Esta clase va a ser

- grabada

Clase 06. DESARROLLO WEB

Grids

Temario

05

Flexbox

- ✓ Flexbox
- ✓ Propiedad de padres e hijos

06

Grids

- ✓ Grids
- ✓ Grids y Flexbox
- ✓ Grids por áreas

07

Grids II

- ✓ Grids Mobile First
- ✓ Media queries
- ✓ Grids+Flex+@Media

Objetivos de la clase

- **Conocer** Grids.
- **Comparar** el alcance de Grids y Flexbox
- **Aplicar** las nuevas formas de modelar con Grids.

CLASE N°05

Glosario

Flexbox: es un modo de diseño que nos permite crear estructuras para sitios web de una forma más fácil. No se trata de una propiedad de CSS, sino de un conjunto de ellas. Se basa sobre un contenedor (padre) para ordenar a sus ítems (hijos).

- ✓ **Row (flex-direction):** esta propiedad nos va a permitir especificar si queremos que los flex items se dispongan en filas o columnas.
- ✓ **Row-reverse (flex-direction):** con el valor row-reverse (fila inversa) los flex items se apilan en una fila de derecha a izquierda.
- ✓ **Column (flex-direction):** con este valor, los flex items se apilan en una columna de arriba hacia abajo.
- ✓ **Column-reverse:** con este valor, los flex items se apilan en una columna de abajo hacia arriba.
- ✓ **Flex-wrap:** permite especificar si queremos que los ítems puedan saltar a una nueva línea, cuando el contenedor flexible se quede sin espacio.
- ✓ **Wrap (flex-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.
- ✓ **Wrap-reverse (flex-wrap):** esta vez el orden es de izquierda a derecha, y de abajo a arriba.

CLASE N°05

Glosario

- ✓ **Flex-flow:** Es la forma abreviada (shorthand) o rápida para las propiedades: flex-direction y flex-wrap. Se pone primero la propiedad de flex-direction, y luego la de flex-wrap.
- ✓ **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.
- ✓ **Flex-start (justify-content):** consiste en alinear los flex items (hijos) al lado izquierdo.
- ✓ **Flex-end (justify-content):** consiste en alinear los flex items (hijos) al lado derecho.
- ✓ **Center (justify-content):** consiste en alinear los flex items (hijos) al centro.

CLASE N°05

Glosario

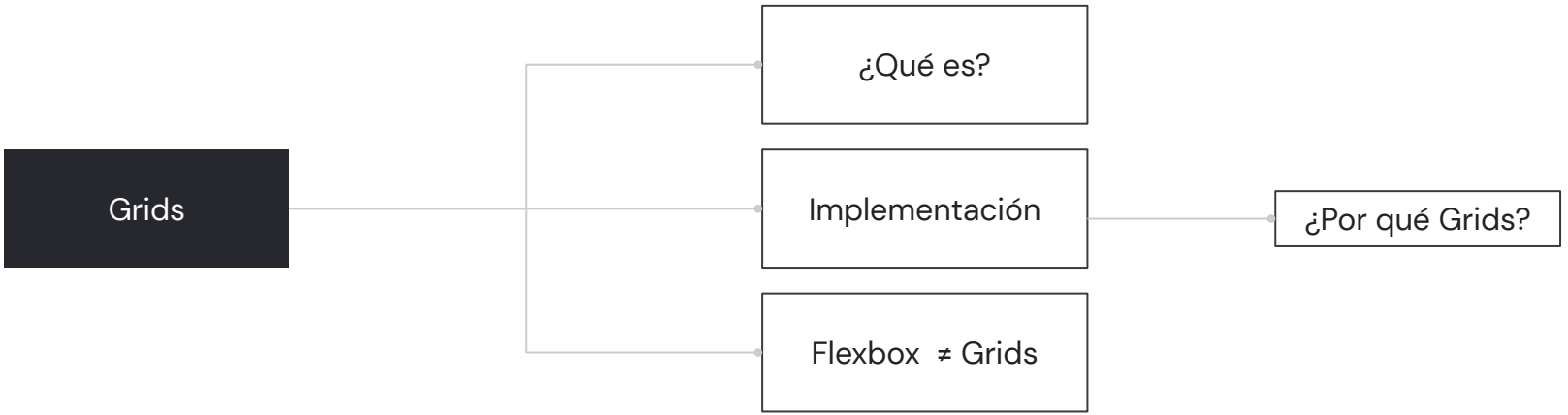
- **Space-between (justify-content):** es hacer que los flex items (hijos) tomen la misma distancia o espaciado entre ellos dentro del contenedor flexible, quedando el primer y último elemento pegados con los bordes del contenedor en el eje principal.
- **Space-around (justify-content):** muestra los flex items (hijos) con el mismo espacio de separación entre sí. El espaciado entre los bordes lo toman del contenedor padre.
- **Space-evenly:** hace que el espacio entre los flex items (hijos) sea igual. No es lo mismo que space-around.
- **Stretch (align-items):** tratará de llenar toda la altura (o anchura) del contenedor, siempre y cuando los hijos no tengan propiedades de dimensión definidas.
- **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.

CLASE N°05

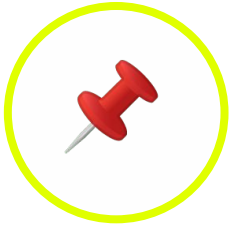
Glosario

- ✓ **Order:** esta propiedad permite modificar el orden de aparición de un elemento. Recibe como valor números enteros.
- ✓ **Flex-basis:** define el ancho de un elemento inicial. Este valor por defecto viene configurado en “auto”. Actúa como “máximo” o “mínimo”, al usar flex-grow o flex-shrink.
- ✓ **Flex-grow:** esta propiedad define la capacidad de un elemento de crecer, cuando en el contenedor todavía hay espacio sobrante.
- ✓ **Flex-shrink:** básicamente es lo mismo que flex-grow, pero con el espacio faltante.

MAPA DE CONCEPTOS



Grids

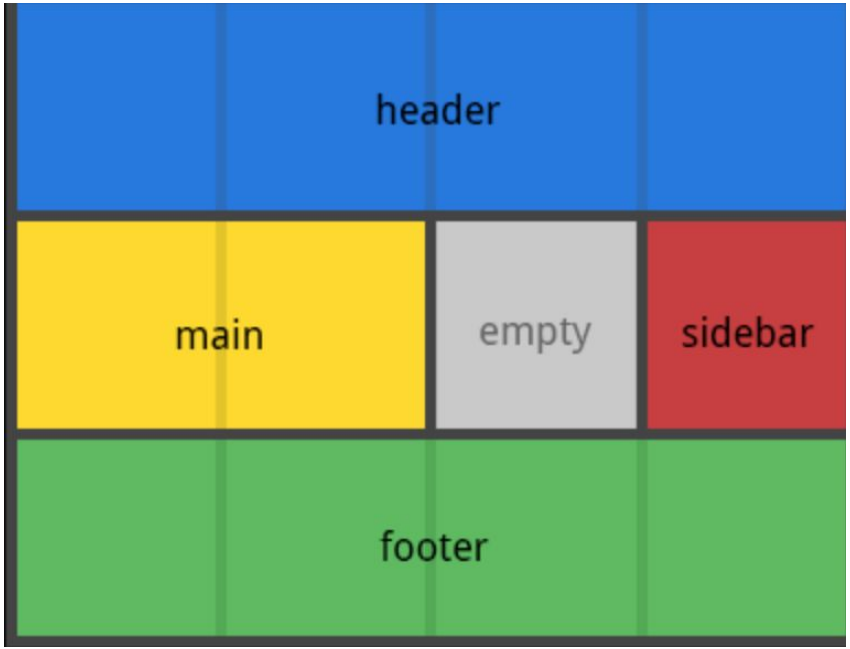


¿Qué es Grids?

👉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

¿Position? ¿Float? ¿Elementos de Bloque? ¿Elementos de líneas?
¿Es suficiente crear un layout/estructuras para páginas web actuales?

Y... ¿Flexbox?

¿Necesitaremos algo más potente para estructuras web?

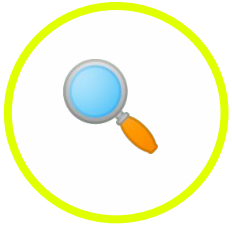
Implementar Grids

¡SI!

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.



