

Esta clase va a ser

- grabada

CODERHOUSE

Clase 06. DESARROLLO WEB

Grids

CODERHOUSE

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).

- ✓ **Flex-direction:** esta propiedad nos va a permitir especificar si queremos que los flex items se dispongan en filas o columnas.
- ✓ **Flex-wrap:** permite especificar si queremos que los ítems puedan saltar a una nueva línea, cuando el contenedor flexible se quede sin espacio.
- ✓ **Flex-flow:** Es la forma abreviada (shorthand) o rápida para las propiedades: flex-direction y flex-wrap. Se pone primero el valor de la propiedad de flex-direction, y luego el de flex-wrap.

CODERHOUSE

Glosario

- ✓ **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.
- ✓ **Align-items:** Así como justify-content alinea los elementos en el eje principal, align-items hace lo propio, pero en el eje secundario. Este será el opuesto al principal
- ✓ **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.

CODERHOUSE

Objetivos de la clase

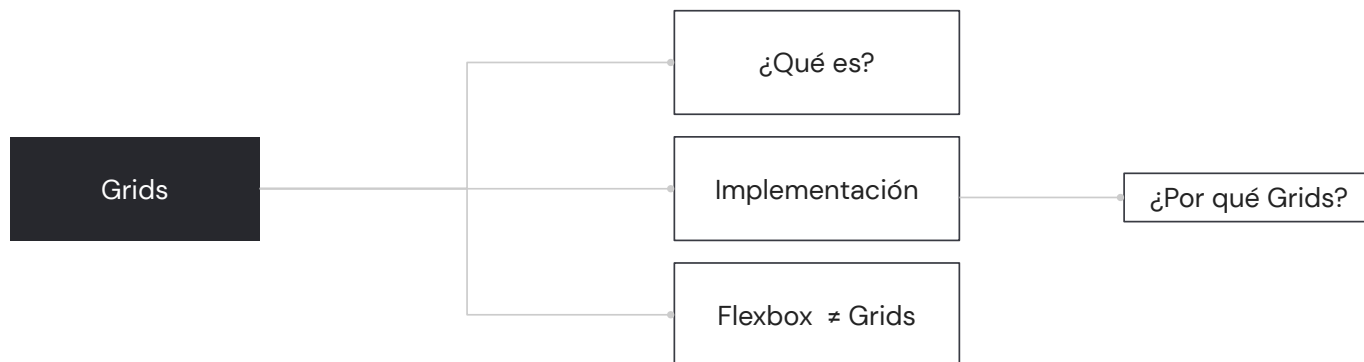
- Conocer Grids.
- Comparar el alcance de Grids y Flexbox.
- Aplicar las nuevas formas de modelar con Grids.

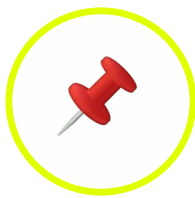
CODERHOUSE

Temario

05	06	07
Flexbox	Grids	Grids II
<ul style="list-style-type: none">✓ Flexbox✓ Propiedad de padres e hijos	<ul style="list-style-type: none">✓ Grids✓ Grids y Flexbox✓ Implementar Grids	<ul style="list-style-type: none">✓ Grids Mobile First✓ Media queries✓ Grids+Flex+@Media

CODERHOUSE





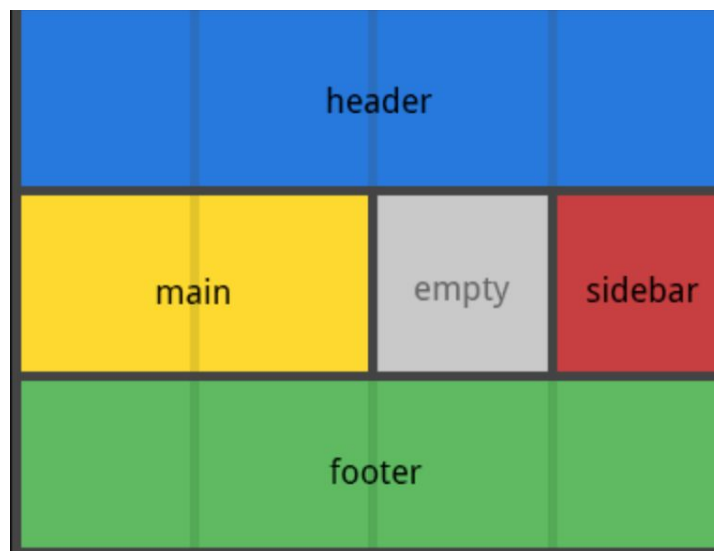
¿Qué es Grids?

