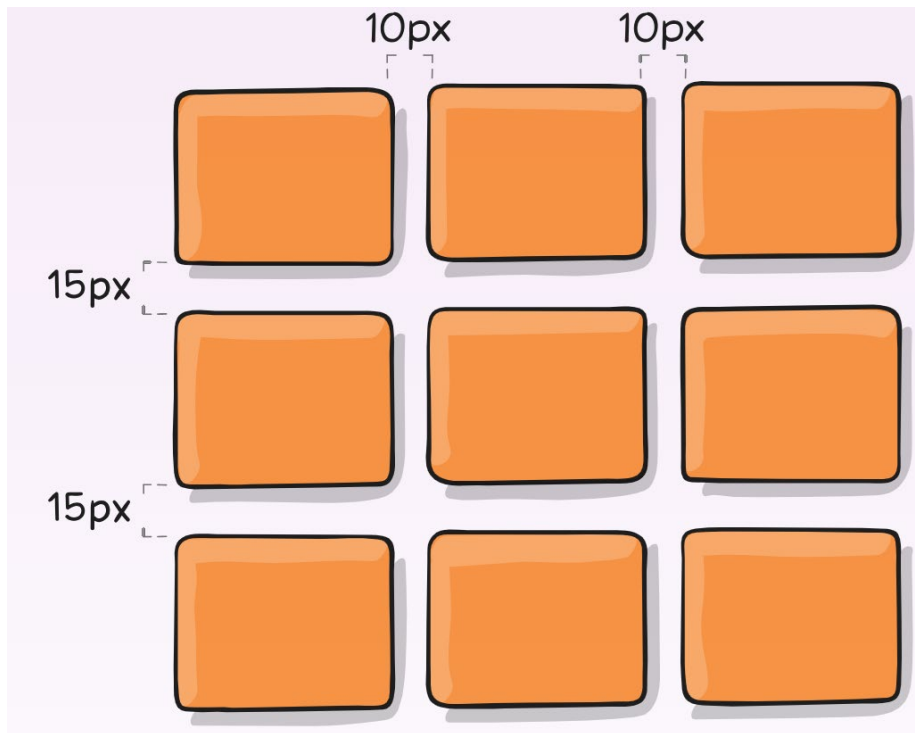
Navegacion	Header	
	Section	Aside
	Section	
	Pie de pagina	

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





En la propiedad **grid-template-areas** también es posible indicar una **palabra clave especial**:

- La palabra clave **none**: Indica que no se colocará ninguna celda en esta posición.
- Uno o más puntos (.): Indica que se colocará una celda vacía en esta posición.

GRID POR ÁREAS

De esta forma, se aproxima a lo que queremos... sólo nos falta darle unas decoraciones:



GRID POR ÁREAS

Llevemos a práctica , agregando más estilos al padre (el contenedor) y verificando que tiene las propiedades `grid-row-gap` y `grid-column-gap`, para hacer separaciones entre las columnas o filas, según el caso.

```
.border {  
  border: 1px solid black;  
  background-color: yellow;  
}
```

```
#grilla {  
  display: grid;  
  grid-template-areas:  
    "nav header header"  
    "nav productos publicidad"  
    "nav servicios publicidad"  
    "nav footer footer";  
  grid-template-rows: 100px 1fr 1fr 75px;  
  grid-template-columns: 20% auto 15%;  
  grid-row-gap: 10px;  
  grid-column-gap: 10px;  
  height: 100vh;  
  margin: 0;  
}
```

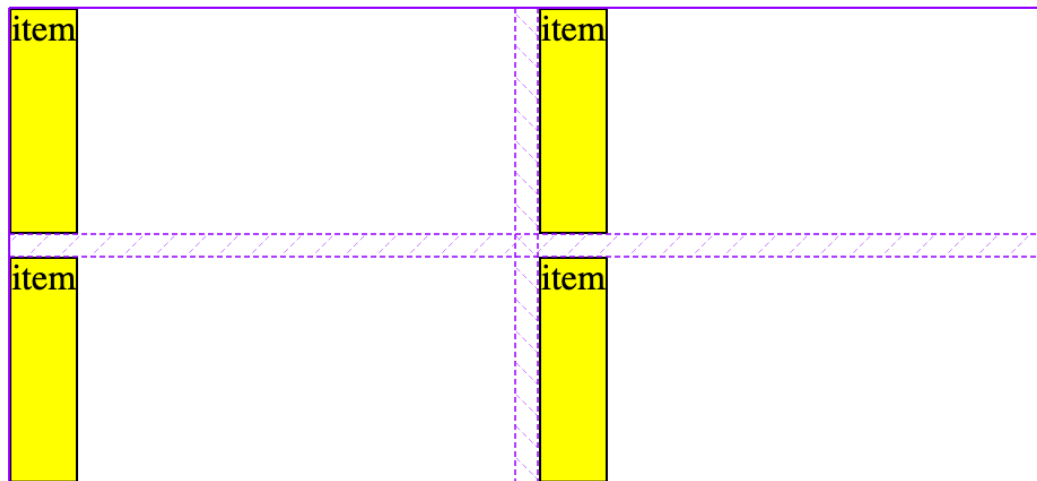
POSICIÓN DE HIJOS (DESDE EL PADRE)

