silocreativo.com

Introducción y primeros pasos con CSS Grid (I) • SiloCreativo

Alfonso Serrano

7-9 minutos

Cuando empecé en el mundo del diseño web, uno de los aspectos más frustrantes que me encontré fue el relacionado con todo aquello de **reordenar, colocar y distribuir diferentes elementos dentro de una web**. Los floats me resultaban tremendamente complejos y confusos. Y el descubrir el sistema Flexbox fue como un soplo de aire fresco, aunque seguía teniendo ciertas limitaciones. Por suerte, desde hace unos meses **los navegadores más usados tienen activado de forma predeterminada CSS Grid**. Así que es el momento idóneo para empezar a usar este módulo CSS en nuestro flujo de trabajo diario.

¿Qué es CSS Grid Layout?

En primer lugar, hay que dejar claro que **CSS Grid no es algo que venga a sustituir a Flexbox**. De hecho, si queremos controlar de forma precisa la maquetación de nuestra web, es imprescindible conocer ambos, y usarlos indistintamente según nuestras necesidades.

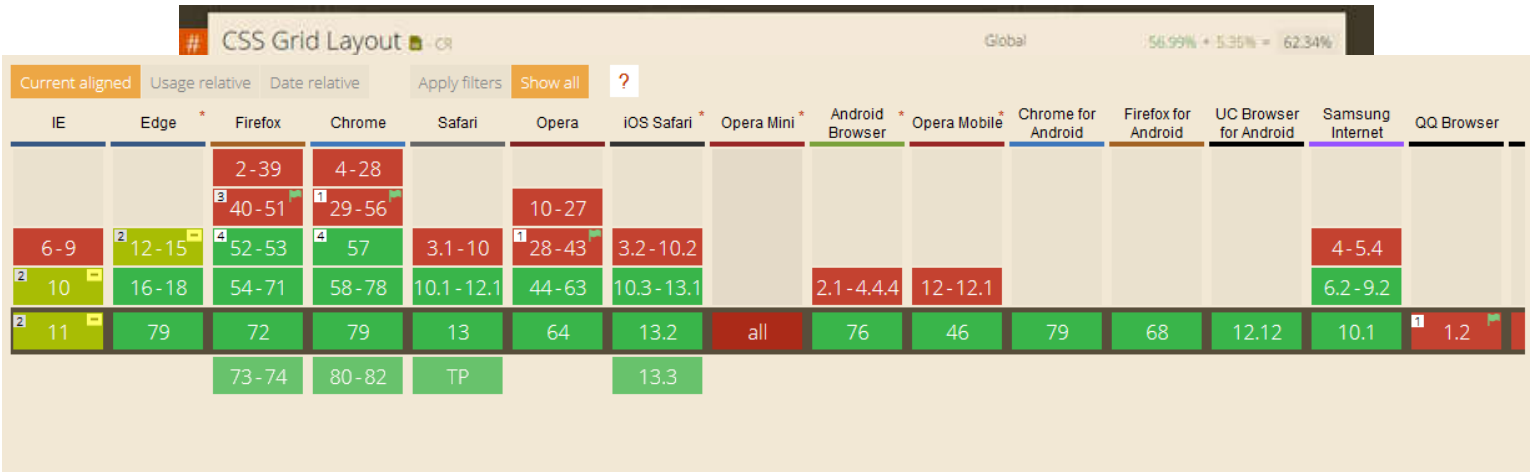
La principal idea tras CSS Grid es la de **dividir una página web en columnas y filas**. Además, añadiendo la posibilidad de posicionar

cada elemento en base a estas filas y columnas que hemos creado, en términos de tamaño, posición y orden.

Una de las ventajas, es que esta rejilla o *grid* que creamos nos permite **cambiar la posición de los elementos modificando simplemente nuestro código CSS, sin tener que realizar cambio alguno en HTML**.

