Manual de CSS Grid Layout

Diana Aceves
Miguel Angel Alvarez

desarrolloweb.com

desarrolloweb.com/manuales/manual-css-grid-layout

Introducción: Manual de CSS Grid Layout

El Manual de CSS Grid Layout te explica una de las nuevas utilidades de CSS, que ya ha revolucionado la manera con la que se crean las páginas web. CSS Grid, o Grid Layout es un nuevo estándar de las Hojas de Estilo en Cascada que nos permite maquetar contenido ajustándolo a una rejilla totalmente configurable mediante estilos CSS. Gracias a CSS Grid Layout podrás colocar elementos dentro de la página como nunca antes habías podido, independientemente de cómo aparecen en el código HTML y de manera extremadamente versátil, ya que podemos cambiar radicalmente el aspecto y disposición del contenido únicamente cambiando el código CSS. Por supuesto, en combinación con las media queries podrás además adaptar la rejilla y la posición de los elementos de la página en función de las condiciones del usuario, el tamaño de su pantalla, etc. El manual de Grid Layout introducirá los conceptos de necesario conocimiento para poder sacar partido a esta herramienta de las CSS, y luego aportará diversas prácticas para que veas cómo usar CSS Grid en ejemplos de la vida real.

Encuentras este manual online en:
http://desarrolloweb.com/manuales/manual-css-grid-layout

Autores del manual

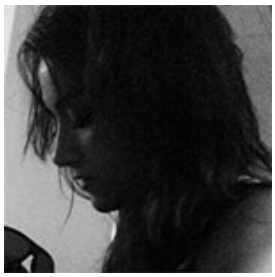
Las siguientes personas han participado como autores escribiendo artículos de este manual.

Miguel Angel Alvarez

Miguel es fundador de DesarrolloWeb.com y la plataforma de formación online EscuelaIT. Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.

Diana Aceves

Licenciada en química, Diana comenzó con la programación con Java y finalmente su vocación la llevó a convertirse en desarrolladora front-end



Qué es CSS Grid Layout

Conoce CSS Grid Layout, el sistema de rejilla para la maquetación web, estándar y compatible en todos los navegadores. Esta nueva especificación de CSS3 te permitirá llegar a donde habías pensado que era imposible.

CSS Grid Layout es un sistema de rejilla en 2 dimensiones, creado dentro del lenguaje CSS. Es un estándar, lo que quiere decir que no necesitas nada para que el navegador lo entienda. Para ponerlo en práctica necesitas simplemente aplicar el nuevo "display: grid" y comenzar a usar sus potentes propiedades y valores CSS. Con ello podrás controlar todos los aspectos imaginables de la rejilla, obteniendo prácticamente todo lo que se te ocurra de una manera cómoda.

Actualmente todos los navegadores soportan CSS Grid Layout, por lo que te recomendamos usarlo. Aplicar el sistema de rejilla es una auténtica maravilla, permitiendo hacer con muy poco esfuerzo cosas que antes necesitabas mucho código CSS, o que directamente eran imposibles de conseguir, al menos con tal versatilidad.

Este artículo comieza el Manual de CSS Grid Layout con una introducción general, en la que te explicaremos qué es Grid Layout, cómo empezar ya mismo a aprender este nuevo modelo de maquetación y cómo aplicarlo en tu trabajo del día a día.

Sistemas de rejilla en CSS

CSS Grid Layout es un modelo de maquetación CSS en base a una rejilla, algo que podría no parecer tan novedoso, si tenemos en cuenta que diversas librerías de CSS lo habían intentado ya. Sistemas como 960 Grid System fueron pioneros en crear una base de código CSS para que los diseñadores pudieran posicionar los elementos en una distribución de filas y columnas. El propio Bootstrap incluye entre otras cosas un sistema de rejilla.

Al final, estos sistemas funcionaron pero tenían varios problemas. Entre ellos destacamos:

- Tenías que agregar peso a tu CSS, con código de cientos de clases que muchas veces ni siquiera llegabas a usar.
- Tenías que aplicar constantemente clases con nombres raros, que ensuciaban tu HTML de manera muy agresiva.
- Creaba código de muy difícil mantenimiento, haciendo que el proyecto estuviera ligado de manera muy directa con un sistema propietario. Cuando el desarrollador tenía que asumir el código de una web creado con un sistema de rejilla, hacía muy difícil su adaptación.

