

Tabla de contenido

Introducción	1
Usando fr	3
Usando otros tamaños	3
Usando repeat	4
Usando min-max.....	5
Tamaño de las filas o columnas adicionales.....	6
Espacio entre celdas.....	6
Usando grid-column y grid-row	6
Ocupar más de una fila y/o columna	10
Creando áreas con nombres	13
Cambiar como se colocan los elementos restantes	14
Alinear los elementos horizontalmente en su celda.....	15
Alinear los elementos verticalmente en su celda	16
Alinear los elementos horizontal y verticalmente en su celda.....	17
Alinear todas las celdas horizontalmente dentro de la rejilla	17
Alinear todas las celdas verticalmente dentro de la rejilla	17
Alinear todas las celdas horizontal y verticalmente en la rejilla	18
Anidaciones	18
Mansory / Albañilería	18

Introducción

Grid surge después de Flexbox para disponer los elementos de una página web en forma de rejilla, ya que Flexbox está pensado para crear disposiciones en filas o columnas, pero no combinaciones de ambas, debiendo usar contenedores Flexbox dentro de otros para conseguirlo, aunque con más complejidad y limitaciones.

Lo primero que debemos hacer es asignar la propiedad **display** con el valor **grid** o **inline-grid** al contenedor deseado.

Luego debemos indicar una o las dos de las siguientes propiedades, también en el contenedor:

- **grid-template-columns:** para indicar el número de columnas y ancho que tendrán
- **grid-template-rows:** para indicar el número de filas y alto que tendrán

En ellas indicaremos las columnas y/o filas que queremos mediante valores separados por espacios, los cuales pueden ir en píxeles, porcentajes, además de fr y auto que veremos posteriormente, pudiendo usar varios de ellos al mismo tiempo.

Ejemplo:

El CSS:

```
.contenedor {
  display: grid;
  grid-template-columns: 100px 200px;
  outline: solid;
}
.caja {
  background-color:rgb(28, 105, 220);
  color: white;
  text-align: center;
  font-size: 2rem;
  /* Dando la misma altura de línea que de fuente, el elemento se centrará
  verticalmente. Este truco solo vale para si el texto ocupa una sola línea */
  line-height: 2rem;
  padding: 2rem;
  font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', Verdana, sans-serif;
  border: 2px solid rgb(13, 119, 1);
}
```

El HTML:

```
<div class="contenedor">
  <div class="caja">1</div>
  <div class="caja">2</div>
  <div class="caja">3</div>
  <div class="caja">4</div>
  <div class="caja">5</div>
  <div class="caja">6</div>
</div>
```

1	2	
3	4	
5	6	

El contenedor sigue ocupando todo el ancho

Podemos ver que tenemos dos columnas, pues hemos indicado dos valores en la propiedad grid-template-columns. Aunque no hemos indicado el número de filas, los demás elementos se colocarán en filas.

Si usamos el valor auto, las columnas ocuparán todo el espacio del contenedor. Lo mismo pasará con las filas.

```
grid-template-columns: auto auto;
```

1	2
3	4
5	6

Añadimos ahora esta propiedad al contenedor:

```
grid-template-rows: 200px 300px;
```

alonso.jad@gmail.com

1	2
3	4
5	6

Como tenemos 6 elementos pero solo dos columnas y dos filas, los dos últimos tendrán un alto auto, pero ocupando solo el espacio que necesite su contenido, pues no hemos indicado un alto al contenedor; si lo hiciéramos ocuparán todo el alto restante.

Si el ancho o alto del contenedor es menor que el de sus elementos, al igual que otros casos, se saldrán por fuera de este.

Usando fr

Mediante fr podemos indicar una fracción o parte del espacio sobrante que debe ocupar un elemento.

Ejemplo:

```
/* Siguiendo el ejemplo anterior, pero sin indicar las filas */
grid-template-columns: 1fr 2fr;
```

1	2
3	4
5	6

La segunda cogerá un 66% del espacio sobrante y la primera un 33%:

- primera columna: $1\text{fr} / 3\text{fr} = 0,33$, siendo 3fr la suma de $1\text{fr} + 2\text{fr}$
- segunda columna: $2\text{fr} / 3\text{fr} = 0,66$

Ejemplo:

```
grid-template-columns: 200px 1fr 1fr;
```

1	2	3
4	5	6

Ahora la primera columna ocupará 100 píxeles de ancho y la segunda y tercera el tamaño de su contenido más un 50% del espacio sobrante cada una.