Soporte en Navegadores Web

Antes de entrar en materia, es necesario conocer el [soporte que ofrecen los navegadores web](), y como podemos habilitar esta característica.



Desde principios Marzo del 2017, **Google Chrome** y **Mozilla Firefox** tienen activado por defecto CSS Grid Layout. De hecho, son estos dos navegadores los que os recomendamos que uséis en vuestras pruebas si estáis recién iniciados en este módulo. Y a finales de ese mismo mes lo añadieron de forma nativa otros navegadores, como **Safari**, **Opera** o **Chrome para Android**.

Sobre **Microsoft Edge** y **Internet Explorer**, tiene un soporte parcial añadiendo el prefijo *-ms-*. Es interesante saber que la primera propuesta de Grid Layout fue desarrollada por Microsoft, y el primer navegador en soportarlo fue Internet Explorer 10.

Conceptos y Terminología básica

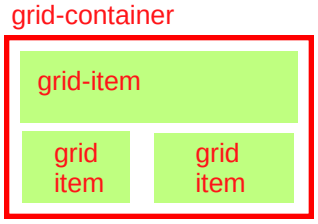
Como hemos marcado anteriormente, **CSS Grid no es un módulo que sustituye a Flexbox**. Por ello, es muy conveniente que dominemos o al menos conozcamos los conceptos básicos de Flexbox, antes de entrar de lleno en CSS Grid. Y por supuesto, también tenemos que estar familiarizados con el uso de CSS.

Tal y como ocurre en Flexbox, CSS Grid funciona con **la idea de un contenedor-padre que alberga unos elementos-hijo**. Por lo que nuestro contenedor-padre es un *grid-container*, y nuestro elemento-hijo es un *grid-item*.

Grid Container / Grid Item

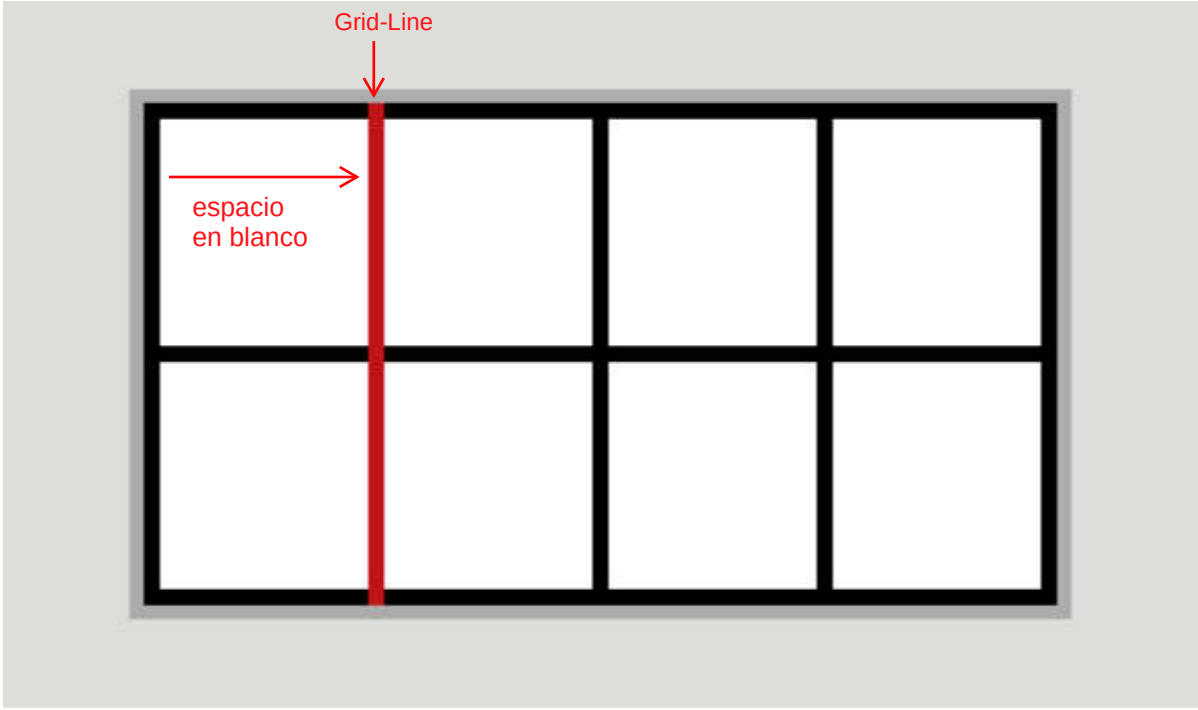
Grid Container es el **elemento padre**, que define nuestra cuadrícula o rejilla. Es donde aplicaremos el *display: grid*.