Existen propiedades que se pueden utilizar para colocar los ítems dentro de la cuadrícula. Es posible distribuir los elementos de una forma muy sencilla y cómoda: *justify-items* y *align-items*, que ya conocemos del módulo CSS Flexbox:

Propiedad	Valores	Descripción
<i>justify-items</i>	start end center stretch	Distribuye los elementos en el eje horizontal.
<i>align-items</i>	start end center stretch	Distribuye los elementos en el eje vertical.

JUSTIFY-ITEMS Y ALIGN-ITEMS

La grilla está, pero las celdas “se achican”, se ajusta. Estas propiedades trabajan sobre la celda:



JUSTIFY-ITEMS Y ALIGN-ITEMS

HTML a usar:

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

JUSTIFY-ITEMS Y ALIGN-ITEMS

CSS a usar:

```
#padre {  
  justify-items: stretch; /* start | end |  
center | stretch */  
  align-items: stretch; /* start | end |  
center | stretch */  
  display: grid;  
  width: 95%;  
  grid-template-columns: auto auto;  
  grid-column-gap: 10px;  
  grid-template-rows: 100px 100px;  
  grid-row-gap: 10px;  
}
```

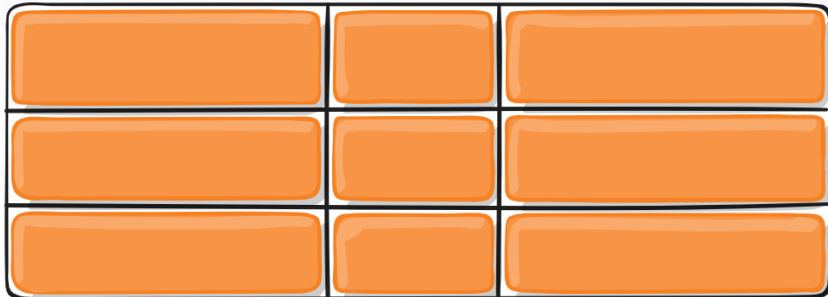
```
#padre div {  
  border: solid 1px;  
  font-size: 21px;  
  padding: 5px;  
  background-color: yellow;  
}
```



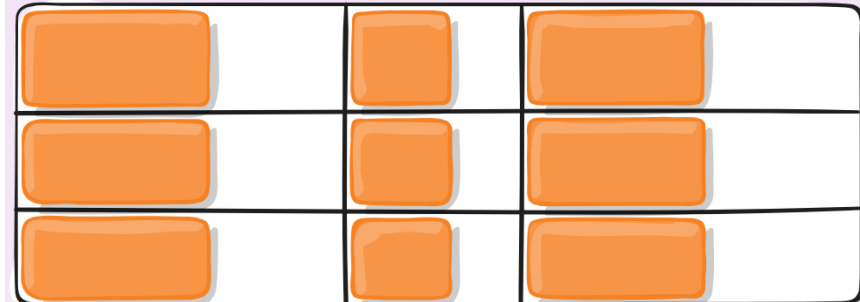
```
.padre {  
  justify-items: ....;  
}
```

Alineando el contenido dentro de las celdas, de forma horizontal.