👉 CSS Grid es un sistema de maquetación muy versátil que está disponible para CSS. 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.

👉 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.

CODERHOUSE



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.**

CODERHOUSE

¿Por qué Grids?

¿Position? ¿Float? ¿Elementos de Bloque?
¿Elementos de líneas?

¿Son suficientes estas herramientas para crear los layouts y estructuras de las páginas web actuales?

Y... ¿Flexbox?

¿Necesitaremos algo más potente para estructuras web?



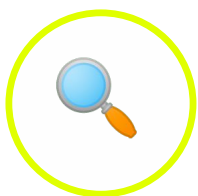
¿Por qué Grids?

¡SI!

Los complementos vistos anteriormente suelen ser insuficientes, o a veces un poco complejos para crear un layout/estructura 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?

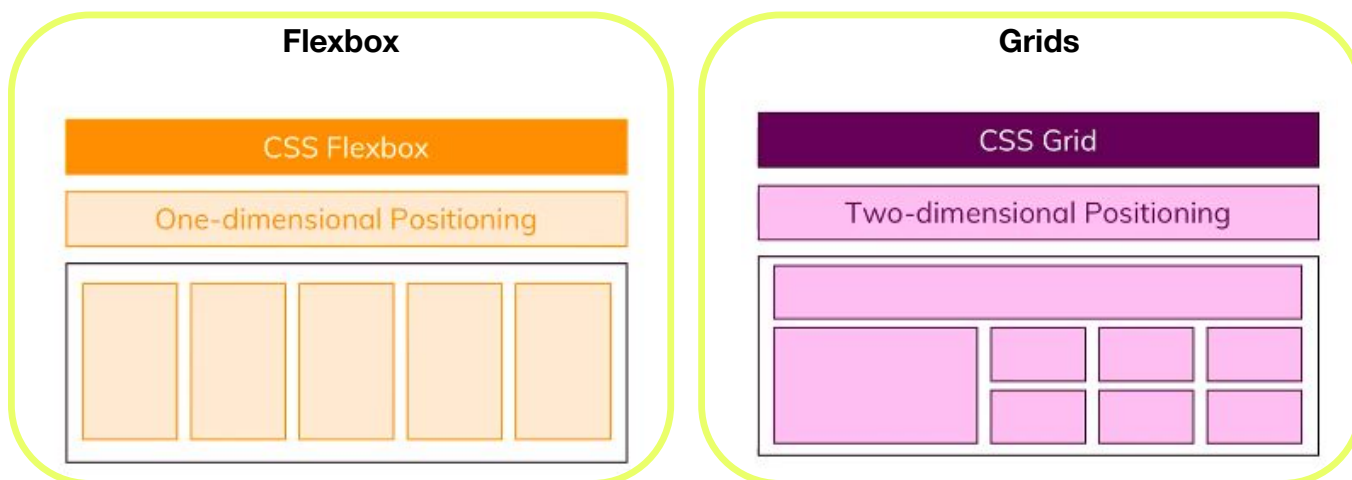
👉 Grid CSS surge de la necesidad de algo más potente, y toma las ventajas del sistema Flexbox, sumándole muchas mejoras y características que permiten crear muy rápido cuadrículas sencillas y versátiles.

👉 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.