Grid Item es cada uno de los **elementos hijo** que contiene la cuadrícula o rejilla.



Grid Line

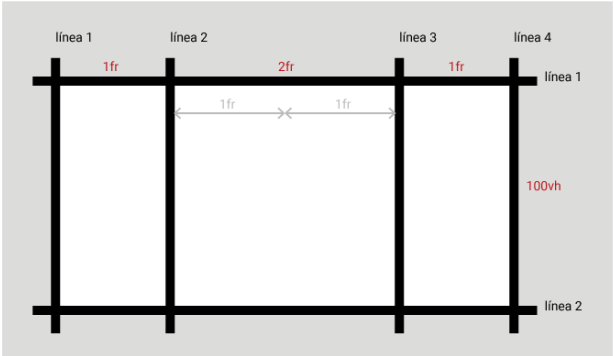
Cada línea divisoria que crean la estructura de la rejilla. Pueden ser tanto verticales (*column grid lines*) como horizontales (*row grid lines*).

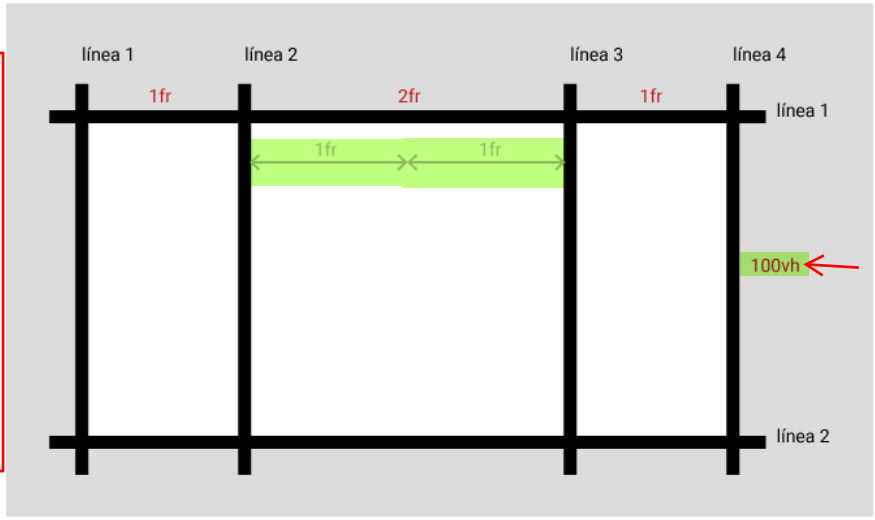


Los espacios que nos quedan entre cada línea nos van a servir para explicar un valor que usaremos en nuestro el escribir nuestro código CSS. Por un lado, **el valor *fr* o *fraction*, es la fracción de espacio restante en el grid**. Por ejemplo, si tenemos tres columnas, y a cada una de ellas le asignamos un valor de 1fr, se repartirán por igual el ancho total del contenedor. Si por el contrario, a una de esas tres columnas le damos un valor de 2fr, ocupará el doble de ancho que las columnas a las que le hemos asignado el valor 1fr. Este valor se podrá asignar tanto a columnas como a filas.