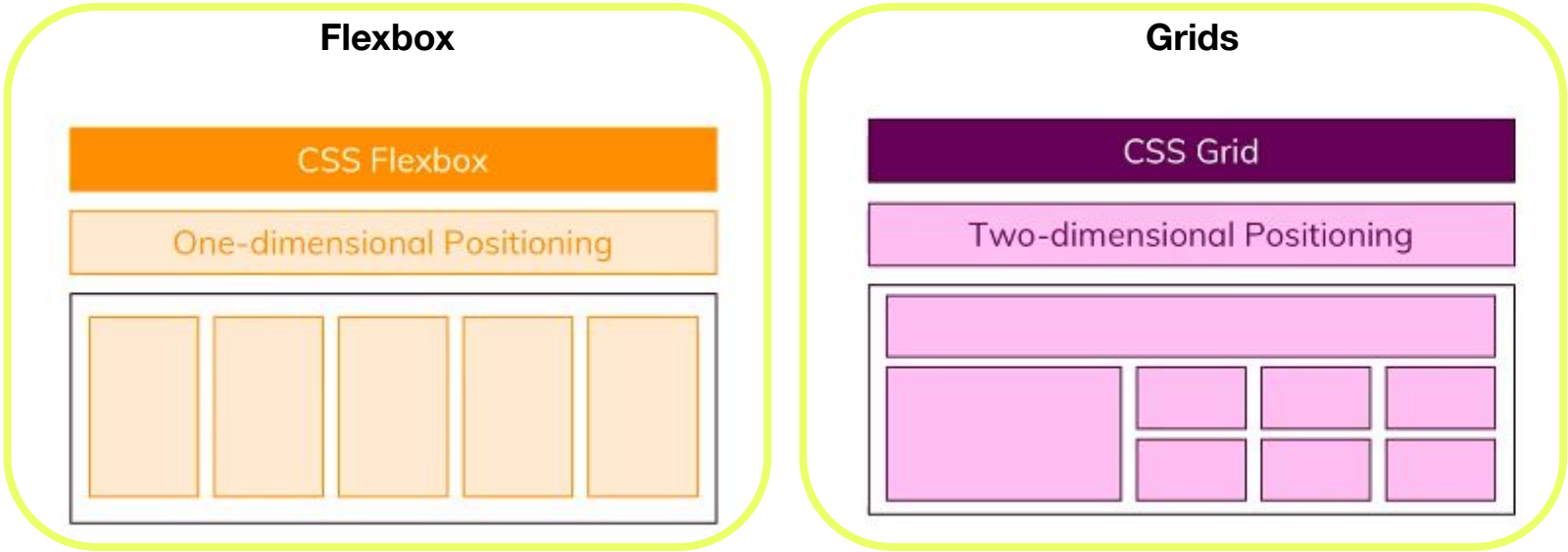
¿Por qué Grids?

👉 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.

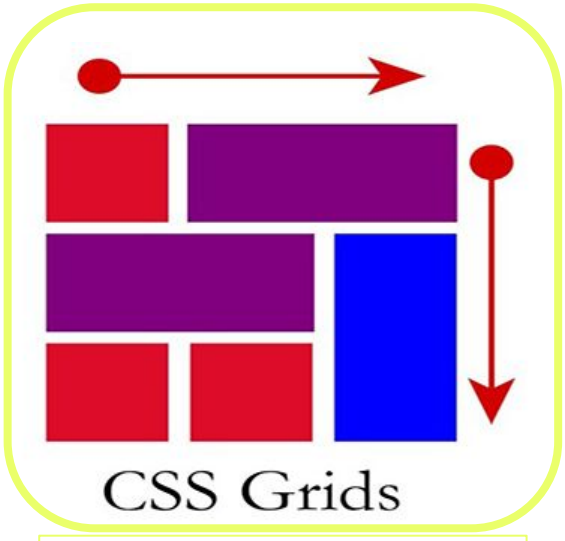
Diferencia entre Flexbox y Grids



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

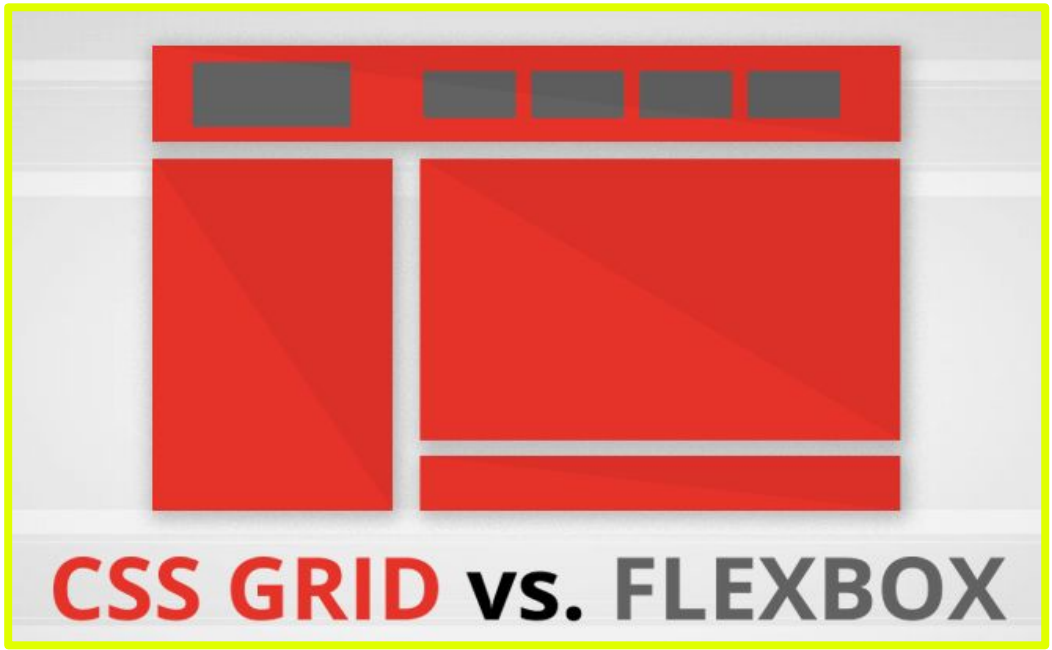
Diferencia entre Flexbox y Grids

UNIDIMENSIONAL



BIDIMENSIONAL

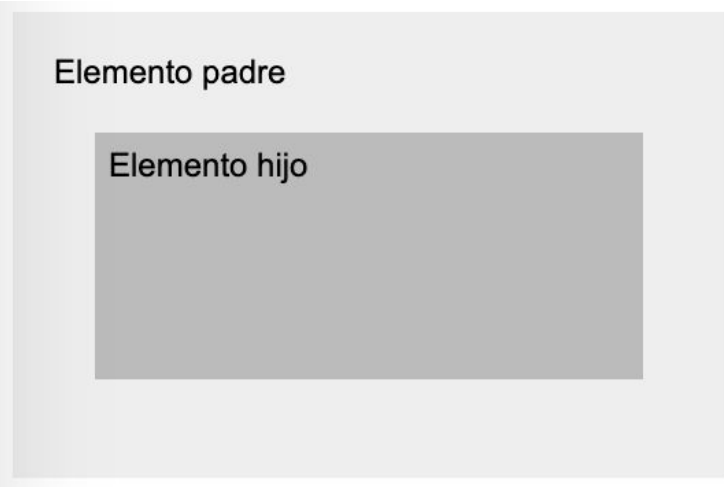
Pueden convivir



CODERHOUSE

Propiedades del padre

Activamos la cuadrícula Grid utilizando, sobre el elemento contenedor, la propiedad display con el valor grid o inline-grid. El primero de ellos permite que 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.




```
<section>
 Elemento padre
 <div>
   Elemento hijo
 </div>
</section>
```

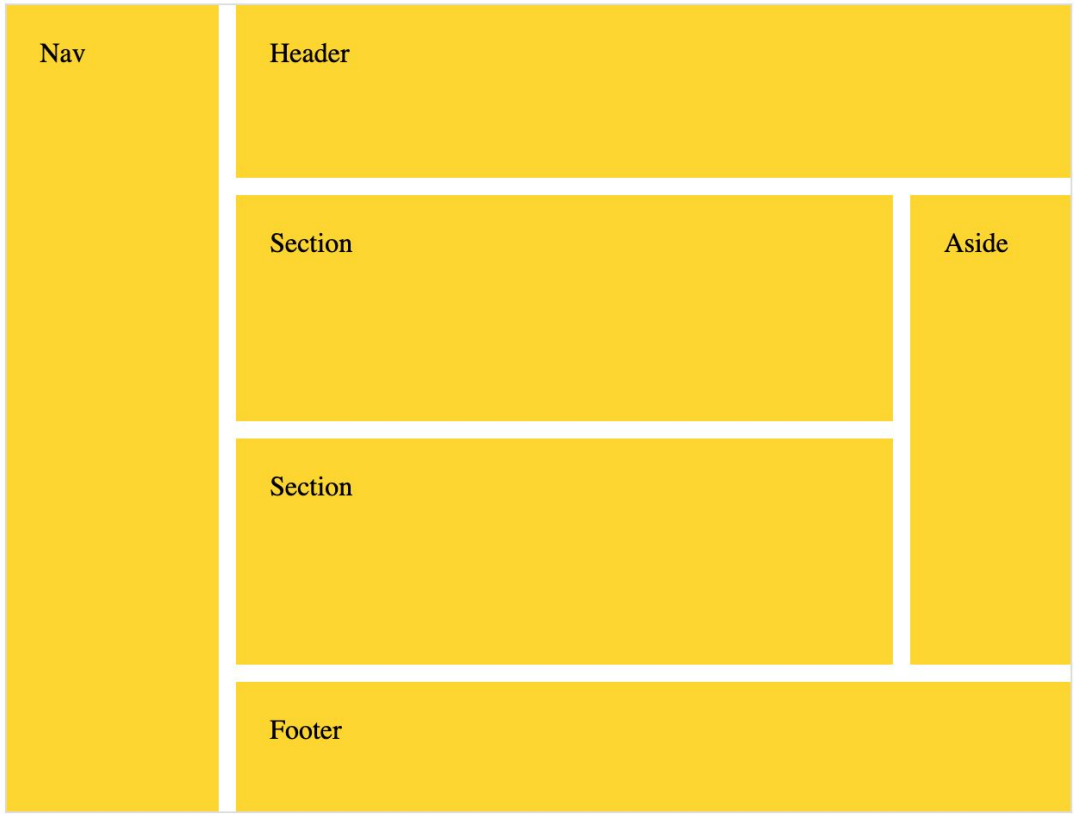
Propiedades del padre

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 (*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).