CODERHOUSE

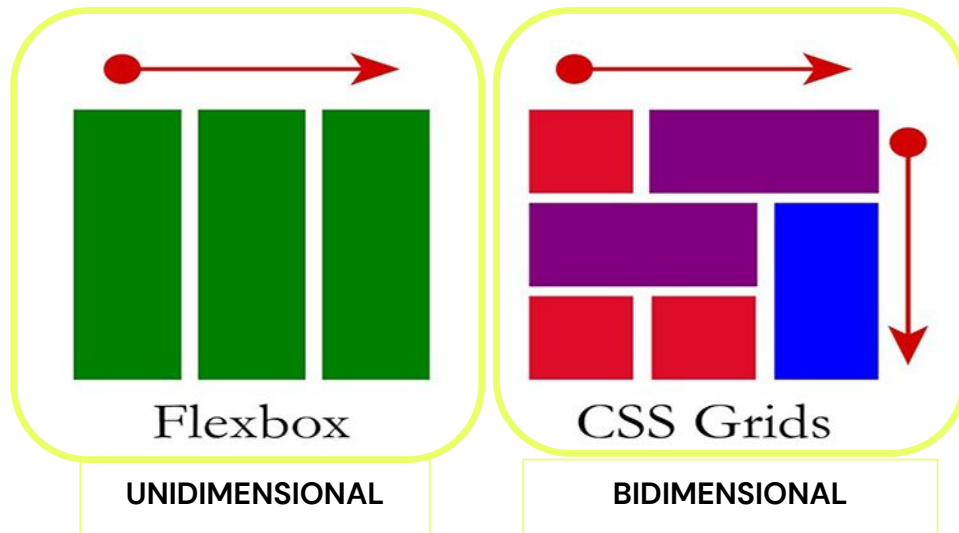
Diferencia entre Flexbox y Grids



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

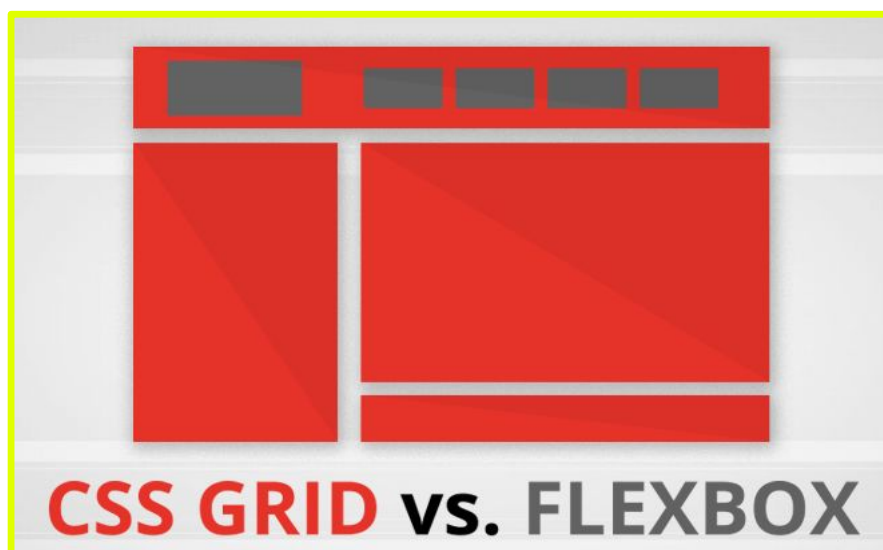
CODERHOUSE

Diferencia entre Flexbox y Grids



CODERHOUSE

Pueden convivir



CODERHOUSE

Implementar Grids

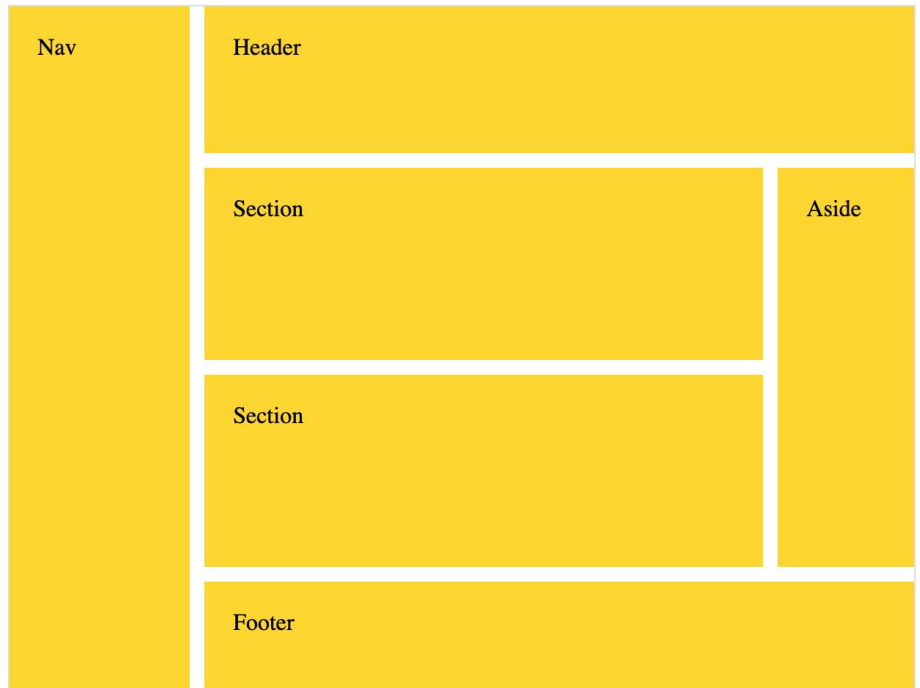
CODERHOUSE

Implementando Grid

CODERHOUSE

El objetivo

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



CODERHOUSE

Activando la grilla



Para empezar debemos utilizar, sobre el elemento contenedor (el *padre*), la propiedad `display` con el valor `grid` o `inline-grid`. El primer valor permite que la grilla aparezca encima/debajo del