viewheight

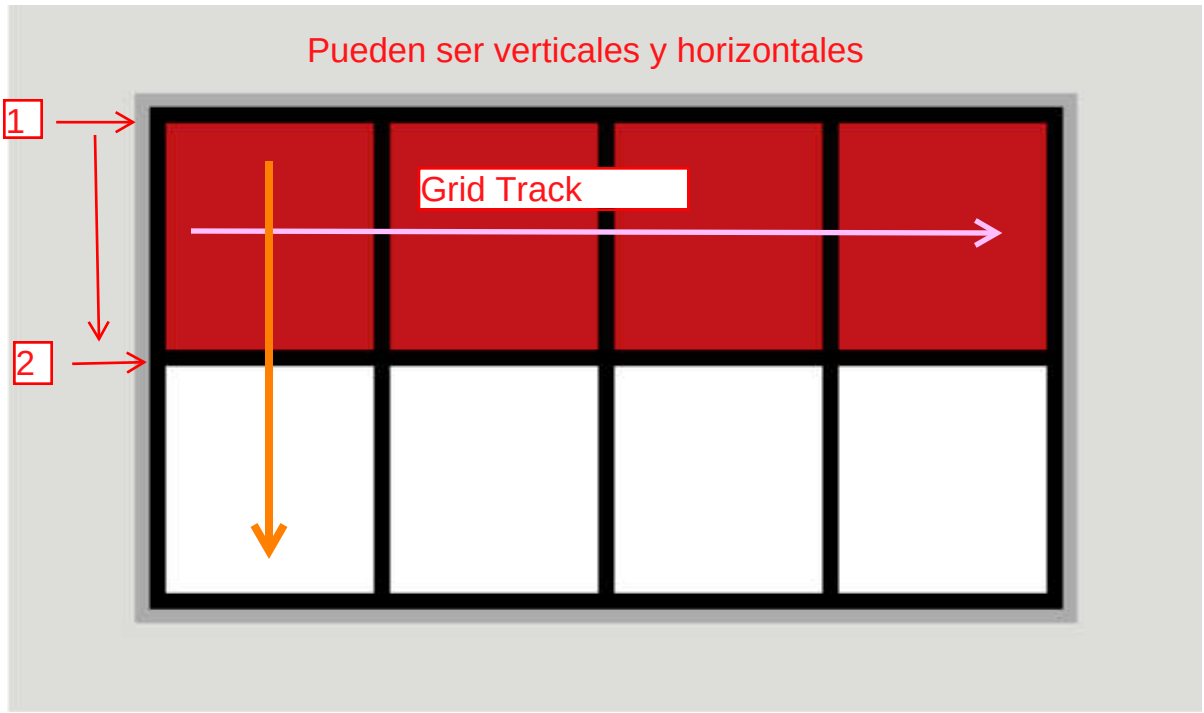
Por otra parte, **el valor *vh* define la altura de la fila**. Si le aplicamos a una fila un valor de 100vh, ocupará el 100%de la altura del contenedor.