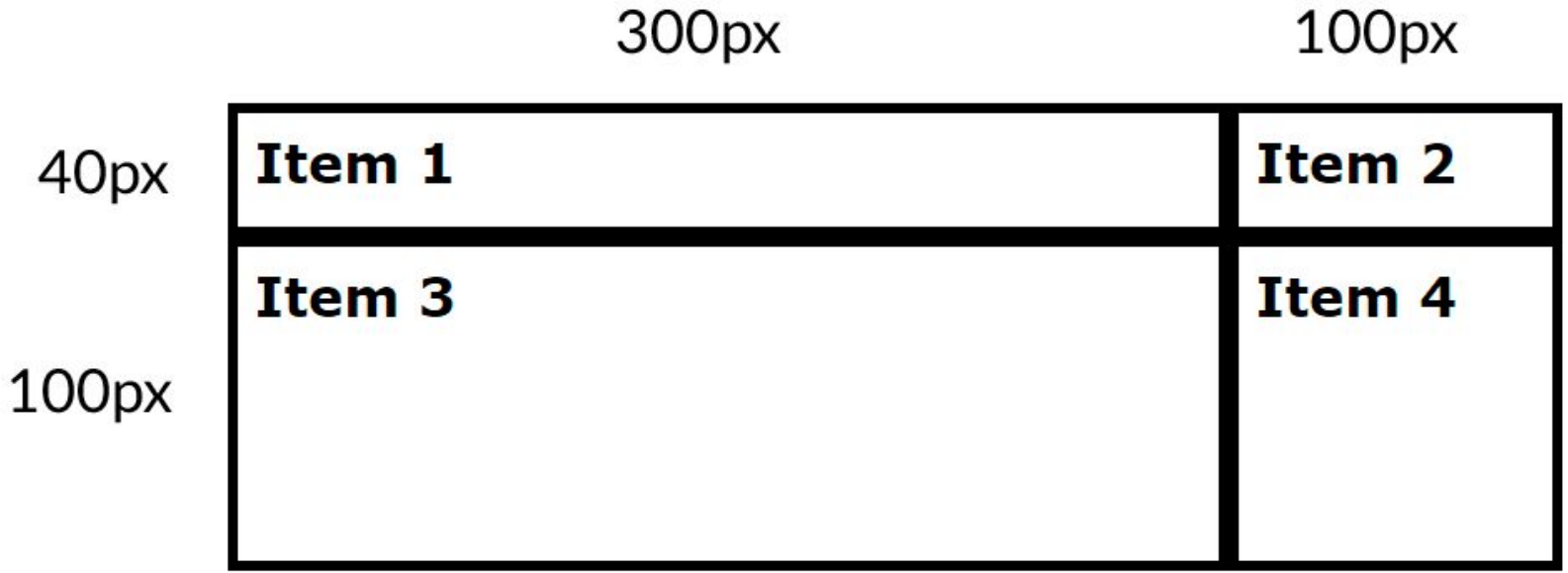
Filas y columnas explícitas

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

```
.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



Filas y columnas explícitas

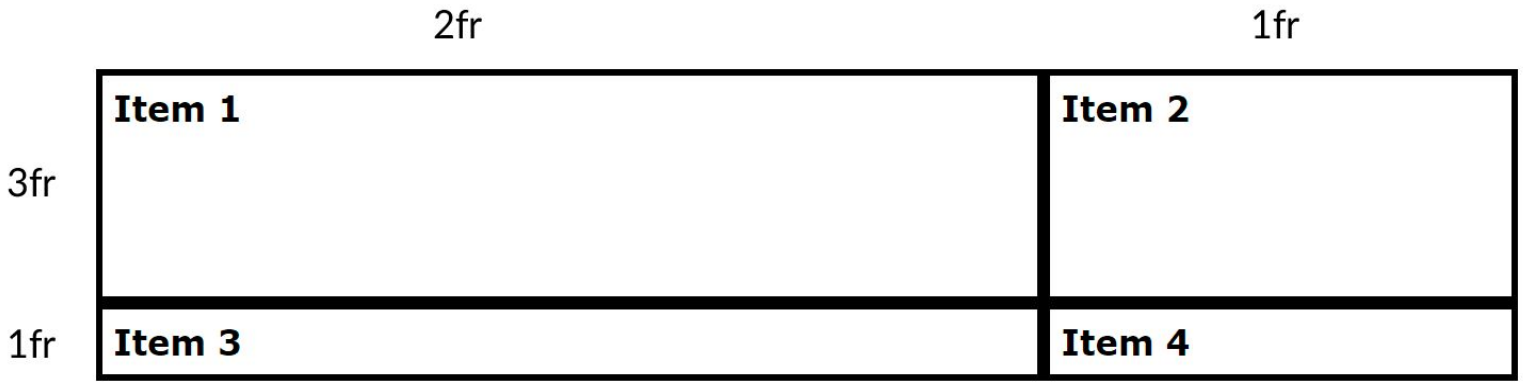
Unidad creada para ser usada en grid (fr (fraction))

```
.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 explícitas

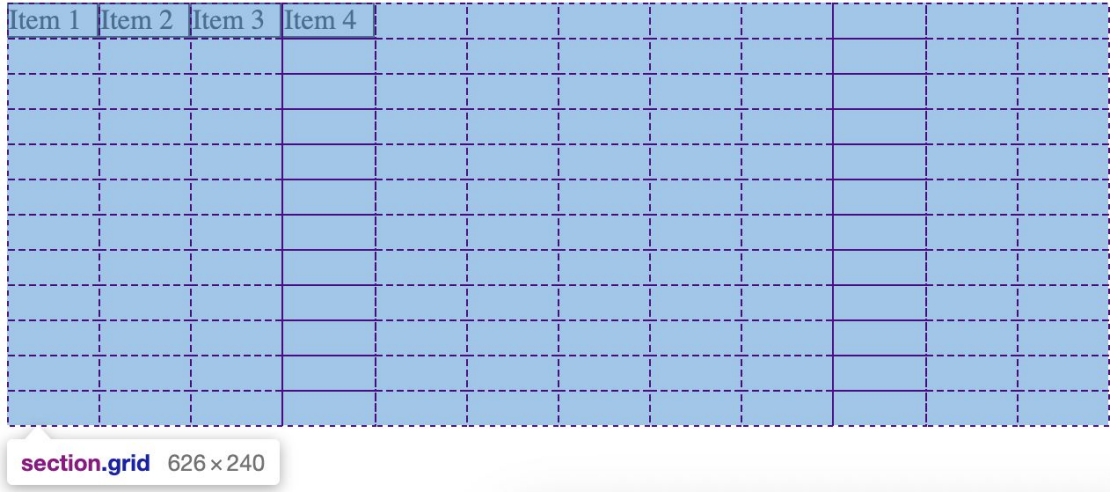
Si necesitas hacer muchas columnas y filas iguales, puedes usar lo siguiente:

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

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

Filas y columnas explícitas

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



👆 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.**

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

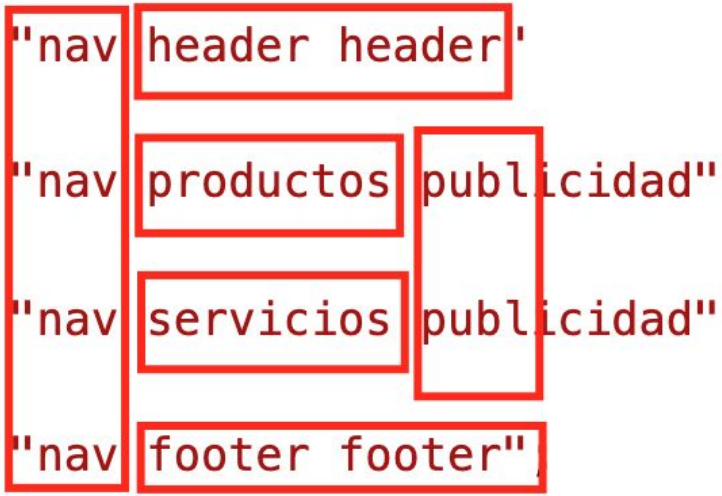
CSS a usar:

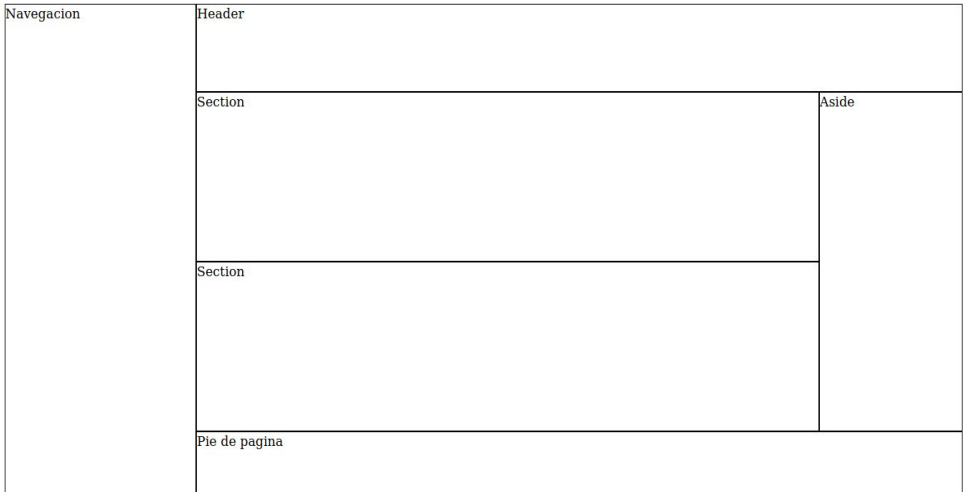
```
#grilla {
   display: grid;
   grid-template-areas:
     "nav header header"
     "nav productos publicidad"
     "nav servicios publicidad"
     "nav footer footer";
   grid-template-rows: 100px 1fr 1fr 75px;
   grid-template-columns: 20% auto 15%;
}
.border {
   border: 1px solid black;
}
```

```
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:





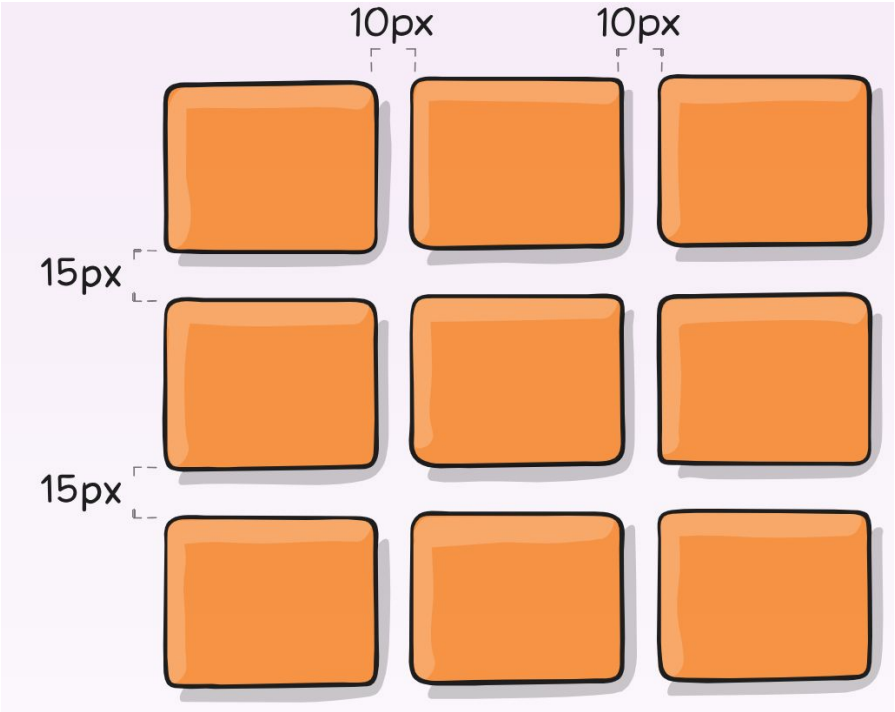
CODERHOUSE

Grid por áreas

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

Grid espacios



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

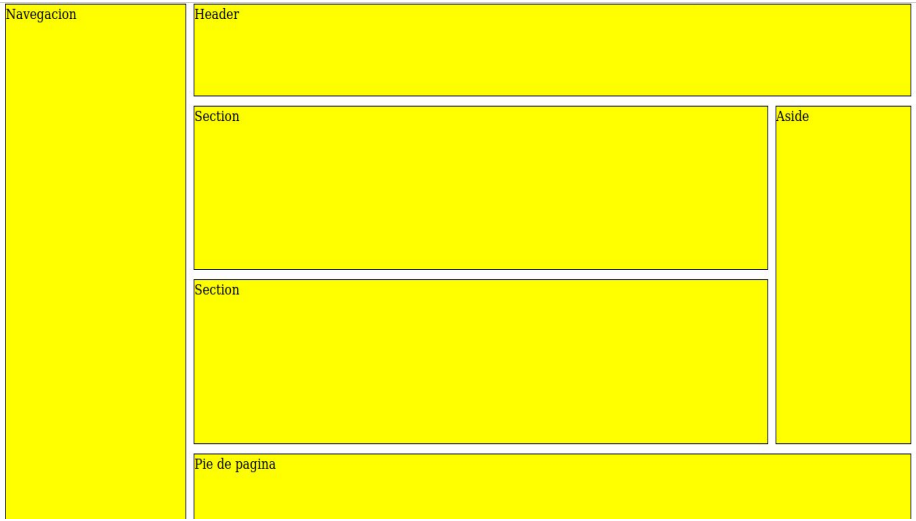
👉 La palabra clave **none**: Indica que no se colocará ninguna celda en esta posición.

👉 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:

```
"nav header header"
"nav productos publicidad"
"nav servicios publicidad"
"nav footer footer";
```



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
justify-items	start | end | center | stretch	Distribuye los elementos en el eje horizontal.
align-items	start | end | center | stretch	Distribuye los elementos en el eje vertical.

Justify – Items y Align – Items

HTML a usar:

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

Justify – Items y Align – Items

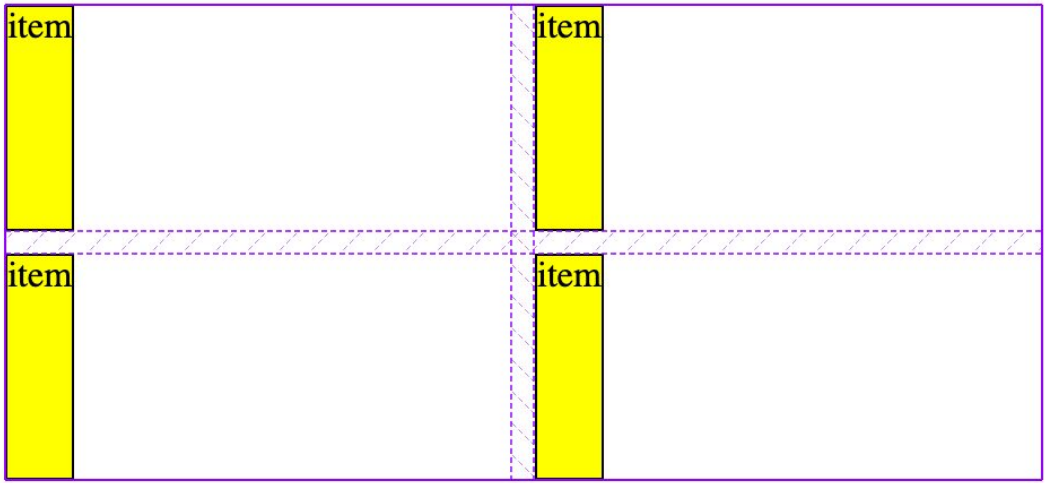
CSS a usar:

```
#padre {
  display: grid;
  justify-items: stretch; /* start | end |
center | stretch */
  align-items: stretch; /* start | end |
center | stretch */
  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;
}
```

Justify - Items y Align - Items

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

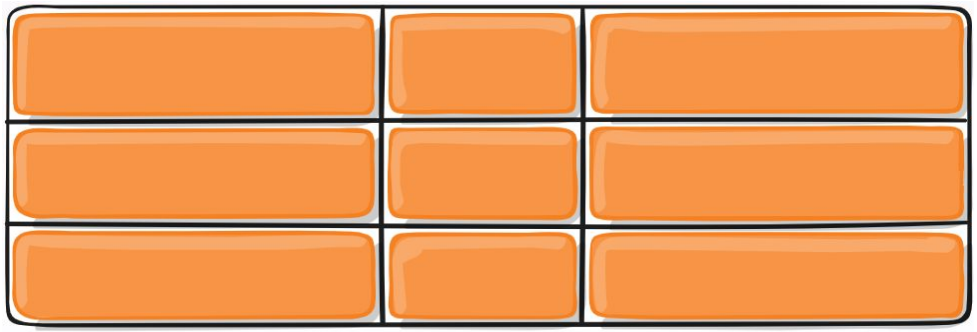


CODERHOUSE

Justify – Items: Center

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