contenido exterior (en bloque). El segundo permite que la cuadrícula se vea a la izquierda/derecha (en línea) del contenido exterior.

HTML

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

CSS

```
.grid-container {
  display: grid;
}
```

CODERHOUSE

Propiedades del contenedor padre

CODERHOUSE

Propiedades del padre



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

CODERHOUSE

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

CODERHOUSE

Filas y columnas explícitas

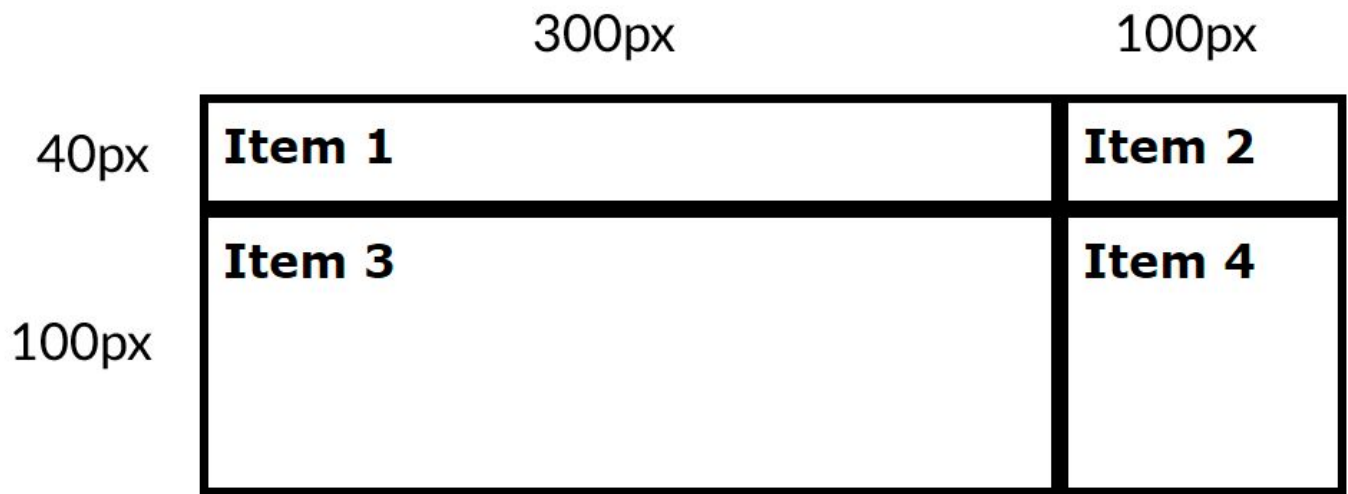


👀 Veamos la forma más simple de crear una grilla, especificando cuántas columnas y filas queremos. Lo haremos usando el html declarado anteriormente.