Usando otros tamaños

Además de los ya conocidos y vistos, podemos usar los siguientes:

alonso.jad@gmail.com

- **min-content:** que intente ocupar el menor tamaño posible. Por ejemplo, si tenemos una frase, que ocupe el ancho de la mayor.

lorem ipsum	2
-------------	---

- **max-content:** que intente ocupar el mayor tamaño posible. Por ejemplo, si tenemos una frase, que ocupe el ancho de toda la frase

lorem ipsum	2
-------------	---

- **min(tamaño):** un ancho o alto mínimo
- **max(tamaño):** un ancho o alto máximo

Usando repeat

También podemos usar **repeat**(númeroColumnas, ancho) para indicar el mismo valor para varias columnas.

Ejemplo:

```
grid-template-columns: repeat(3, 1fr);
```

1	2	3
4	5	6

Tenemos tres columnas ocupando cada una un 33% del espacio sobrante.

Es posible indicar repeat con otras opciones sin él.

Ejemplo:

```
grid-template-columns: 200px repeat(2, 1fr);
```

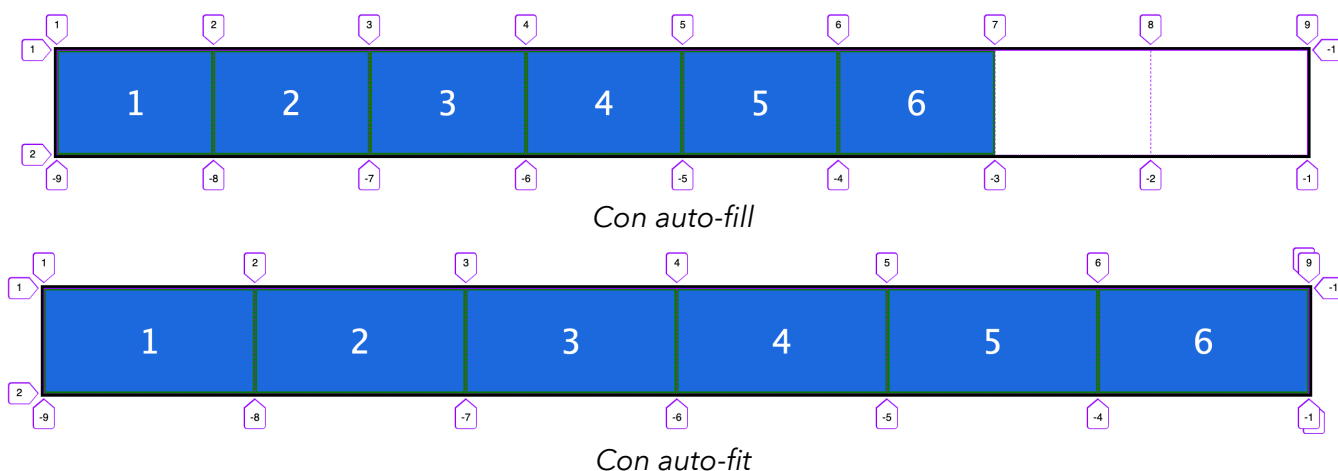
1	2	3
4	5	6

La primera columna ocupa 100 píxeles de ancho y las dos siguientes cada una un 50% del espacio sobrante.

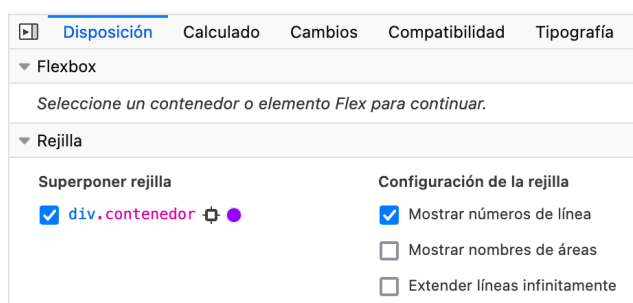
Una posibilidad interesante es usar algo como lo siguiente, que permite que se muestren tantas columnas con espacio haya para ellas:

```
grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
```

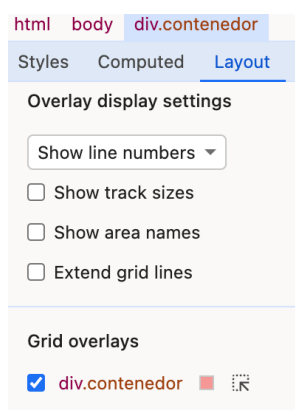
También existe la opción `auto-fill`, que es muy similar, pero mientras que `auto-fit`, si no hay más celdas, añade celdas sin ancho, agrandando las existentes para que quepan todo el ancho posible, mientras que `auto-fill` creará las celdas vacías que pueda para llenar el espacio.