```
.padre {
   justify-items: stretch;
                  /* predeterminado */
}
```

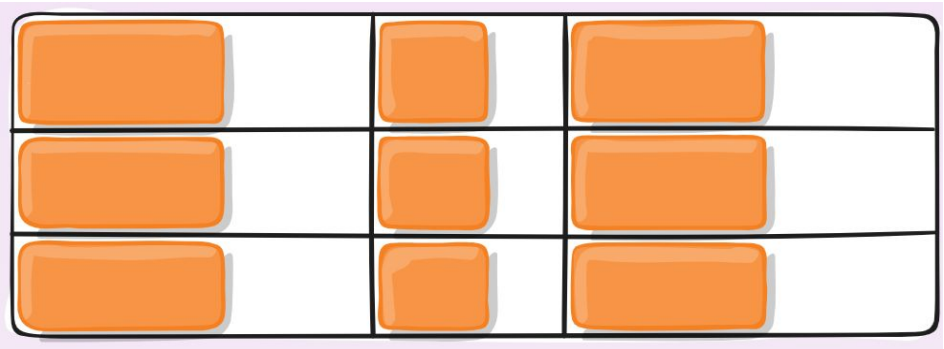


CODERHOUSE

Justify – Items: Start

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

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

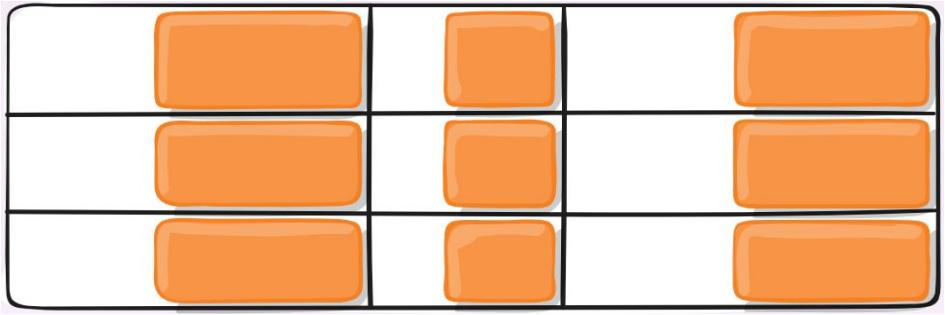


CODERHOUSE

Justify – Items: End

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

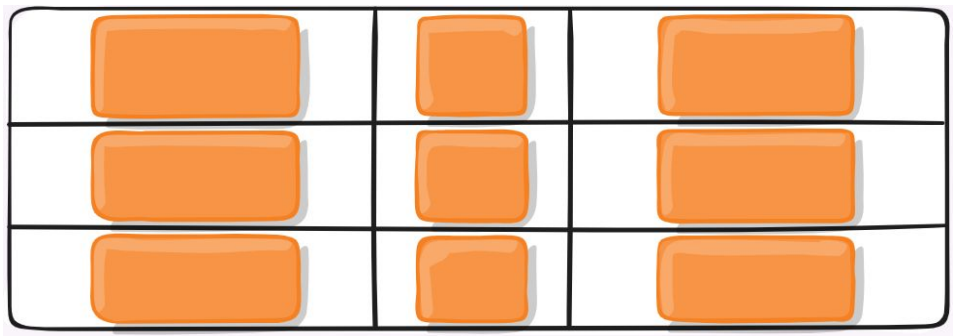
```
.padre {
    justify-items: end;
}
```



Justify – Items: Center

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

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

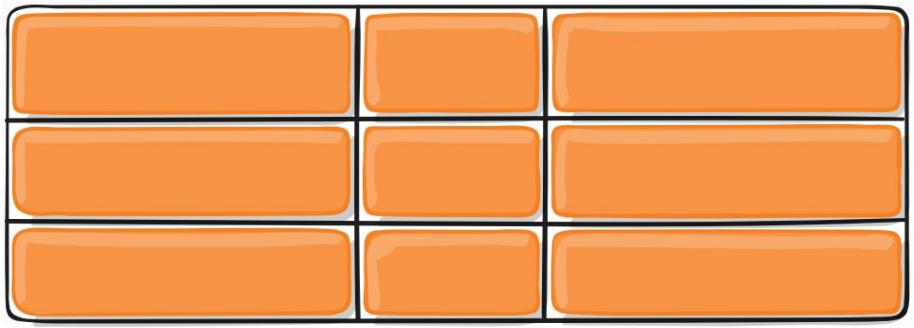


CODERHOUSE

Align – Items: Strech

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