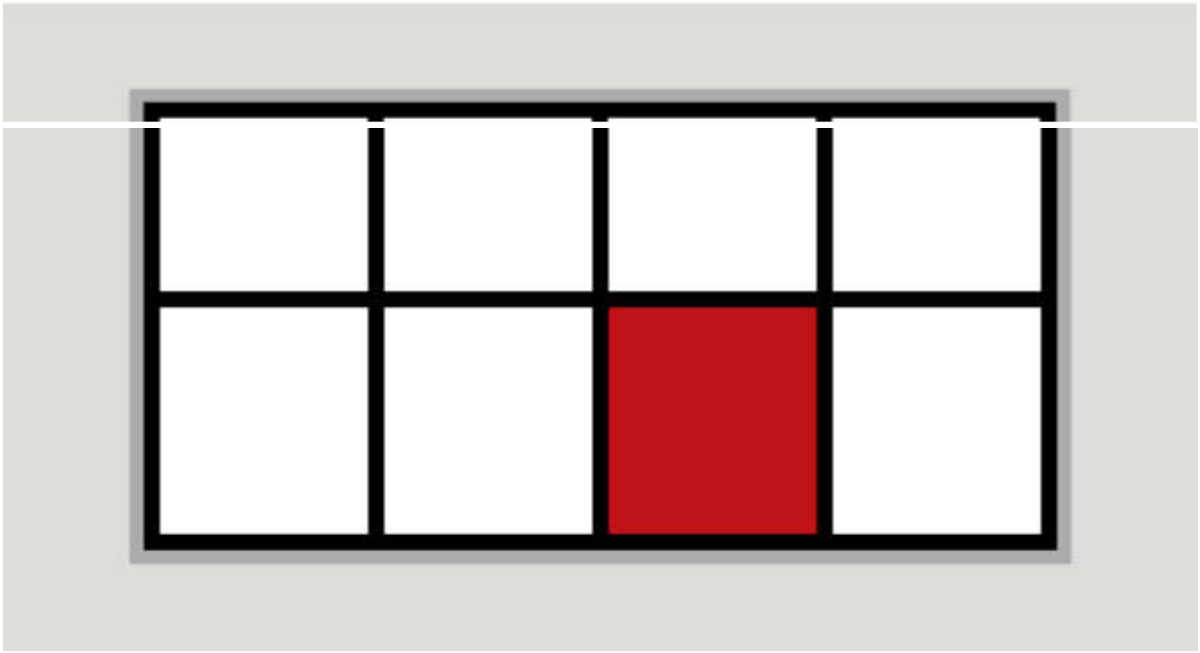
Grid Track

El espacio entre dos *grid lines* adyacentes. Pueden formar tanto columnas como filas. En la imagen vemos el *grid track* entre la primera línea de fila y la segunda.