Estos valores anteriores podemos verlos en el inspector de Firefox del contenedor marcando las siguientes opciones:



Y en Chrome:



Usando min-max

Con esta opción podemos indicar un ancho o alto mínimo y máximo para una columna o fila, de la forma **min-max(tamañoMínimo, tamañoMáximo)**.

Ejemplo:

```
grid-template-columns: minmax(100px, 1fr) minmax(200px, 1fr);
```

alonso.jad@gmail.com

La primera columna ocupará un mínimo de 100 píxeles y la segunda 200; ambas un máximo de un 50% de espacio sobrante.

1	2
3	4
5	6

Si el contenedor es muy grande, no veremos diferencia

1	2
3	4
5	6

Al bajar su tamaño, sí

Tamaño de las filas o columnas adicionales

Como ya hemos visto, si ponemos más filas o columnas de las especificadas, por defecto se les da un ancho y/o alto de auto, con lo que ocuparán su contenido. Podemos cambiar esto con las siguientes propiedades:

- **grid-auto-rows**: alto de las filas adicionales
- **grid-auto-columns**: ancho de las columnas adicionales

Espacio entre celdas

Disponemos de las siguientes propiedades para indicar el espacio deseado entre celdas:

- **row-gap**: espacio entre filas
- **column-gap**: espacio entre columnas
- **gap**: espacio entre filas y columnas

Ejemplo:

```
gap: 10px;
```

Usando grid-column y grid-row

Podemos usar estas dos propiedades en los elementos, para cambiar el sitio donde mostrarse, desplazándose las demás consecuentemente. En ellas indicaremos el número de fila o columna deseado.

Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto;
```

```
/* En el primer div */
grid-column: 2
```

	1
2	3
4	5
6	

```
/* En el segundo div */
grid-column: 1
```

1	
2	3
4	5
6	

```
/* En el primer div */
grid-column: 2;
/* En el segundo div */
grid-column: 2;
```

	1
	2
3	4
5	6

```
/* En el primer div */
grid-column: 3;
```

		1
2	3	4
5	6	

La nueva columna tendrá un ancho auto

```
/* En el primer div */
grid-row: 2;
```

2	3
1	4
5	6

Ahora los demás elementos no dejan un hueco vacío,
colocándose antes