```
.padre {
   align-items: stretch;
                /* predeterminado */
}
```

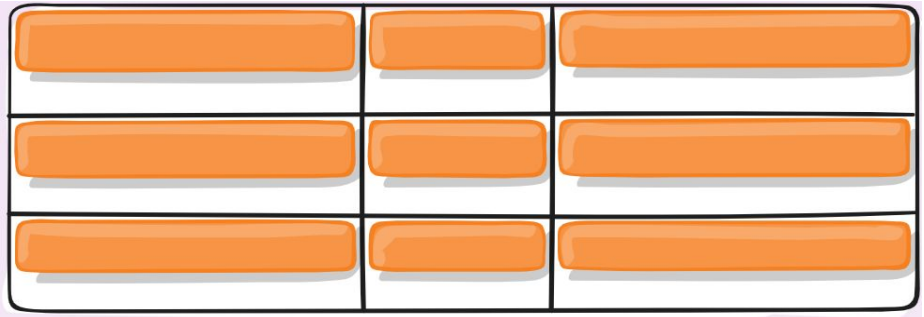


CODERHOUSE

Align – Items: Start

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

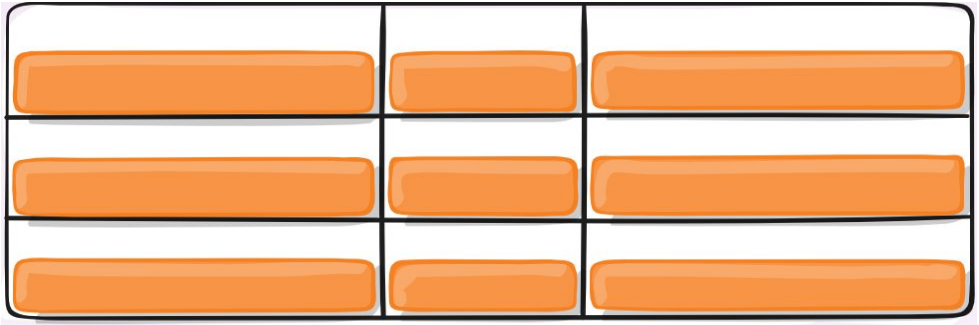
```
.padre {
    align-items: start;
}
```



Align – Items: End

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

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

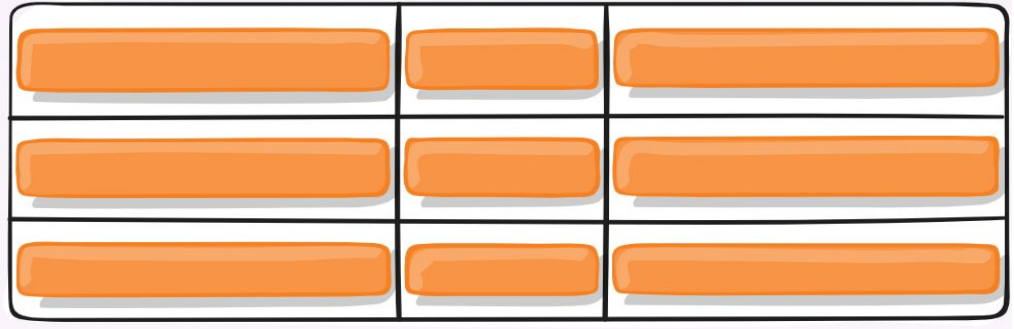


CODERHOUSE

Align – Items: Center

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

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



CODERHOUSE

Posición de Elementos

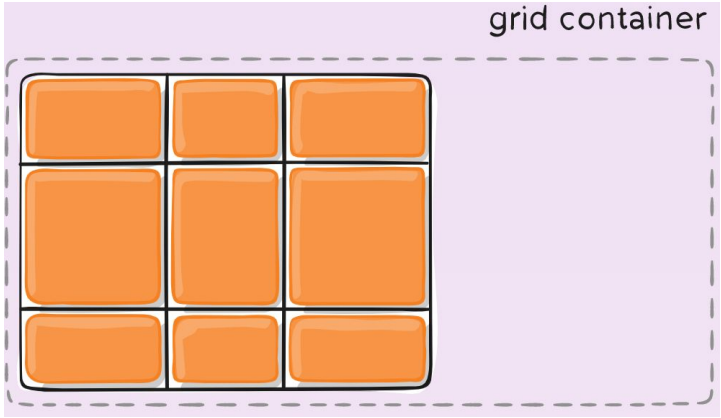
Es posible utilizar las propiedades justify-content o align-content para cambiar la distribución de todo el contenido en su conjunto. Puedes hacer pruebas en este enlace.

Propiedad	Valores	Afecta a
justify-content	start | end | center | stretch | space-around | space-between | space-evenly	Eje horizontal
align-content	start | end | center | stretch | space-around | space-between | space-evenly	Eje vertical

Justify – Content: Start

Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).

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

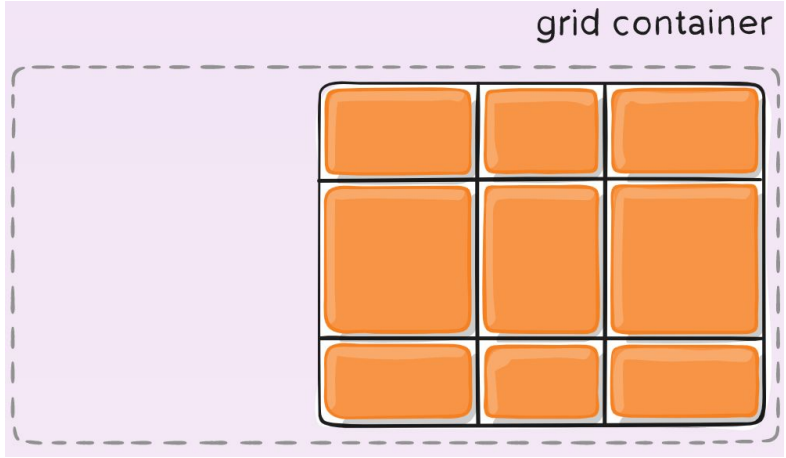


CODERHOUSE

Justify – Content: End

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

Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).

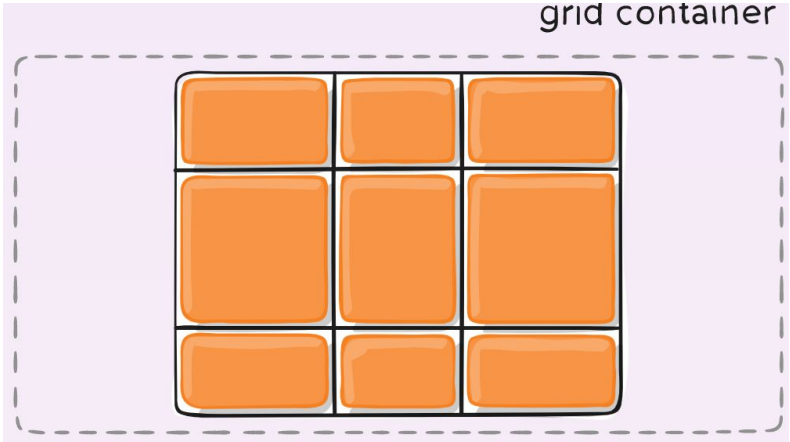


CODERHOUSE

Justify – Content: Center

Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).

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

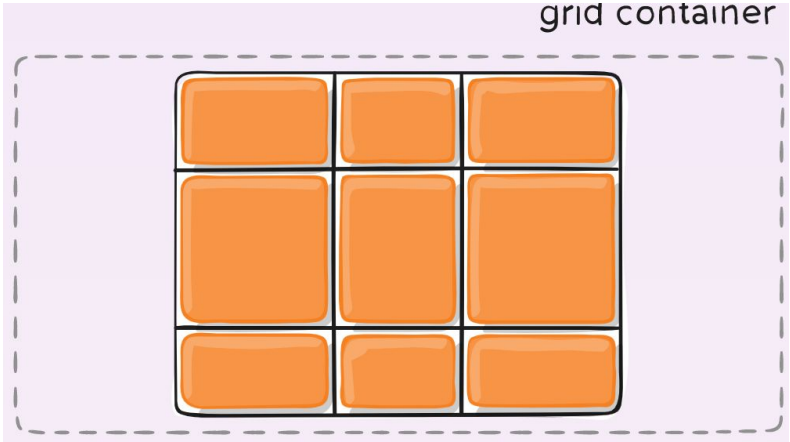


CODERHOUSE

Justify – Content: Stretch

Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).

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

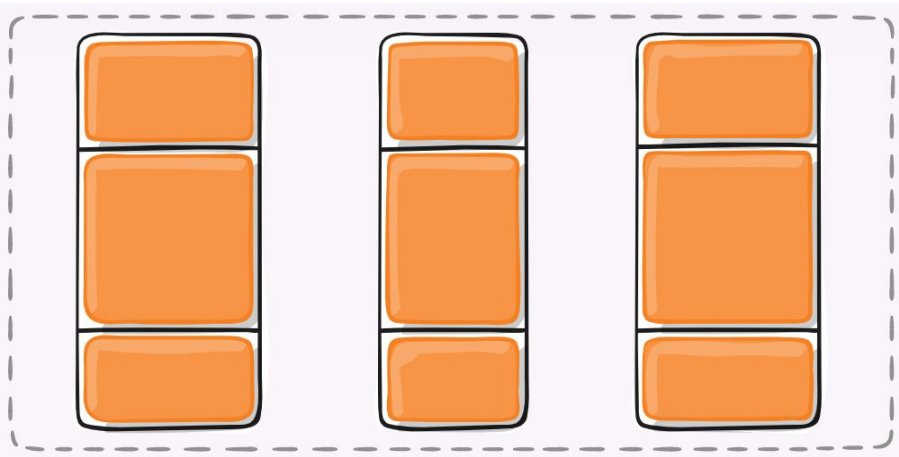


CODERHOUSE

Justify – Content: Space – Around

Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).

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

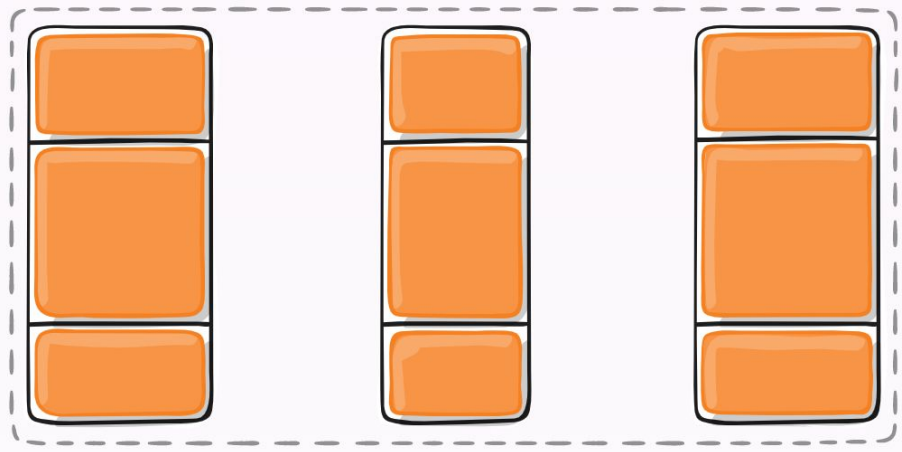


CODERHOUSE

Justify – Content: Space – Between

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

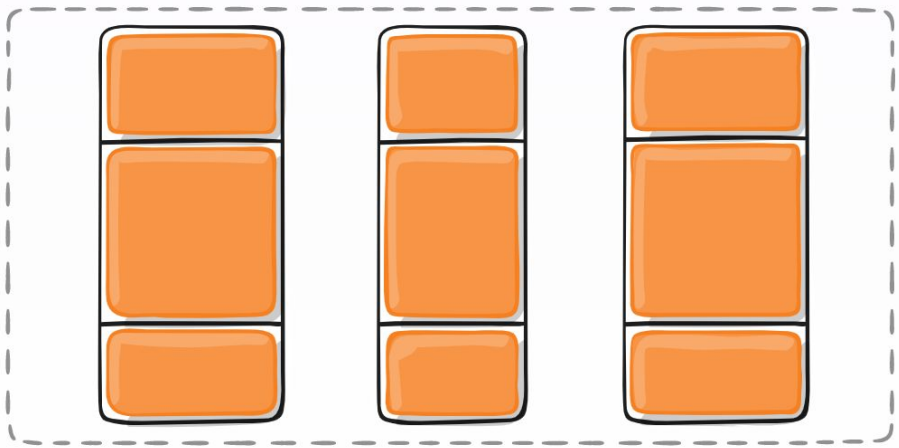
Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).



Justify – Content: Space – Evenly

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

Alineando todo el conjunto de celdas, de forma horizontal.
Dentro de su “padre” (es requisito que tenga ancho).

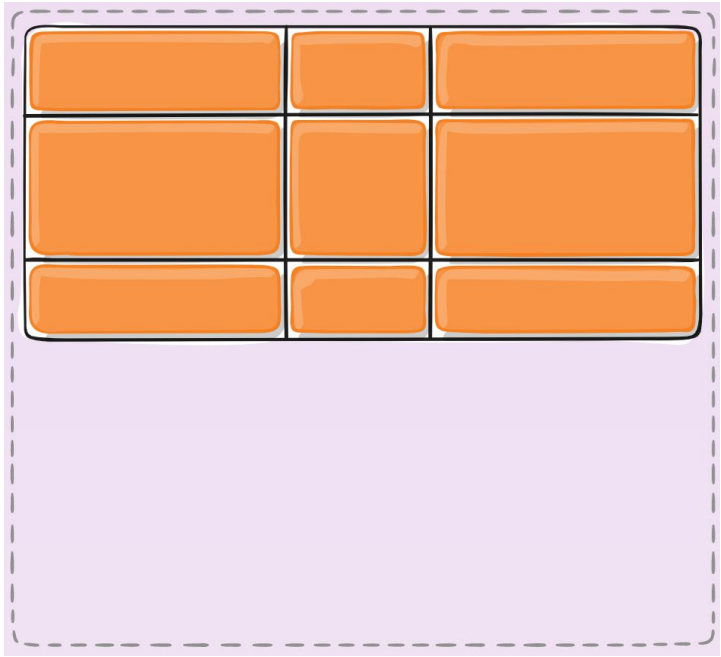


CODERHOUSE

Align – Content: Start

Alineando todo el conjunto de celdas, de forma vertical.
Dentro de su “padre” (es requisito que tenga altura).

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

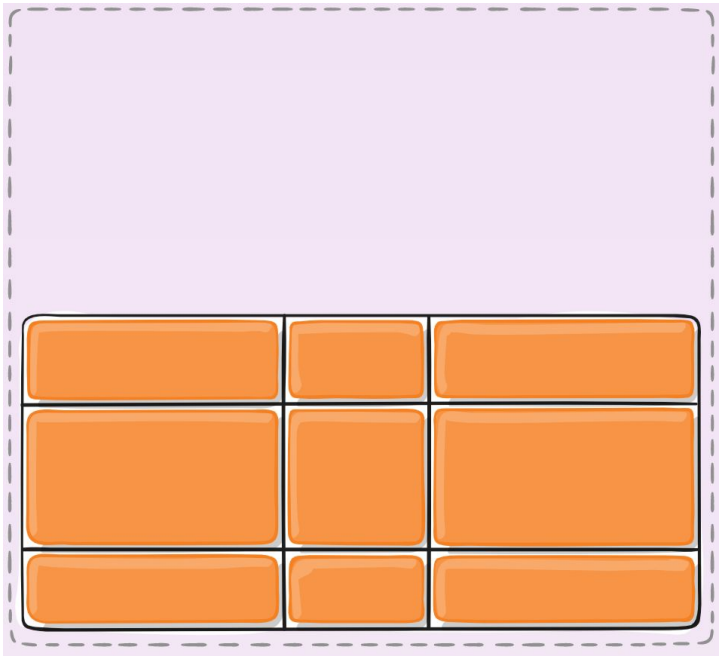


CODERHOUSE

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;
}
```

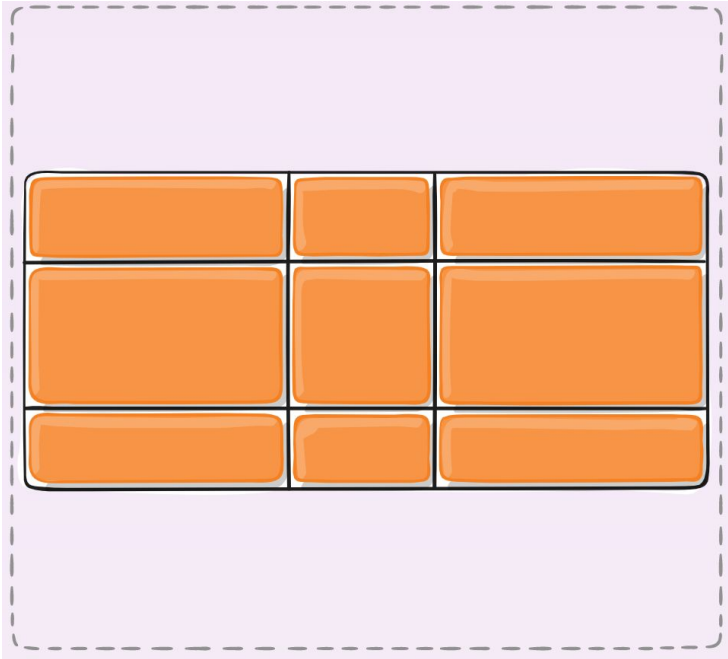


CODERHOUSE

Align – Content: Center

Alineando todo el conjunto de celdas, de forma vertical.
Dentro de su “padre” (es requisito que tenga altura).