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

CODERHOUSE

Filas y columnas explícitas



CODERHOUSE

Filas y columnas explícitas



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

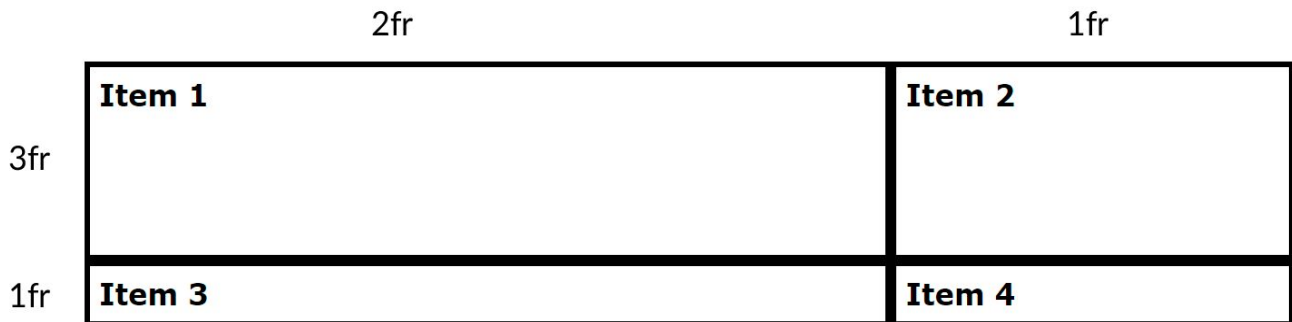
```
.grid-container {  
  display: grid;  
  grid-template-columns: 2fr 1fr;  
  grid-template-rows: 3fr 1fr;  
}
```

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

CODERHOUSE

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):



CODERHOUSE

Filas y columnas explícitas



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

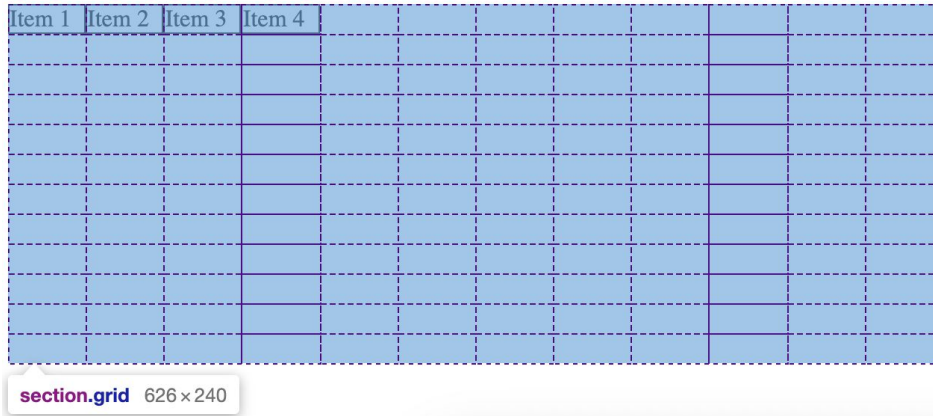
```
.grid-container {  
  display: grid;  
  grid-template-columns: repeat(12, 1fr);  
  grid-template-rows: repeat(12, 1fr)  
}
```

`repeat([número de veces], [valor o valores])`

CODERHOUSE

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.

CODERHOUSE

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.

CODERHOUSE



Ejemplo en vivo

¡Vamos a practicar lo visto!

CODERHOUSE

Siguiendo la línea del objetivo que tenemos podemos hacer lo siguiente...



CODERHOUSE

Grid por áreas



Este será el HTML base. Es muy importante marcar la estructura HTML.