La diferencia ahora es que Grid Layout es un estándar y que no necesitas recargar tu página y tu código CSS para aplicarlo. No necesitas ensuciar tu HTML, ya que todo se aplica desde el CSS, por medio de nuevos atributos, valores y unidades. Además, un sitio con Grid Layout resulta sencillo de mantener y de aplicar otras herramientas de CSS3 como las mediaqueries. Por supuesto, es totalmente adaptable (responsive).

Dónde se pueden aplicar los sistemas de rejilla

Otra pregunta típica que nos hacemos al introducirnos en los sistemas de rejilla y en concreto con el nuevo estándar de CSS es ¿Dónde lo puedo aplicar? Realmente, tienes que pensar que es CSS y por tanto lo puedes aplicar en todos los sitios donde aplicas actualmente código CSS.

El sistema de rejilla no es algo que puedas usar sólo en sitios estáticos, en programación con un lenguaje determinado, en un framework o en un tipo de dispositivos. Lo puedes usar en todos esos sitios, y es aplicable a la web para ordenadores de escritorio, para dispositivos con pantallas pequeñas y para el desarrollo de webapps, como las que haces con Ionic. No hay límites para usarlo. Donde puedas desarrollar tus CSS para definir cualquier estilo, puedes usar Grid Layout para aplicar una rejilla.

Compatibilidad de navegadores de Grid Layout

Al aparecer un nuevo estándar lo primero que nos preguntamos es ¿lo puedo usar ya? ¿los navegadores son compatibles con él? Afortunadamente, la tecnología está disponible ya en todos los navegadores del mercado. Tenemos luz verde para empezar a beneficiarnos de este nuevo modelo de maquetación.

Por tanto, no hay motivo para esperar más. Usa Grid Layout y tu vida será más agradable a partir del primer minuto. Podrás hacer en instantes cosas que antes tenías que sufrir para conseguir.

Esta imagen corresponde con el estado de CSS Grid Layout en el panorama de navegadores del mercado, en septiembre de 2017. Sacada de Caniuse.com.

Lo puedes usar en todos los navegadores (excepto Opera Mini), incluso en IE 11 (con uso de prefijos CSS) o Edge.

Para los navegadores más antiguos todavía existen polyfills (fallbacks) capaces de aplicar soporte por medio de CSS o Javascript.

> **Nota:** No es que sea recomendable aplicar polyfills en todas las situaciones. Si nos toca dar soporte a navegadores tan antiguos quizás sería más interesante seguir una estrategia de "graceful degradation", aplicando estilos alternativos para esos navegadores. Es decir, no aplicaríamos la rejilla y el contenido estaría visible, pero no dispuesto en las filas y columnas que representarían el resto de navegadores actuales.

En resumen, ya puedes usar CSS Grid Layout. Es una especificación estable y con soporte en los principales navegadores. Cuando veas lo sencillo que puede llegar a ser tu trabajo con una rejilla estándar se te caerán las lágrimas de alegría. En resumen, si no usas CSS Grid Layout, lo que estás perdiendo es tu tiempo.

Comparación con Flexbox

Existe otro estándar para la maquetación con grandes posibilidades, llamado Flexbox. Quizás hayas comenzado a usar Flexbox, ya que lleva más tiempo con nosotros. Grid Layout va un paso más allá, ya que aplica a dos dimensiones mientras que Flexbox aplica a sólo una.

Con Flexbox sólo podías definir qué es lo que tenían que hacer los items en el eje horizontal o en el eje vertical. Cuando en el eje vertical u horizontal se terminaba el espacio, entonces los items se iban colocando dependiendo de las configuración de los atributos Flexbox. Por ejemplo, en el caso de usar flex-wrap, cuando llegaba un item que no cabía en la horizontal, se iba a la siguiente fila. Pero tú no controlas esas dos filas, los propios ítems son los que se van acomodando.

La diferencia en CSS Grid es que tienes realmente dos dimensiones. Tus casillas podrás decidir dónde quieres que se pongan, tanto en la horizontal como en la vertical, coordinando perfectamente las dimensiones de cada fila o columna, de manera global o perfectamente

independiente. No necesitas hacer trucos, ni forzar que se queden sin sitio, realmente tienes el control detallado de la posición y dimensiones de cada elemento en cada eje.