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

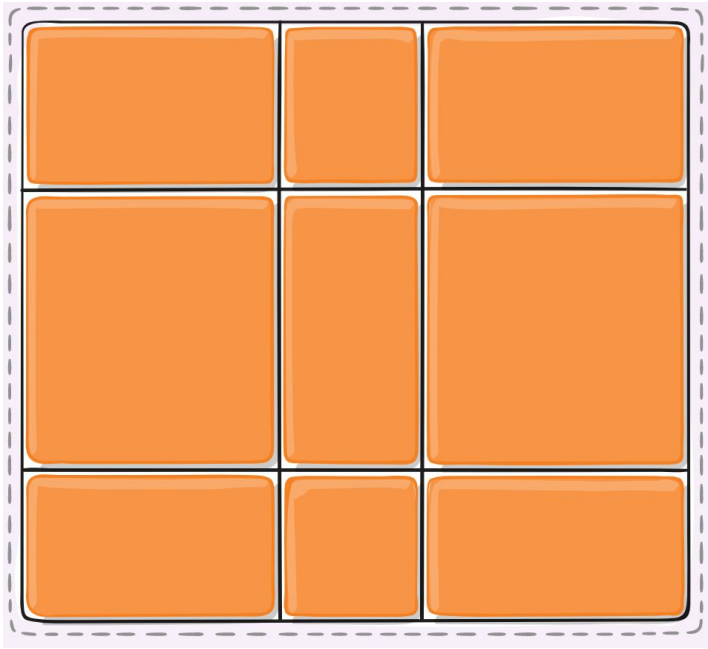


CODERHOUSE

Align – Content: Stretch

Alineando todo el conjunto de celdas, de forma vertical.
Dentro de su “padre” (es requisito que tenga altura).

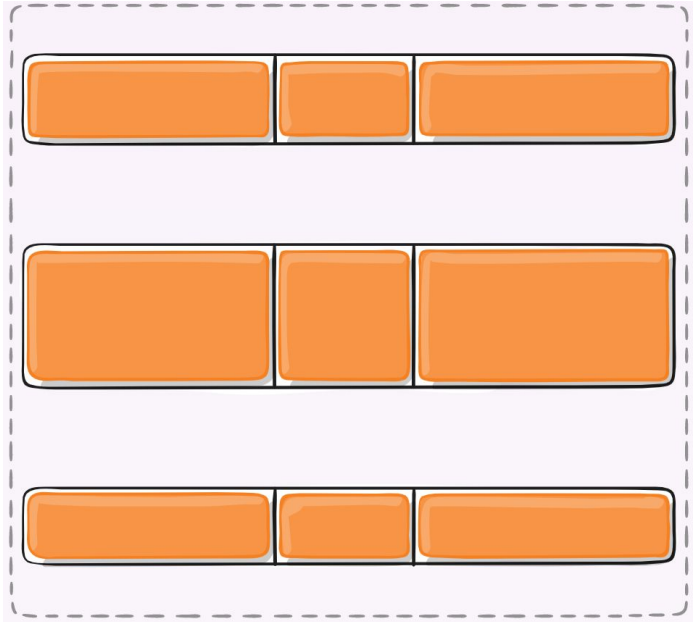
```
.padre {
    align-content: stretch;
}
```



Align – Content: Space – Around

Alineando todo el conjunto de celdas, de forma vertical.
Dentro de su “padre” (es requisito que tenga altura).

```
.padre {
    align-content: space-around;
}
```

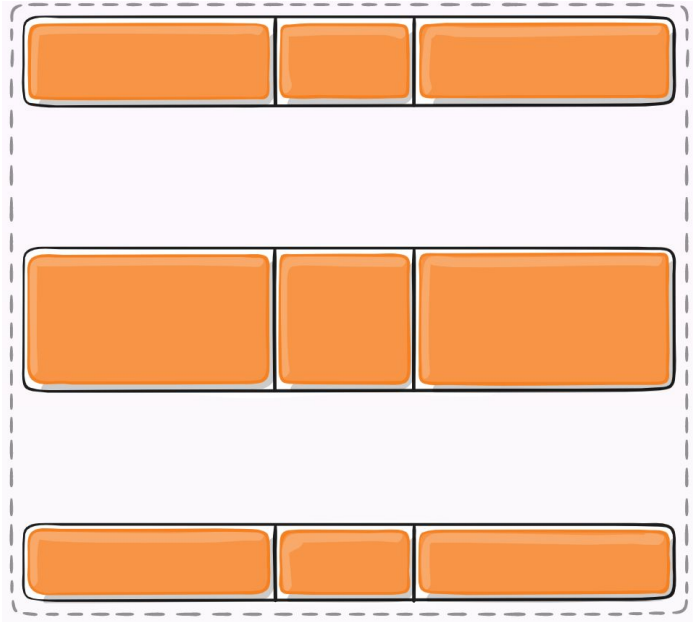


CODERHOUSE

Align – Content: Space – Between

Alineando todo el conjunto de celdas, de forma vertical.
Dentro de su “padre” (es requisito que tenga altura).

```
.padre {
    align-content: space-between;
}
```

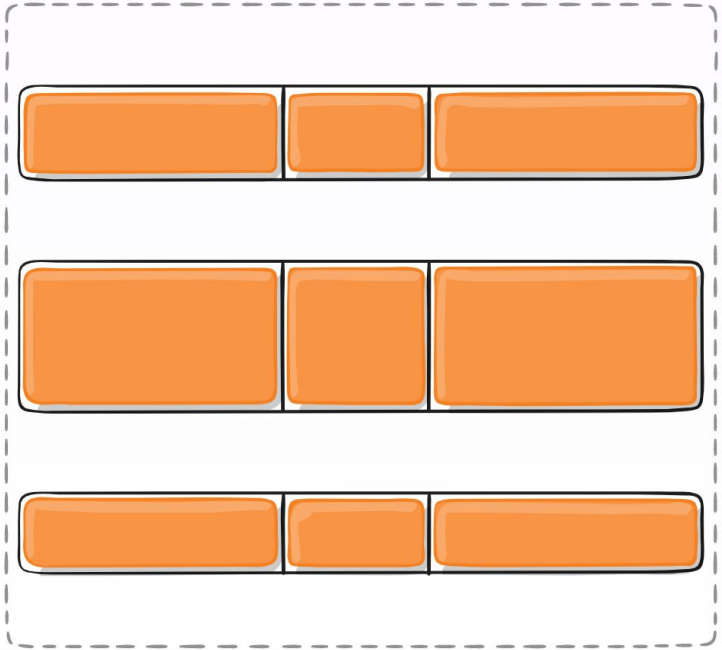


CODERHOUSE

Align – Content: Space – Evenly

Alineando todo el conjunto de celdas, de forma vertical.
Dentro de su “padre” (es requisito que tenga altura).

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



Ítems propiedades

CODERHOUSE

Ítems propiedades

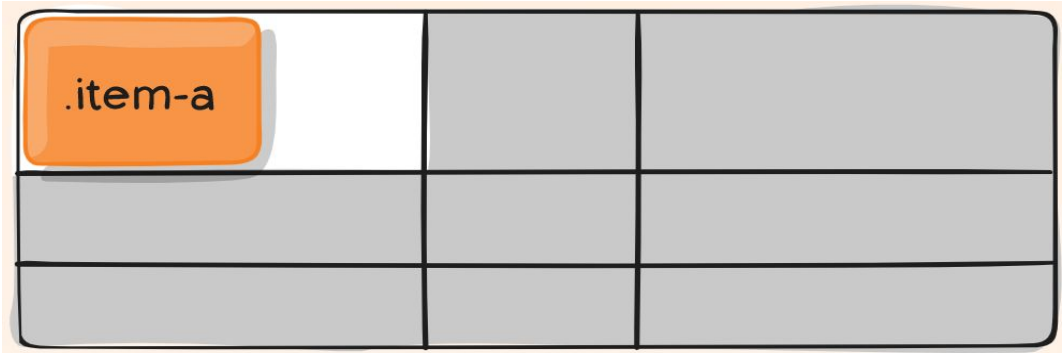
☝Hasta ahora hemos visto propiedades CSS que se aplican solamente al contenedor padre de una cuadrícula.
✌Ahora vamos a ver ciertas propiedades que se aplican a cada ítem hijo de la cuadrícula, para alterar o cambiar el comportamiento específico de dicho elemento.
Prueba sus propiedades aquí.