JUSTIFY-ITEMS: STRETCH












JUSTIFY-ITEMS: START



JUSTIFY-ITEMS: END

JUSTIFY-ITEMS: CENTER

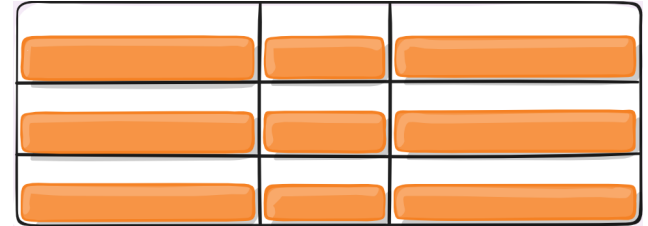
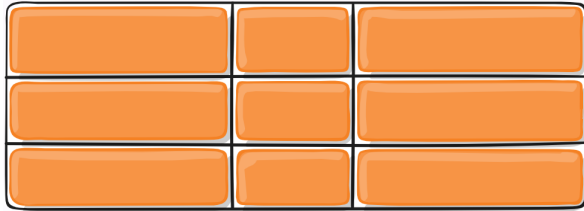
		
		
		

```
.padre {
    align-items: stretch;

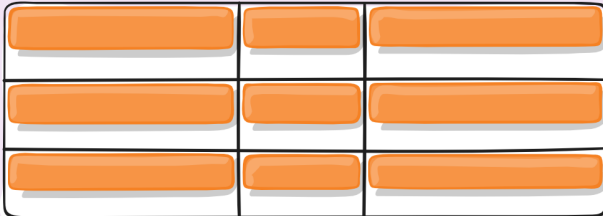
    /* predeterminado */
}
```

Alineando el contenido dentro de las celdas, de forma vertical.

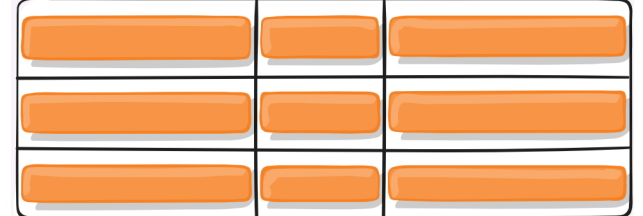
ALIGN-ITEMS: STRETCH



ALIGN-ITEMS: START



ALIGN-ITEMS: CENTER



POSICIÓN DE ELEMENTOS

Es posible utilizar las propiedades *justify-content* o *align-content* para cambiar la distribución de todo el contenido en su conjunto. Puedes hacer pruebas en [este enlace](#).

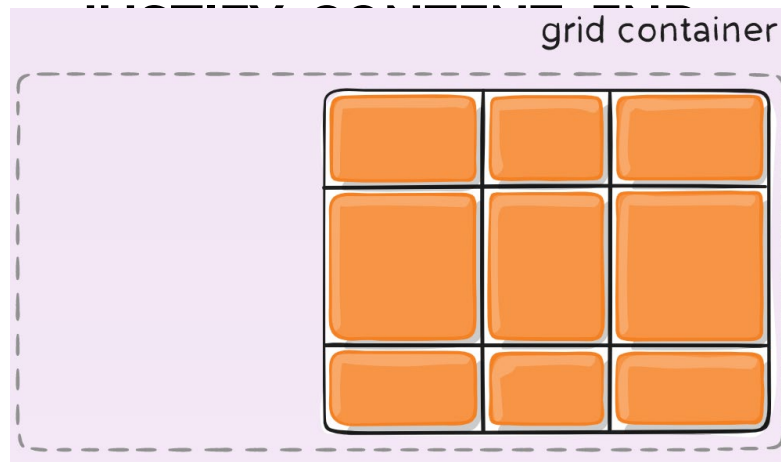
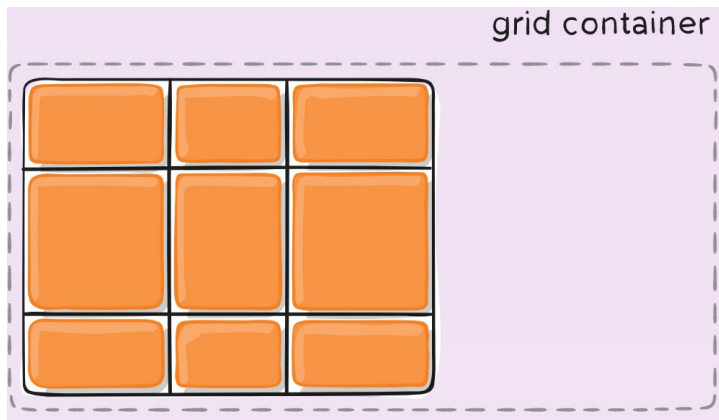
Propiedad	Valores	Afecta a
<i>justify-content</i>	start end center stretch space-around space-between space-evenly	Eje horizontal
<i>align-content</i>	start end center stretch space-around space-between space-evenly	Eje vertical

```
.padre {  
  justify-content: start;  
}
```