Resumen de las posibilidades de CSS Grid

Grid Layout te ofrece varias novedades, con respecto a todo lo anterior existente en CSS. A poco que comiences a usar Grid Layout apreciarás la diferencia y encontrarás un sin fin de variantes que antes no disponías para posicionar de elementos. Un resumen de las novedades más importantes sería este:

- Puedes colocar los ítem donde quieras, en cualquiera de las celdas que el grid describe, incluso en celdas que no has descrito
- La posición de los elementos en la página puede ser totalmente distinta a cómo aparecen ordenados en el código HTML
- Pueden haber ítems que se coloquen solos (auto-placement)
- Controlas las dos dimensiones
- La colocación de los ítems es libre, no es una tabla
- Tiene una variada sintaxis y nuevas unidades, atributos y funciones que te ofrecen nuevas posibilidades.
- Ofrece una cantidad amplia de nuevos atributos CSS para los elementos de la página. A veces permiten conseguir comportamientos similares y hay que estar atento para entender los distintos matices que los diferencian

Con todas estas novedades puede ocurrir que te despistes cuando empiezas a usarlo, pero lo cierto es que va a cambiar el CSS para siempre. Una vez que tengas asimilado lo fácil que es hacer layouts complejos, no querrás volver a las técnicas que usabas antes.

Clase de Grid Layout

Para continuar te ofrecemos una completa clase de Grid Layout. En 120 minutos podrás conocer lo básico de Grid Layout y apreciar algunas de sus increíbles posibilidades.

En esta clase conocerás los fundamentos de Grid Layout y comenzarás a ver ejemplos de aplicación de las principales propiedades existentes para implementar la rejilla y controlar el aspecto, posición y dimensiones de las celdas.

Gracias a este vídeo te ponemos muy fácil ponerte al día y agregar una nueva herramienta a tu kit de desarrollador. Es la primera clase del Curso de CSS Grid Layout de EscuelaIT, en el que tendremos 6 sesiones de formación como la que podrás ver a continuación.

> Para ver este vídeo es necesario visitar el artículo original en:
> https://desarrolloweb.com/articulos/que-es-css-grid-layout.html

En el próximo artículo de este manual abordaremos una descripción completa de los conceptos que debes conocer para empezar con CSS Grid Layout.

Este artículo es obra de *Diana Aceves*
Fue publicado / actualizado en *06/02/2020*
Disponible online en *http://desarrolloweb.com/articulos/que-es-css-grid-layout.html*