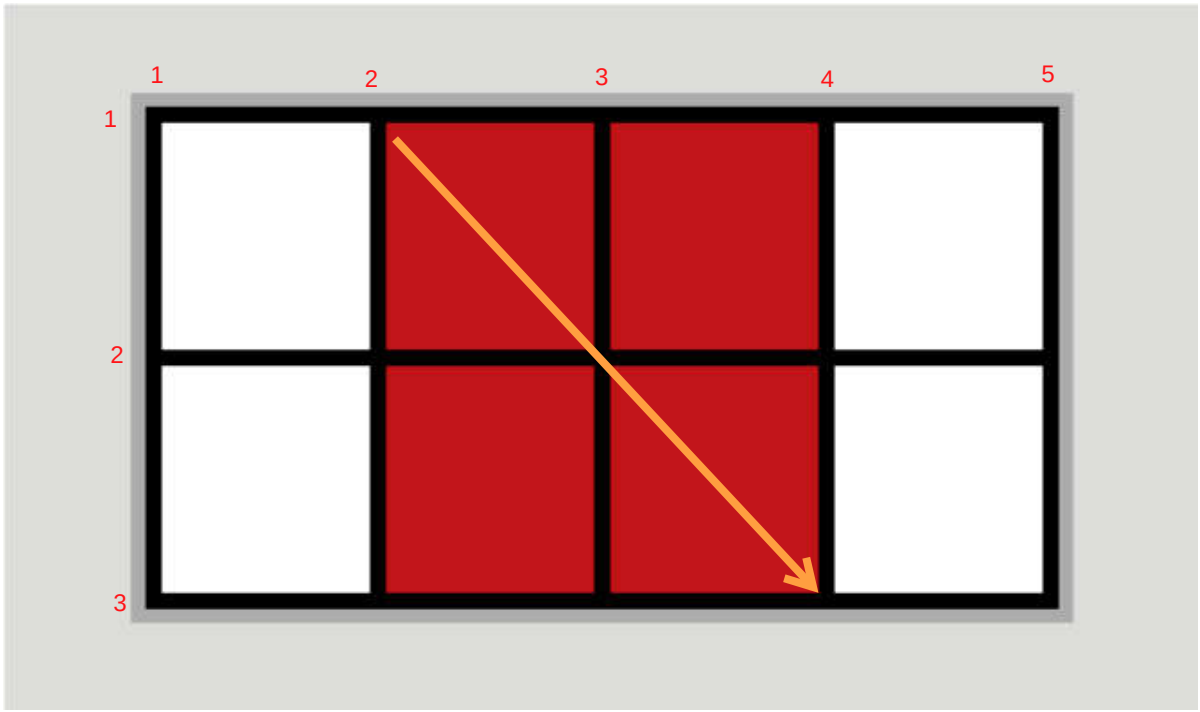
Grid Cell

El espacio entre dos lineas verticales adyacentes y dos líneas horizontales adyacentes. Es decir, sería una «unidad» dentro de la rejilla.



Grid Area

El espacio entre dos líneas verticales y dos líneas horizontales, no teniendo que ser forzosamente adyacentes. En la imagen, tendríamos la *grid area* creada entre las *column grid lines* 2 y 4, y las *row grid lines* 1 y 3.



Construir un grid o rejilla con filas y columnas

Vamos a hacer dos pequeños ejemplos para poner en marcha los conceptos que llevamos hasta ahora. En el primero, usaremos fracciones del espacio libre de la rejilla y alturas de filas, mientras que el el segundo usaremos pixeles y porcentajes. Así practicamos ambos, y no nos dejamos nada detrás. **Nuestra una rejilla esta compuesta por cuatro columnas y dos filas**, y usaremos como base el siguiente código HTML. No lo hemos mencionado antes, pero también es esencial saber escribir en lenguaje HTML.

```
</div class="container">
  </div class="item-1"> </div>
  </div class="item-2"> </div>
  </div class="item-3"> </div>
  </div class="item-4"> </div>
  </div class="item-5"> </div>
  </div class="item-6"> </div>
  </div class="item-7"> </div>
  </div class="item-8"> </div>
</div>
```