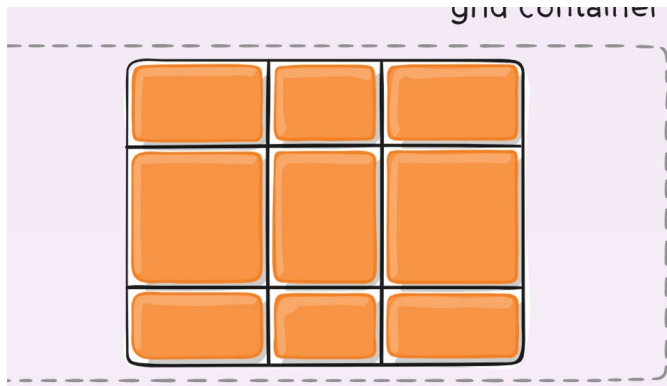
Alineando **todo** el conjunto de celdas, de forma **horizontal**.

Dentro de su “*padre*” (es requisito que tenga **ancho**).

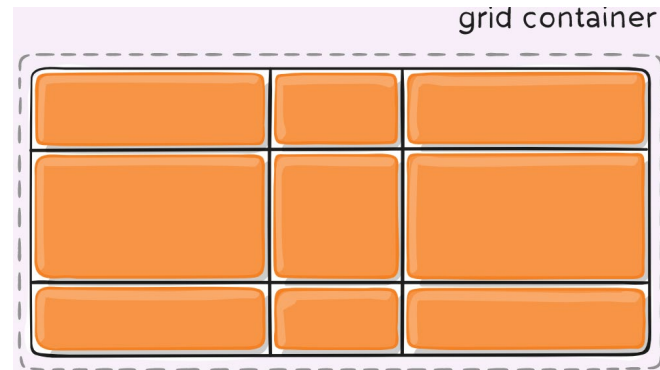
JUSTIFY-CONTENT: START



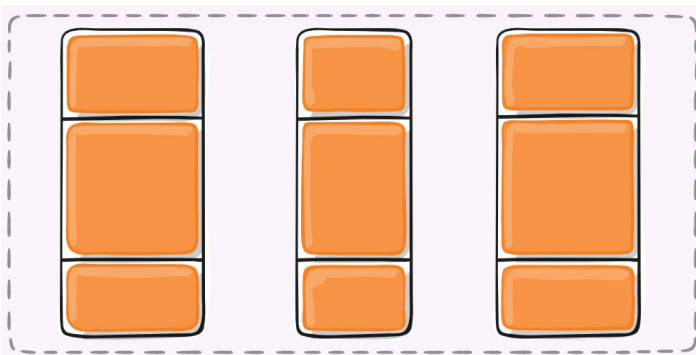
JUSTIFY-CONTENT: CENTER



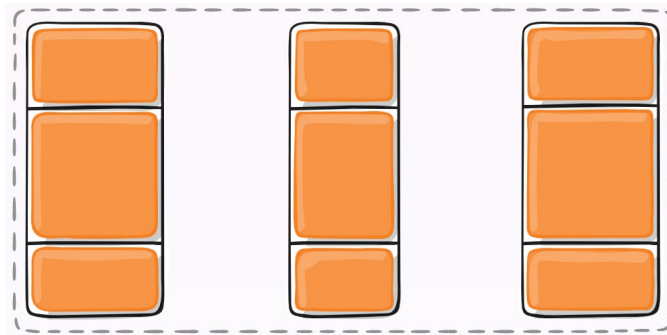
JUSTIFY-CONTENT:



JUSTIFY-CONTENT: SPACE-

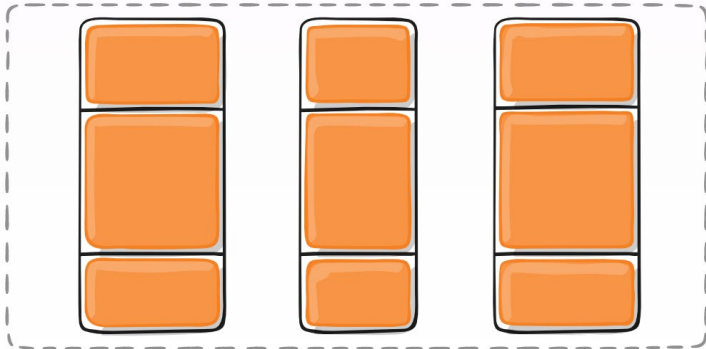


JUSTIFY-CONTENT:



```
.padre {  
  justify-content: space-evenly;  
}
```

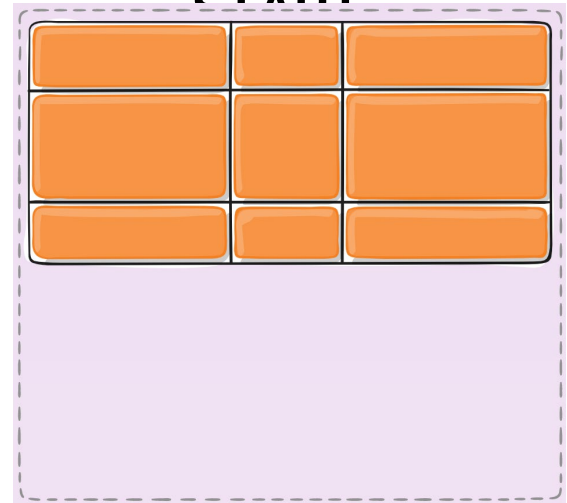
JUSTIFY-CONTENT: SPACE-EVENLY



Alineando **todo** el conjunto de celdas, de forma **horizontal**.

Dentro de su “*padre*” (es requisito que tenga **ancho**).

ALIGN-CONTENT: START

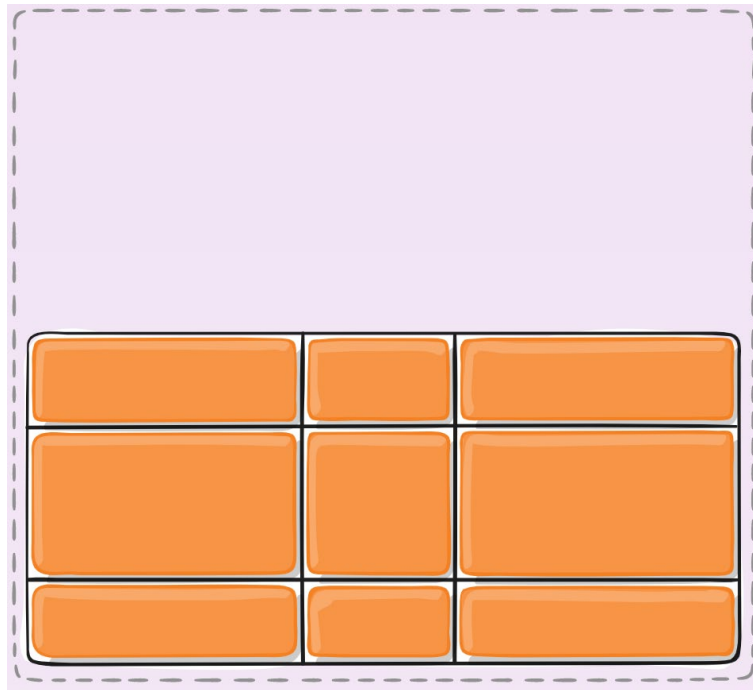


ALIGN-CONTENT: END

Alineando **todo** el conjunto de celdas, de forma **vertical**.

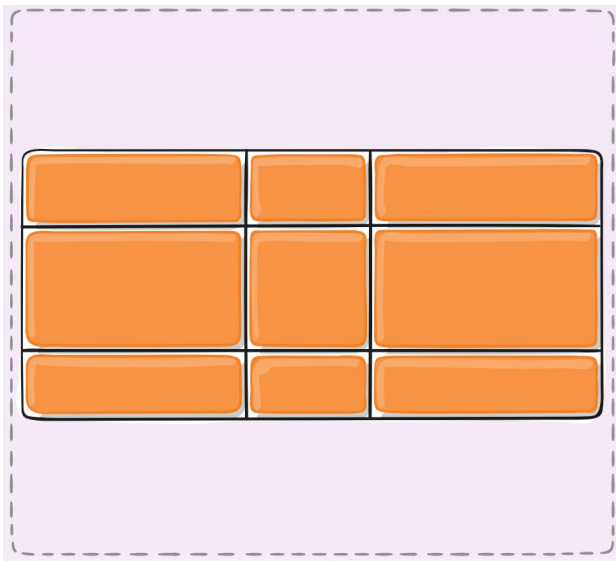
Dentro de su “*padre*” (es requisito que tenga **altura**).

```
.padre {  
  align-content: end;  
}
```




```
.padre {  
  align-content: center;  
}
```