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">Navegación</nav>
  <aside class="border">Aside</aside>
  <footer class="border">Pie de página</footer>
</div>
```

CODERHOUSE

Grid por áreas



CSS

```
#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;
}
```

CODERHOUSE

Grid por áreas

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

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

Navegacion	Header	
	Section	Aside
	Section	
	Pie de pagina	

CODERHOUSE

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.

CODERHOUSE

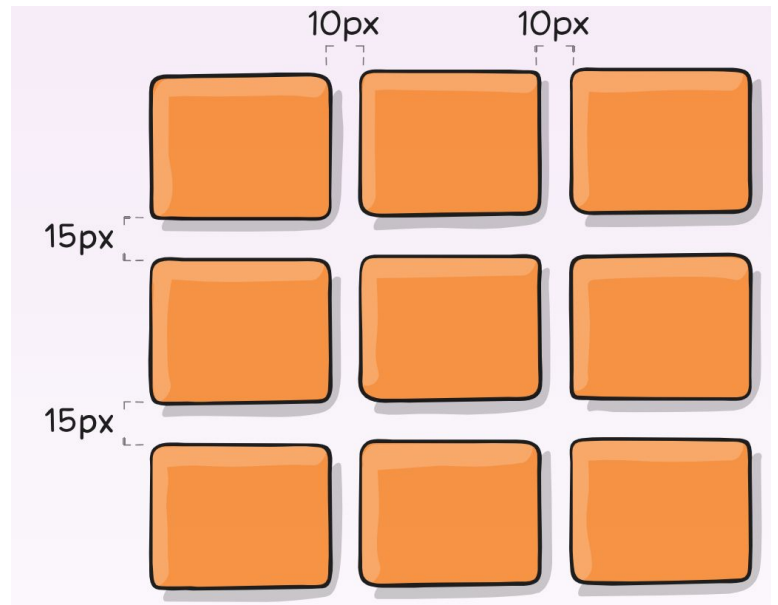
Grid gap



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: los **huecos (gutters)**.

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



CODERHOUSE

Combinando propiedades



Aplicamos las propiedades vistas, agregando más estilos al padre (el contenedor).

```
#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%;  
  row-gap: 10px;  
  column-gap: 10px;  
  height: 100vh;  
  margin: 0;  
}
```

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

CODERHOUSE



Break

¡10 minutos y volvemos!

CODERHOUSE

**Posicionamiento de los
hijos (desde el padre)**

CODERHOUSE

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, usando *justify-items* y *align-items*, que ya conocemos del módulo CSS Flexbox.

CODERHOUSE

justify-items y align-items



Veamos un ejemplo:

HTML

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

CODERHOUSE

justify-items y align-items



CSS

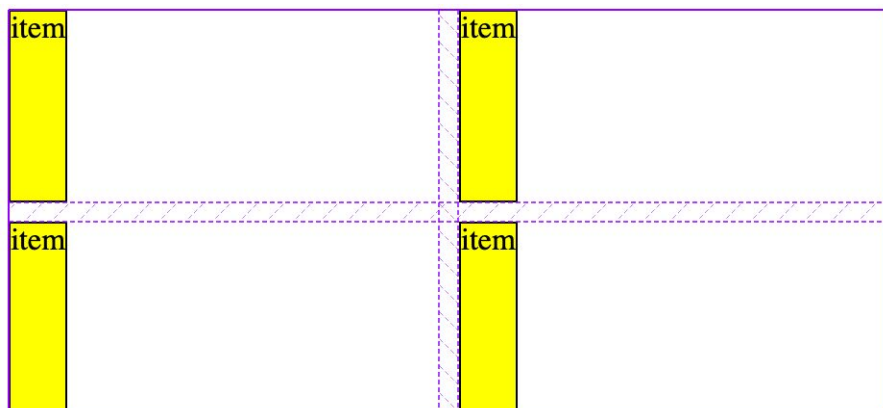
```
.grid-container {  
  justify-items: start; /* start | end |  
  center | stretch */  
  align-items: stretch; /* start | end |  
  center | stretch */  
  display: grid;  
  width: 95%;  
  grid-template-columns: auto auto;  
  grid-column-gap: 10px;  
  grid-template-rows: 100px 100px;  
  grid-row-gap: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

```
.grid-container .grid-item {  
  border: solid 1px;  
  font-size: 21px;  
  padding: 5px;  
  background-color: yellow;  
}
```

CODERHOUSE

justify-items y align-items

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



CODERHOUSE

Posición de hijos (desde el padre): resumen

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

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.