fila 1

fila 2

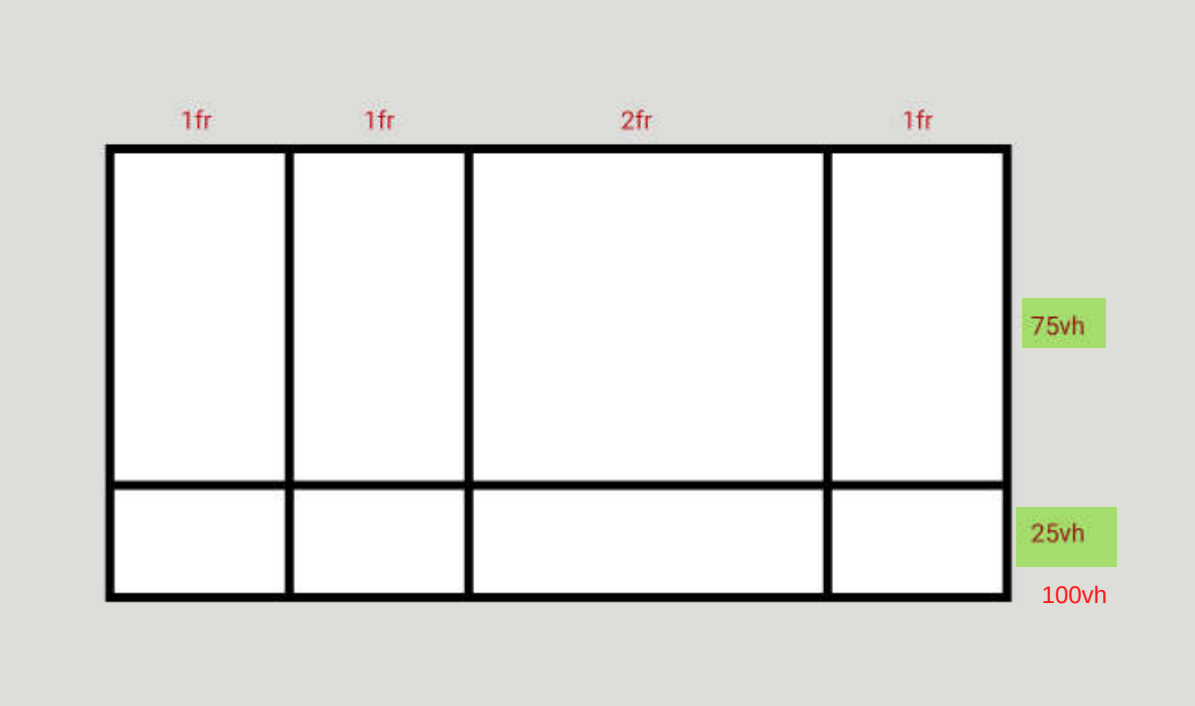
columna 1

columna 2

columna 3

columna 4

Como hemos dicho anteriormente, **vamos a crear una rejilla usando solo fracciones y alturas de filas**. La imagen a continuación nos muestra de manera más clara que es lo que vamos buscando.



Pues el proceso no puede ser más sencillo. En CSS, asignamos un *display: grid* a nuestro contenedor. **No olvidemos nunca dar este primer paso**, ya que de lo contrario, el navegador «no sabrá» que queremos usar CSS Grid Layout continuación especificar la anchura de cada columna mediante un *grid-template-columns*, mediante cuatro valores, uno por columna. Y con un *grid-template-rows*, la altura de cada fila mediante dos valores. Nos quedaría algo parecido a esto:

```
.container{
    display: grid;
    grid-template-columns: 1fr 1fr 2fr 1fr;
    grid-template-rows: 75vh 25vh;
}
```

Como veis, hacerlo más rápido se antoja complicado. No podemos pedir más.

Ahora vamos a crear otra rejilla, pero en este caso **vamos a utilizar valores que no son más comunes: pixeles, porcentajes,** auto

auto… Seguiremos usando el mismo código HTML base, aunque nuestra estructura de lo que queremos crear cambiará un poco. En esta imagen podemos ver cual es el resultado que queremos.

Seguimos **el mismo flujo de trabajo que la vez anterior**. Primero, *display: grid* en el container para que el navegador sepa en que módulo estamos trabajando. Y a continuación, el valor de cada columna y cada fila detras de su *grid-template-columns* y *grid-template-rows* correspondiente. Este es el código CSS resultante.

```
.container{
   display: grid;
   grid-template-columns: 30px auto 20px 25%;
   grid-template-rows: 50px auto;
}
```

Y como en el caso anterior, obtenemos un código muy limpio y que se lee perfectamente de un golpe de vista. ¡Fantástico!

Espaciado entre items

Aunque queda un aspecto por resolver, que seguro que alguno habréis notado. El espacio entre los **grid items** es nulo, no existe un espacio vacío entre unos y otros. Y a veces necesitamos ese **espaciado entre nuestros elementos de la rejilla**. Este problema lo solucionaremos con dos propiedades: ***grid-row-gap*, que es el espacio entre filas**, y ***grid-column-gap*, que es el espacio entre columnas**. Y con la propiedad *grid-gap* le daremos espacio tanto a filas como a columnas. Cuando lo usemos, hay que tener en cuenta una cosa muy importante. el primer *grid gap* se corresponde con la segunda *grid line*, que es la que realmente está entre dos elementos de nuestra rejilla. Podeis usar estas propiedades en el código CSS que hemos estado usando antes, y seguir jugando con la rejilla que hemos creados.

Esperemos que os haya gustado esta pequeña introducción a CSS Grid. **En el siguiente artículo seguiremos profundizando en él, y llevaremos a cabo rejillas más complejas** que nos permitan resolver prácticamente cualquier diseño que nos propongamos.