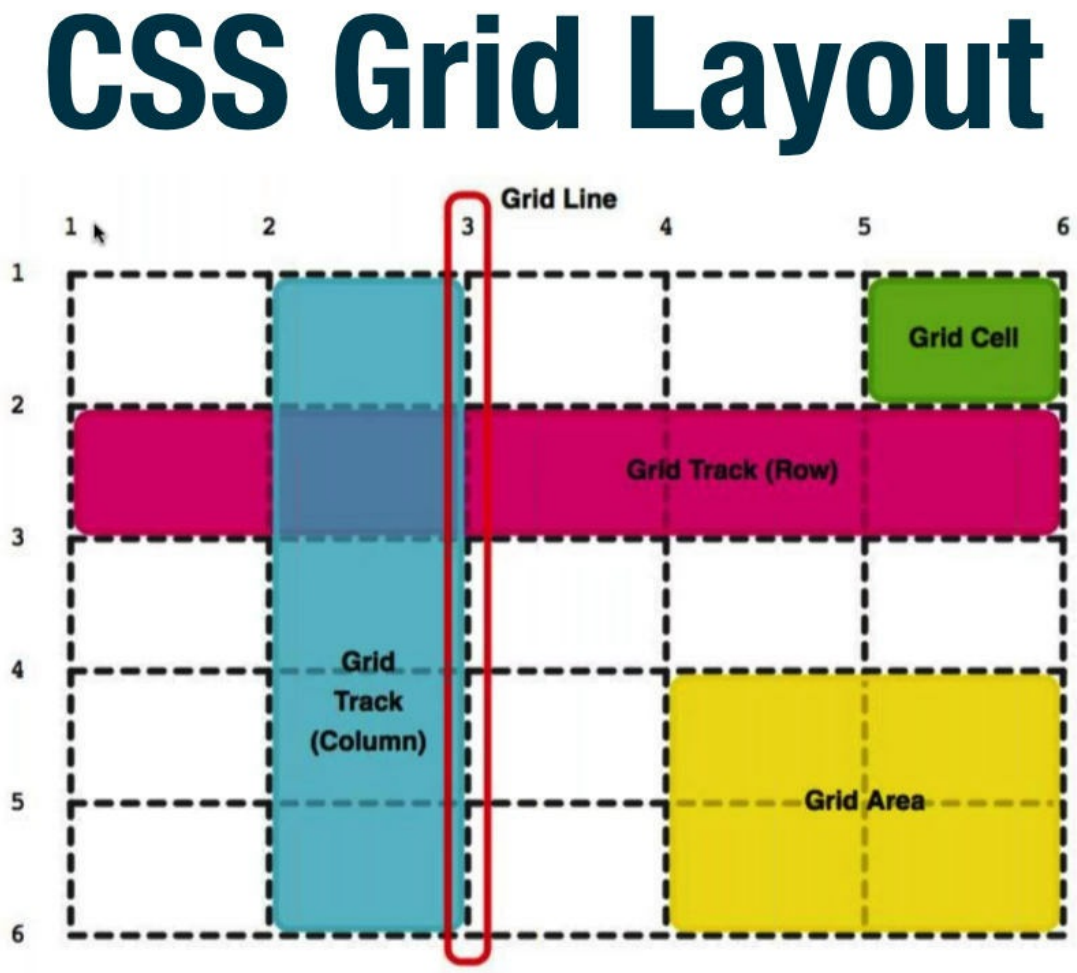
Conceptos básicos de CSS Grid

Hacemos un repaso a toda una serie de nuevos conceptos que tienes que entender para poder dominar el estándar de CSS Grid Layout y sacar partido al sistema de rejilla. Qué son líneas, celdas, áreas, columnas y demás.

En el artículo anterior dedicado a este estándar explicamos qué es CSS Grid Layout, qué problemas vino a resolver y sus características generales. En este artículo vamos a continuar conociendo este estándar CSS mediante el repaso de los conceptos básicos que debemos entender para comenzar a usarlo con soltura.

Es importante tener claros todos estos conceptos para que luego, cuando comencemos a estudiar la aplicación no tengamos dudas sobre qué es cada cosa que iremos nombrando. Así que ahí vamos!

Antes que nada, compartimos un diagrama con vosotros, donde podemos ver ilustrados todos los conceptos de los que vamos a hablar a continuación.



Este diagrama es de Diana Aceves, sacado de la primera clase del curso de CSS Grid de EscuelaIT, cuyo vídeo compartimos también en el anterior artículo de presentación de Grid Layout.

Líneas

Son las divisiones que permiten acotar las celdas. Tenemos tanto líneas horizontales que dividen las filas como líneas verticales que dividen las columnas.

En una rejilla, todas las líneas están numeradas o mejor dicho, tienen un número que comienza en la línea más exterior de la rejilla. Obviamente la primera línea será la 1 y aumentarán de izquierda a derecha y de arriba a abajo.

Las líneas son muy importantes, porque serán las que nos permitan definir la posición de los contenidos (elementos del HTML) dentro de la rejilla.

Tracks

Un track es lo que nosotros llamaríamos una columna o una fila. Una columna completa o fila completa. Por tanto un track horizontal irá de izquierda a derecha, ocupando una única fila completa y un track vertical de arriba a abajo, ocupando una única columna completa.

Los tracks están separados por dos líneas consecutivas. Sería lo que entendemos por una columna o una fila.

Son importantes porque nos ayudará a entender que CSS Grid no es una tabla de toda la vida (tablas de HTML), es algo inmensamente más flexible. De hecho, es incluso incorrecto denominar tracks horizontales y tracks verticales, porque con una configuración de la rejilla lo que eran tracks horizontales pueden pasar a ser verticales y viceversa. Por lo que podemos denominarlos simplemente tracks, tanto sean horizontales como verticales.

Celda

En CSS Grid una celda no tiene ninguna connotación especial con respecto a lo que ya podemos conocer. Es simplemente eso, un espacio definido entre dos líneas horizontales consecutivas y dos líneas verticales consecutivas. En la intersección de todas estas líneas se acota una celda.

Como puedes entender, una celda tiene el tamaño de 1x1 en nuestra rejilla.

Área

Es una porción de nuestra rejilla que comprende más de una celda. Si hemos dicho que una celda es de 1 x 1, un área puede ser un conjunto de celdas de 2 x 1, 2 x 2, etc.

Las áreas son siempre cuadradas o rectangulares y se componen por celdas que están consecutivas, ya sea en la horizontal, vertical o ambas. No puedo hacer un área en forma de "L", tiene que ser rectangular (o cuadrado).

Numeración dinámica de las líneas en CSS Grid

Es importante que conozcamos y dominemos todos los conceptos anteriores. No son tantos, por lo que estamos seguros que será fácil para ti. Antes de terminar queremos aportar alguna idea extra que nos haga entender lo flexible que puede llegar a ser el sistema de rejilla y lo fácil que es alterarlo con muy pocos atributos. De paso queremos hacer un primer ejemplo, ya que estamos seguros que más de un lector querrá ver ya algún código para trabajar con Grid Layout.