CODERHOUSE

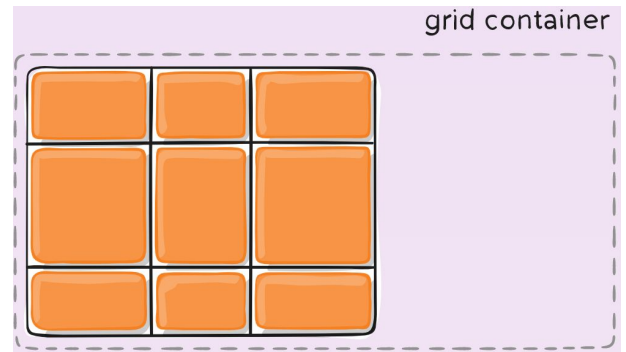
justify-content

Esta propiedad alinea **todo** el conjunto de celdas, de forma **horizontal**, dentro de su "padre" (es requisito que tenga **ancho**).

Su valores posibles son:

- ✓ start
- ✓ end
- ✓ center
- ✓ stretch
- ✓ space-around
- ✓ space-between
- ✓ space-evenly

```
.grid-container {  
  justify-content: start;  
}
```



CODERHOUSE

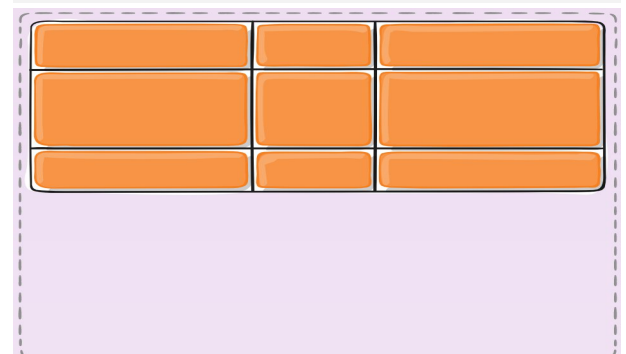
align-content

Esta propiedad alinea **todo** el conjunto de celdas, de forma **vertical**, dentro de su "padre" (es requisito que tenga **altura**).

Su valores posibles son:

- ✓ start
- ✓ end
- ✓ center
- ✓ stretch
- ✓ space-around
- ✓ space-between
- ✓ space-evenly

```
.grid-container {  
  align-content: start;  
}
```



CODERHOUSE

Posición de Elementos: resumen



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

Puedes hacer pruebas en [este enlace](#).

CODERHOUSE

Propiedades de los ítems hijos

CODERHOUSE

Propiedades de los ítems

👉 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> .

CODERHOUSE

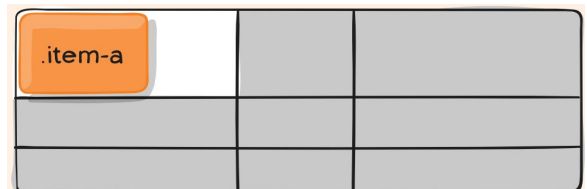
justify-self

Alínea **específicamente** a la celda (ítem, hijo) que necesites, de forma **horizontal**.

Su valores posibles son:

- ✓ start
- ✓ end
- ✓ center
- ✓ stretch

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



CODERHOUSE

align-self

Alínea **específicamente** a la celda (item, hijo) que necesites, de forma **vertical**.
Su valores posibles son:

- ✓ start
- ✓ end
- ✓ center
- ✓ stretch

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

.item-a		

CODERHOUSE



Ejemplo en vivo

¡Vamos a practicar lo visto!

CODERHOUSE

Profundiza y conoce atajos en
nuestra [Cheat Sheet](#)



CODERHOUSE

¿Preguntas?

CODERHOUSE

Resumen de la clase hoy

- ✓ Grids.
- ✓ Nuevos layouts con Grids.

CODERHOUSE

Opina y valora esta clase

CODERHOUSE

Muchas gracias.

CODERHOUSE

#DemocratizandoLaEducación

CODERHOUSE