```
/* En el primer div */  
grid-row: 2;  
grid-column: 2;
```

2	3
4	1
5	6

```
/* En el segundo div */  
grid-row: 2;  
grid-column: 2;
```

1	3
4	2
5	6

```
/* En el primer y segundo div */  
grid-row: 2;  
grid-column: 2;
```

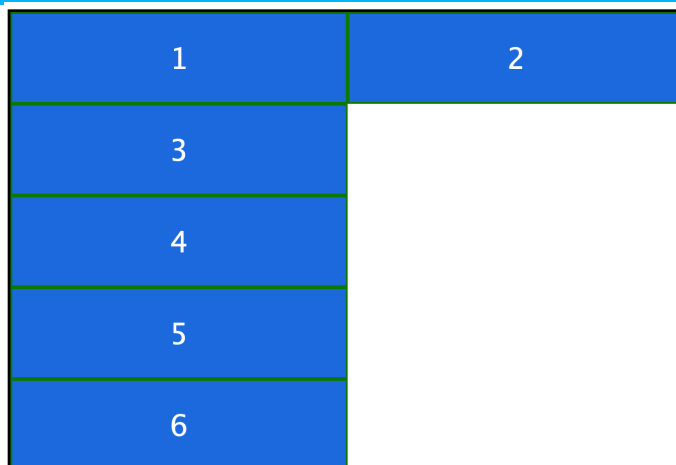
3	4
5	2
6	

Vemos que el primer div queda tapado por el segundo

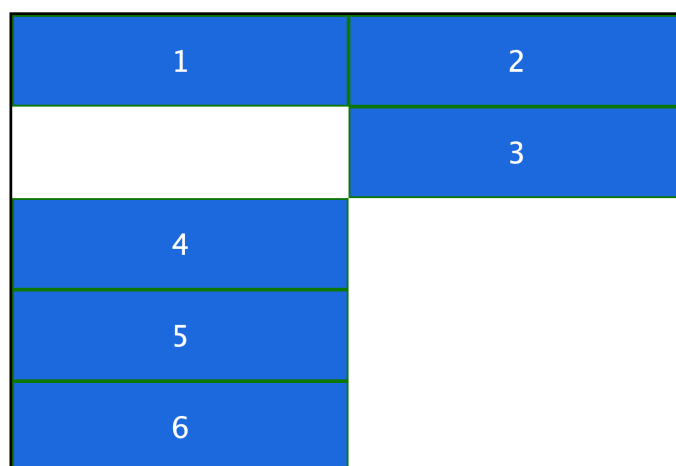
```
/* En todos los div */  
grid-column: 1;
```

1	
2	
3	
4	
5	
6	


```
/* En todos los div salvo el
segundo*/
grid-column: 1;
/* En el segundo div */
grid-column: 2;
```



```
/* En todos los div salvo el
segundo*/
grid-column: 1;
/* En el segundo y tercer div */
grid-column: 2;
```

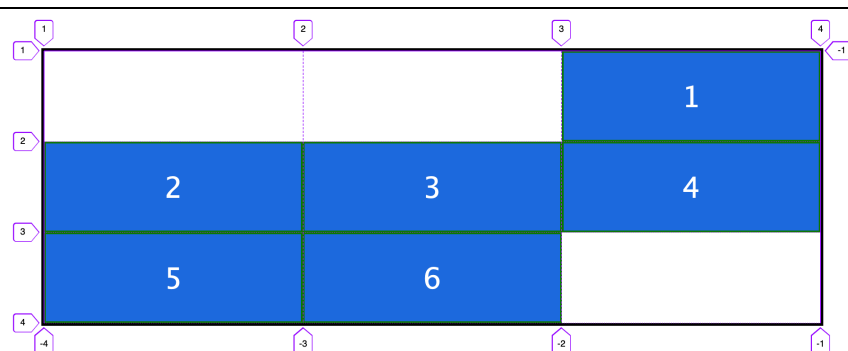


Podemos indicar también valores negativos, siendo -1 la última línea, -2 la penúltima, ...

Ejemplos:

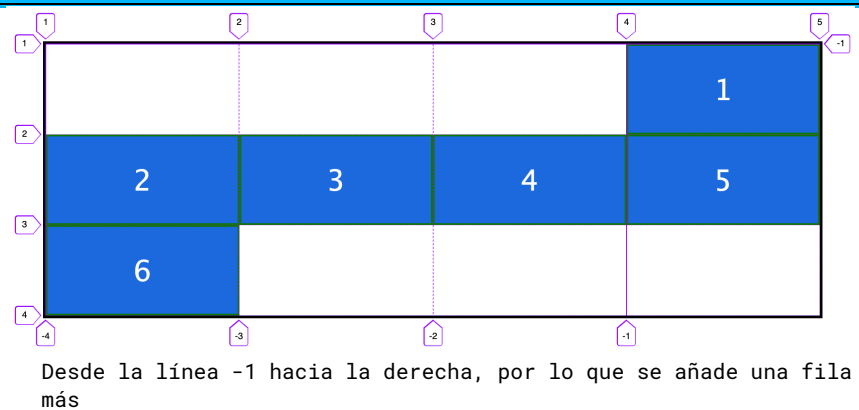
```
/* En el contenedor */
grid-template-columns: auto auto auto;
```

```
/* En el primer div */
grid-column: -2;
```



Desde la línea -2 hacia la derecha