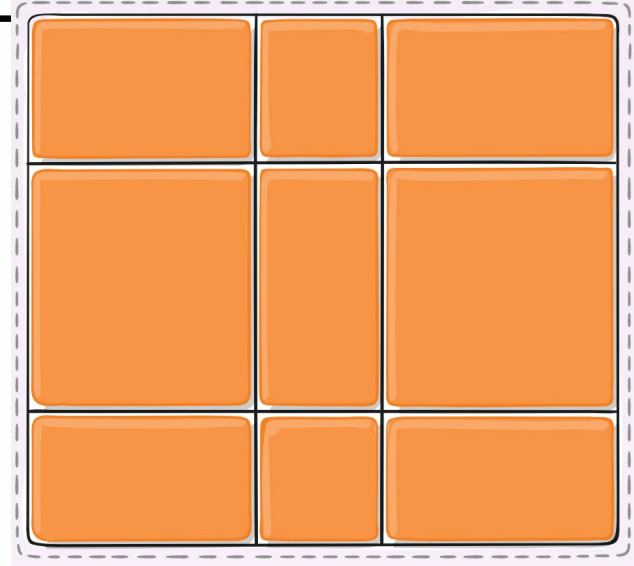
ALIGN-CONTENT: CENTER



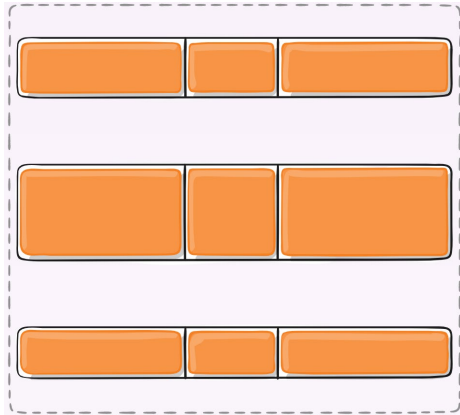
Alineando **todo** el conjunto de celdas, de forma **vertical**.

Dentro de su “*padre*” (es requisito que tenga **altura**).

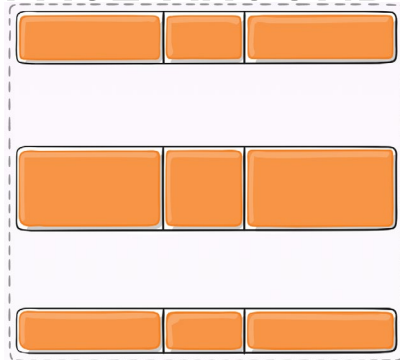
ALIGN-CONTENT: STRETCH



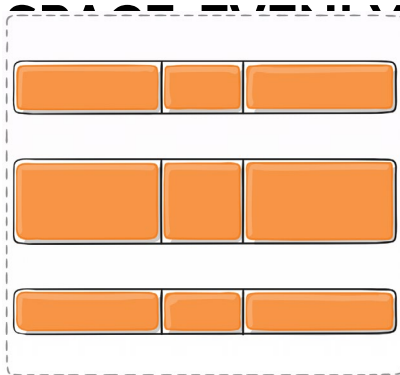
ALIGN-CONTENT: SPACE-AROUND



ALIGN-CONTENT: SPACE-AROUND



ALIGN-CONTENT:



ITEMS PROPIEDADES

Hasta ahora hemos visto propiedades CSS que se aplican solamente al contenedor padre de una cuadrícula. Ahora vamos a ver ciertas propiedades que se aplican a cada ítem hijo de la cuadrícula, para alterar o cambiar el comportamiento específico de dicho elemento.

Prueba sus propiedades [aquí](#).

Propiedad	Descripción
<i>justify-self</i>	Altera la justificación del ítem hijo en el eje horizontal .
<i>align-self</i>	Altera la alineación del ítem hijo en el eje vertical .
<i>grid-area</i>	Indica un nombre al área especificada, para su utilización con <i>grid-template-areas</i>.

Alinea específicamente a la celda (item, hijo) que necesites, de forma horizontal:

```
.hijo {  
  justify-self: start;  
}
```

JUSTIFY-Self: START

item-a		

JUSTIFY-Self: END

	item-a	

JUSTIFY-Self: CENTER

.item-a		

JUSTIFY-Self: STRETCH

.item-a		

```
.hijo {  
  align-self: start;  
}
```

ALIGN-SELF: START

.item-a		

Alinea específicamente a la celda (item, hijo) que necesites, de forma vertical:

.item-a		

ALIGN-SELF: CENTER

.item-a		

ALIGN-SELF: STRETCH

.item-a		