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

Justify – Self: Start

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

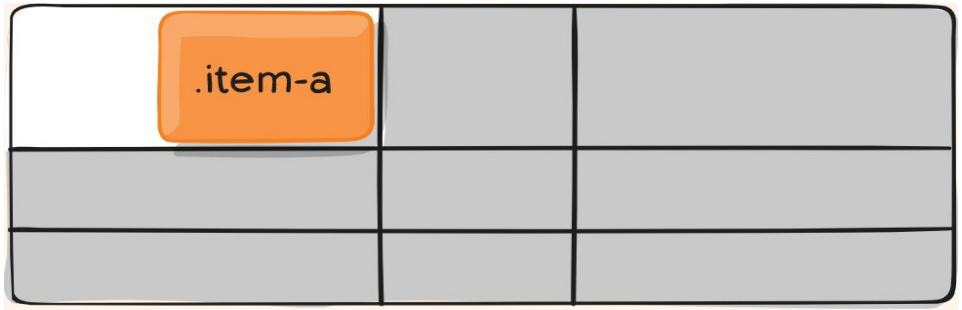
```
.hijo {
    justify-self: start;
}
```



Justify – Self: End

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

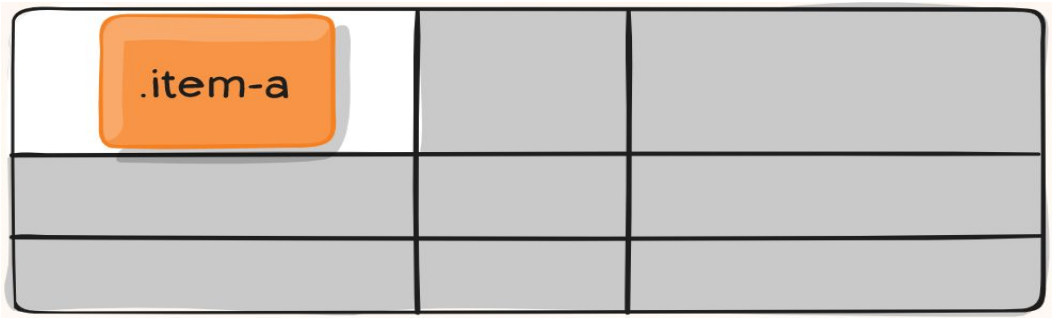
```
.hijo {
    justify-self: end;
}
```



Justify – Self: Center

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

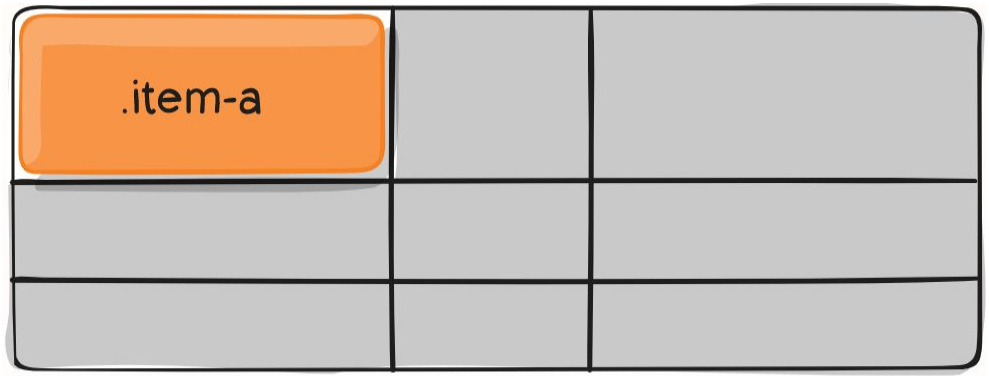
```
.hijo {
   justify-self: center;
}
```



Justify – Self: Stretch

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

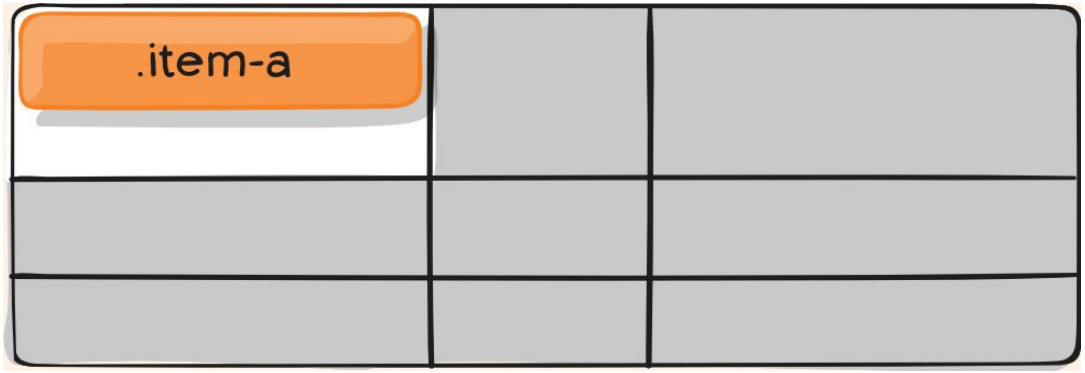
```
.hijo {
   justify-self: stretch;
}
```



Align – Self: Start

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

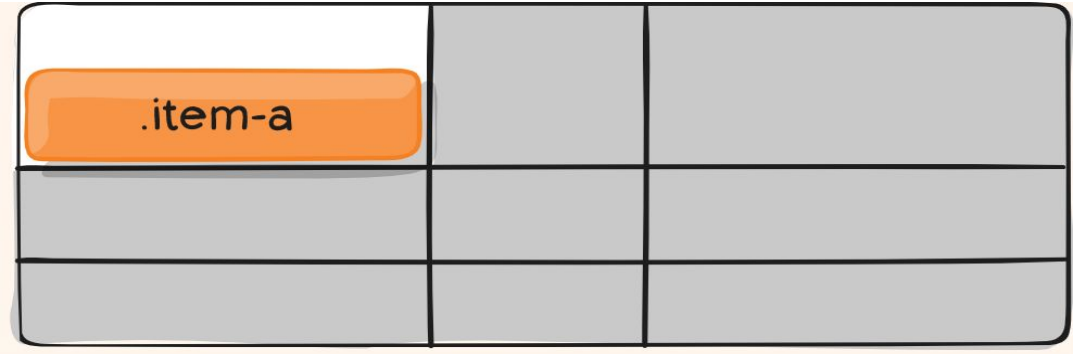
```
.hijo {
   align-self: start;
}
```



Align – Self: End

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

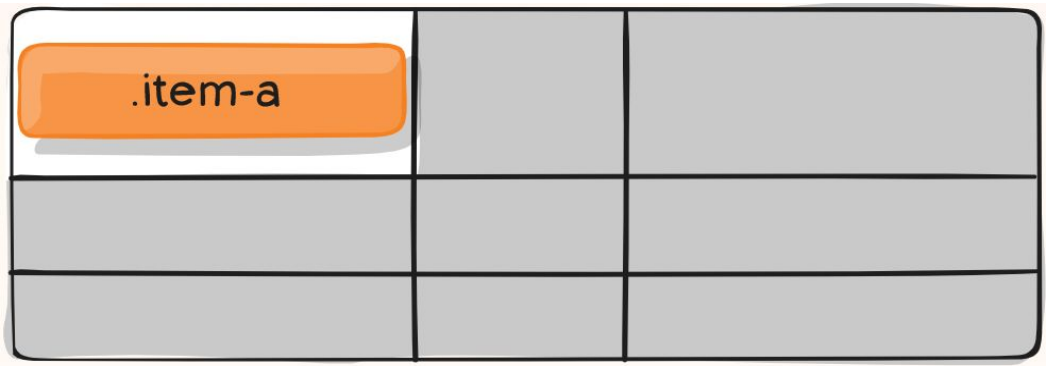
```
.hijo {
    align-self: end;
}
```



Align – Self: Center

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

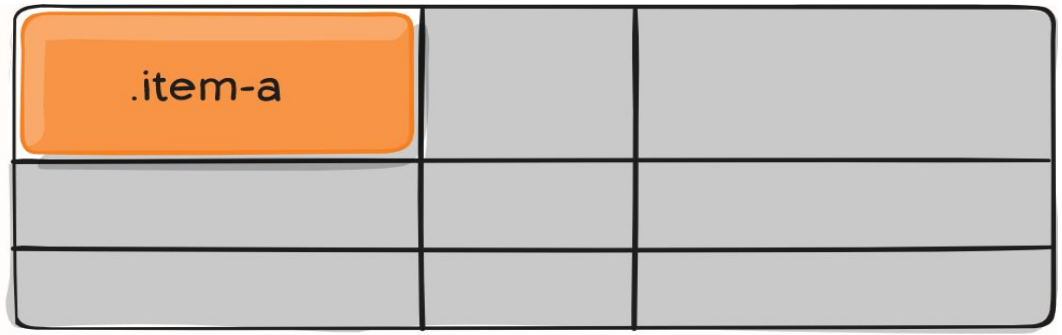
```
.hijo {
    align-self: center;
}
```



Align – Self: Stretch

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

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



Break

¡10 minutos y volvemos!

Ejemplo en vivo

¡Vamos a practicar lo visto!

¿Preguntas?

Resumen
de la clase hoy

- ✓ Grids.
- ✓ Nuevas formas de modelar Grids

Opina y valora esta clase

Muchas gracias.

#DemocratizandoLaEducación