```
/* En el primer div */
grid-column: -1;
```



Usando estos valores negativos podemos hacer que una celda ocupe todo el ancho, o a partir de la línea deseada, independientemente del número de columnas que haya, indicando -1 como línea hasta la que ocupar. Lo mismo para el alto.

Ejemplo:

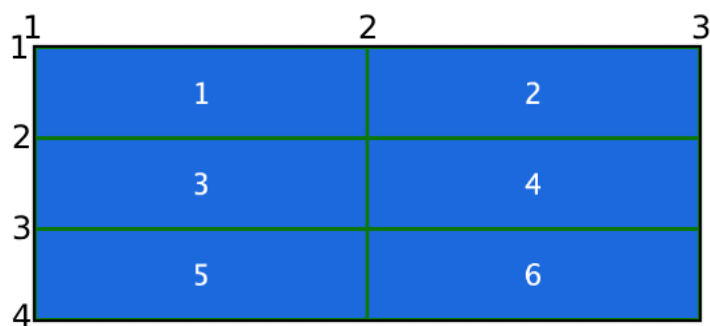
```
grid-column: 1 / -1;
```

También tenemos las propiedades **grid-row-start**, **grid-row-end**, **grid-column-start** y **grid-column-end**.

Ocupar más de una fila y/o columna

A modo similar de las propiedades **rowspan** y **colspan** de las tablas, podemos hacer que un elemento ocupe más de una fila y/o columna.

Para ello usaremos las propiedades **grid-row** y **grid-column** a las que indicaremos un segundo valor, separador por /, con la línea hasta la que ocupar (no incluida), no cuantas.



Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto;
```

```
/* En el primer div */
grid-column: 1 / 3;
```

1	
2	3
4	5
6	

```
/* En el primer div */
grid-column: 1 / 3;
grid-row: 1 / 3;
```

1	
2	3
4	5
6	

No vemos diferencia pues, al tener un alto de auto las filas, no ocupa más de lo que toca, con lo que parece que solo ocupa una fila

```
/* En el primer div */
grid-row: 1/3;
```

1	2
	3
4	5
6	

```
/* En el primer div */
grid-row: 1/3;
/* EN el cuarto div */
grid-column: 1/3;
```

1	2
	3
4	
5	6

Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
```

```
/* En el primer div */
grid-row: 1 / 3;
grid-column: 1 / 3;
```

1		2
		3
4	5	6

```
/* En el primer div */
grid-column: 2 / 4;
```

	1	
2	3	4
5	6	

```
/* En el primer div */
grid-column: 2 / 4;
/* En el segundo div */
grid-column: 1;
grid-row: 1 / 3
```

2	1	
	3	4
5	6	

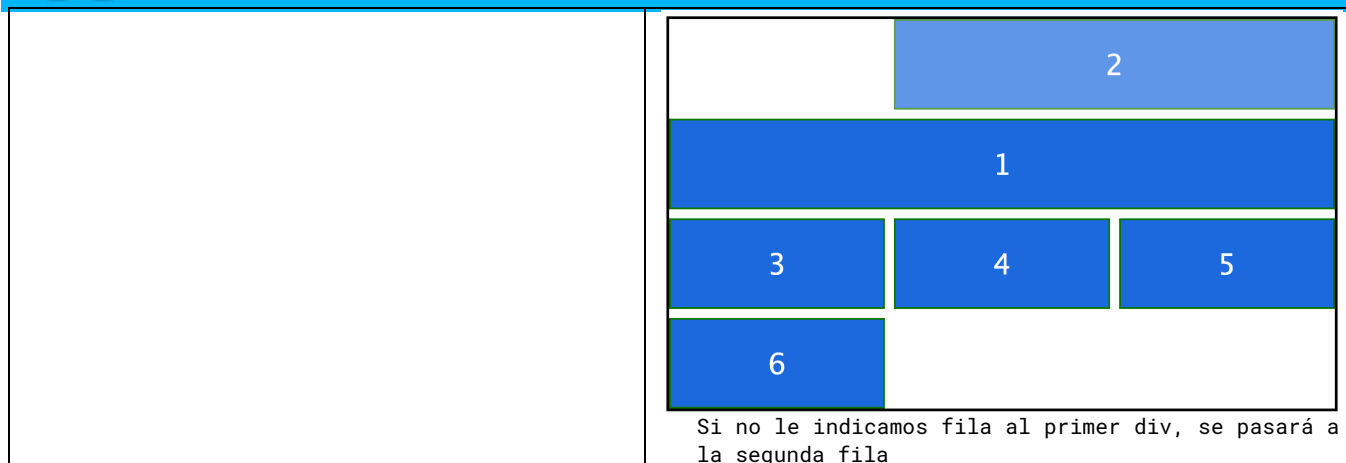
```
/* En el primer div */
grid-row: 1 / 5;
/* En el segundo div */
grid-column: 3;
grid-row: 1 / 5;
```

1	3	2
	4	
	5	
	6	

```
/* En el primer div */
grid-column: 1 / 4;
grid-row: 1;
/* En el segundo div */
grid-column: 2/4;
grid-row: 1;
opacity: 0.7
```

	1	2
3	4	5
6		

Podemos ver que parte del primer div queda por debajo del segundo div



Podemos usar la propiedad reducida **grid-area** para indicar todos los valores:

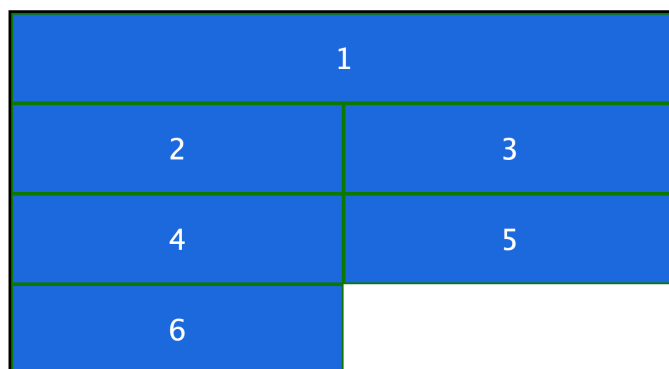
```
grid-area: (inicioFila / inicioColumna / finFila / finColumna)
```

Podemos usar **span** para indicar hasta que columna o fila ocupar en lugar de la línea.

Ejemplo:

```
/* En el contenedor */
grid-template-columns: auto auto;

/* En el primer div */
grid-column: 1 / span 2;
```



Creando áreas con nombres

Otra opción de definir nuestra rejilla es usando la propiedad **grid-template-areas** en las que indicaremos, mediante cadenas de texto, el diseño de esta. Pondremos varias cadenas de texto entre comillas dobles, donde escribiremos una o varias letras con el diseño. Luego usaremos la propiedad **grid-area** en los elementos para indicar qué posición ocuparán.

Lo mejor es verlo mediante un ejemplo.

```
/* En el contenedor */
grid-template-areas:
  "a a b"
  "c d b"
  "e f b";

/* En el primer div */
grid-area: a;

/* En el segundo div */
```



```
grid-area: c;
```

```
/* En el tercer div */  
grid-area: b;
```

```
/* En el cuarto div */  
grid-area: d;
```

Como al quinto y sexto div no les hemos indicado nada, se colocan donde haya sitio

En lugar de usar a, b, c, ... podemos usar palabras para dejar más claro el diseño, como menú, contenido, encabezado,

Junto con esta propiedad podemos seguir usando grid-template-rows y grid-template-columns para indicar los tamaños de las filas y columnas.

Cambiar como se colocan los elementos restantes

Si queremos que los demás elementos se coloquen en filas y no en columnas, usaremos la propiedad **grid-auto-flow** con el valor column.

```
.contenedor {  
  display: grid;  
  grid-template-columns: 100px 200px;  
  grid-auto-flow: column;  
  outline: solid;  
}
```

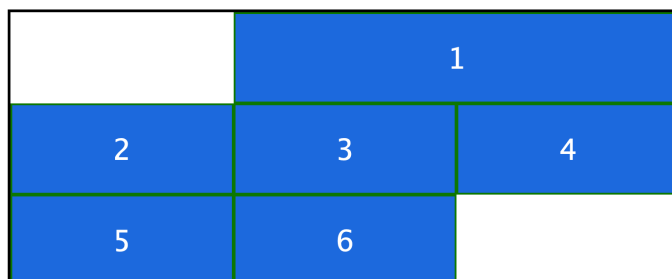


También disponemos de la opción dense, row-dense y column-dense, para intentar rellenar los vacíos que queden, si un elemento cabe en los mismos. Esto puede hacer que los elementos aparezcan desordenados.

Ejemplos:

```
/* En el contenedor */  
grid-template-columns: auto auto auto;
```

```
/* En el primer div */  
grid-column: 2 / 4;
```



```
/* En el contenedor*/
/* En el primer div */
grid-column: 2 / 4;
```

2	1	
3	4	5
6		

Alinear los elementos horizontalmente en su celda

Podemos usar la propiedad **justify-items** en el contenedor para ello, con los valores start(inicio), center (centrado), end (final) y stretch (que ocupe más que su contenido), siendo este último el valor por defecto.

Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
```

```
/* En el contenedor */
justify-items: end;
```

```
/* En el contenedor */
justify-items: start;
```

	1		2		3
	4		5		6

1		2		3	
4		5		6	

Si queremos que algún elemento tenga una alineación diferente, podemos usar la propiedad **justify-self** en este.

Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
```

```
/* En el contenedor */
justify-items: start;
/* En el segundo div */
justify-self: end;
```

```
/* Ninguna alineación en el contenedor */
/* En el segundo div */
justify-self: end;
```

1				2	3	
4				5	6	

1			2	3	
4		5		6	

Alinear los elementos verticalmente en su celda

Podemos usar la propiedad **align-items** en el contenedor para ello, con los valores start(inicio), center (centrado), end (final) , baseline (línea base) y stretch (que ocupe más que su contenido), siendo este último el valor por defecto.

Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
grid-template-rows: 130px 130px; /* Ya que en nuestro ejemplo todas las columnas tienen el mismo alto,
debemos dar uno mayor para que podamos ver el efecto en los ejemplos */
```

<pre>/* En el contenedor */ align-items: start;</pre>	<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td colspan="3"></td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td colspan="3"></td></tr></table>	1	2	3				4	5	6			
1	2	3											
4	5	6											
<pre>/* En el contenedor */ align-items: end;</pre>	<table><tr><td colspan="3"></td></tr><tr><td>1</td><td>2</td><td>3</td></tr><tr><td colspan="3"></td></tr><tr><td>4</td><td>5</td><td>6</td></tr></table>				1	2	3				4	5	6
1	2	3											
4	5	6											

Si queremos que algún elemento tenga una alineación diferente, podemos usar la propiedad **align-self** en este.

Ejemplos:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
grid-template-rows: 130px 130px;
/* En el contenedor */
align-items: start;
/* En el segundo div */
align-self: end;
```

1	2	3
4	5	6

```
/* Ninguna alineación en el contenedor */
/* En el segundo div */
align-self: end;
```

1	2	3
4	5	6

Disponemos de la propiedad **place-self** para indicar las propiedades align-self y justify-self al mismo tiempo. Le podemos asignar un valor, con lo que será un valor para ambas propiedades, o dos valores, primero para align-self y luego para justify-self.

Alinear los elementos horizontal y verticalmente en su celda

Podemos usar la propiedad **place-items** para combinar las propiedades justify-items y align-items en una sola. Le podemos asignar un valor, con lo que será un valor para ambas propiedades, o dos valores, primero para align-items y luego para justify-items.

Ejemplo:


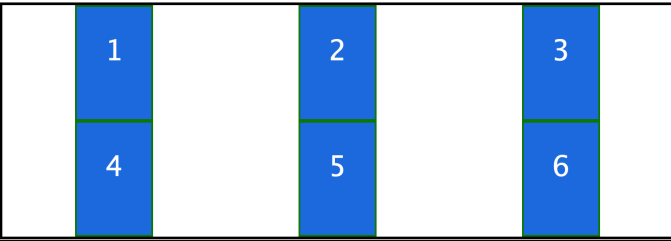
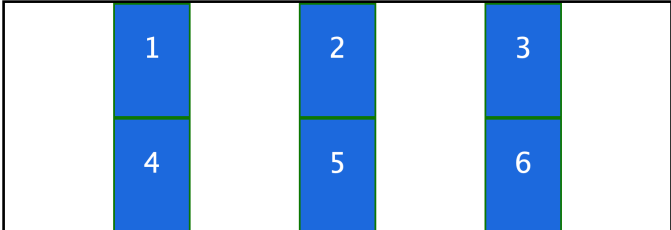
```
place-items: start end;
```

Alinear todas las celdas horizontalmente dentro de la rejilla

Con la propiedad justify-items alineamos horizontalmente el contenido dentro de la celda, con **justify-content** alinearemos todas las celdas dentro de la rejilla, si está es más grandes. Admite los valores start, end, center, stretch (valor por defecto), space-around (con espacio a los lado e interior, siendo a los lados más pequeño), space-between (sin espacio a los lados y el mismo espacio interior) y space-evenly (mismo espacio en todos los lados) que vimos en Flexbox.

Ejemplo:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
```

/* En el contenedor */ justify-content: start;	
/* En el contenedor */ justify-content: space-around;	
/* En el contenedor */ justify-content: space-evenly;	

Alinear todas las celdas verticalmente dentro de la rejilla

Con la propiedad align-items alineamos verticalmente el contenido dentro de la celda, con **align-content** alinearemos todas las celdas dentro de la rejilla, si está es más grandes. Admite los valores start, end, center, stretch (valor por defecto), space-around (con espacio a los lado e interior, siendo a los lados más pequeño), space-between (sin espacio a los lados y el mismo espacio interior) y space-evenly (mismo espacio en todos los lados) que vimos en Flexbox.

Ejemplo:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
height: 300px; /* Para que podamos ver los efectos de esta propiedad */
```

```
/* En el contenedor */
align-content: start;
```

1	2	3
4	5	6

```
/* En el contenedor */
align-content: space-around;
```

1	2	3
4	5	6

```
/* En el contenedor */
align-content: space-evenly;
```

1	2	3
4	5	6

Alinear todas las celdas horizontal y verticalmente en la rejilla

Podemos usar la propiedad **place-content** para combinar las propiedades justify-content y align-content en una sola. Le podemos asignar un valor, con lo que será un valor para ambas propiedades, o dos valores, primero para align-content y luego para justify-content.

Ejemplo:

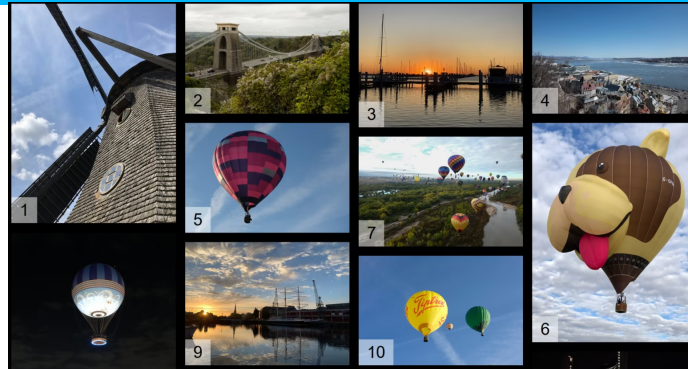
```
place-items: start end;
```

Anidaciones

Podemos tener un diseño Grid o Flexbox dentro de un Grid, un Grid dentro de un Flexbox, ...

Masonry / Albañilería

Aunque tengamos elementos de diferente alto o ancho, todas las celdas de una misma fila o columna ocupan lo mismo, con lo que no podemos conseguir algo como esto:



<https://www.smashingmagazine.com/native-css-masonry-layout-css-grid/>

Pero en la especificación de módulo Grid Level 3 de noviembre de 2023 se ha añadido la opción **masonry** a grid-template-rows y grid-template-columns, con lo que casi ningún navegador la soporta.

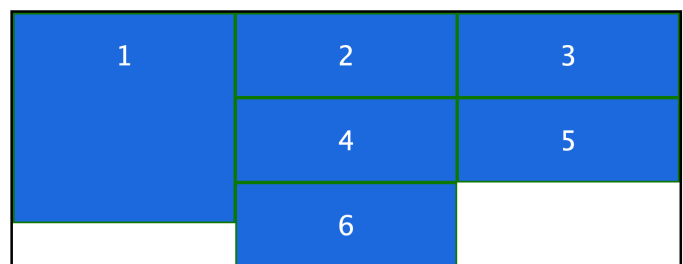
En Firefox podemos hacerlo con:

- poniendo about:config en la barra de direcciones:
- buscando layout.css.grid-template-masonry-value.enabled y poniéndola a true

Ejemplo:

```
/* En el contenedor */
grid-template-columns: auto auto auto;
grid-template-rows: masonry;

/* En el primer div */
height: 180px;
```



Una alternativa hasta que esta opción esté activa es usar un diseño de columnas en lugar de una rejilla.

```
/* En el contenedor */
column-count: 3;
column-gap: 10px;

/* En el primer div */
height: 180px;
```