Nuestro ejemplo será sencillo y como estamos simplemente empezando vamos a saltarnos varias explicaciones que detallaremos más adelante. El objetivo en este caso será explicar cómo es la numeración de las líneas.

Vamos a trabajar con un código HTML en el que tenemos un contenedor que dentro tiene 6 elementos.

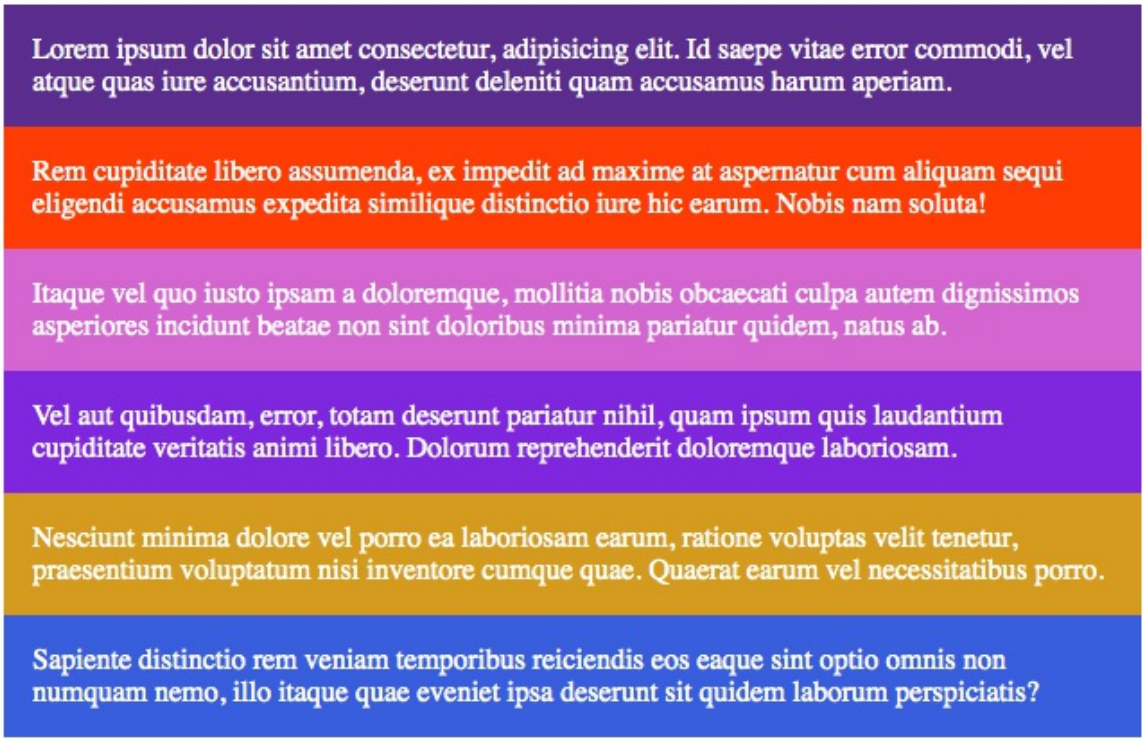
```
<div class="migrid">
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
```

```
  <div></div>
</div>
```

Para conseguir que un contenedor trabaje con el sistema de rejilla, simplemente tenemos que colocar "display: grid" en su CSS.

```
.migrid {
  display: grid;
}
```

Si no indicamos nada más, las celdas se colocarán una debajo de otra, tal como lo harían normalmente los elementos de bloque. Nos quedaría una distribución con la siguiente:



Como puedes ver, en este caso tenemos dos líneas verticales y 7 líneas horizontales.

Todas las líneas están numeradas. Las llamamos líneas indiferentemente que sirvan para dividir el espacio de la rejilla en la horizontal o en la vertical. Por tanto:

- La línea 1 es la vertical es la que va de arriba a abajo en la parte de la izquierda de la rejilla.
- La línea 1 en la horizontal es la que comienza la rejilla en la parte de arriba.

Siempre hay una línea más que el número de celdas. Si tenemos 1 columna, tendremos dos líneas verticales. Si tenemos 6 filas, entonces serán 7 líneas horizontales.

La numeración de las líneas, tanto en la horizontal como la vertical la realiza el propio navegador. Es automática, pero flexible como todo en la rejilla. Es decir, en la medida en la que

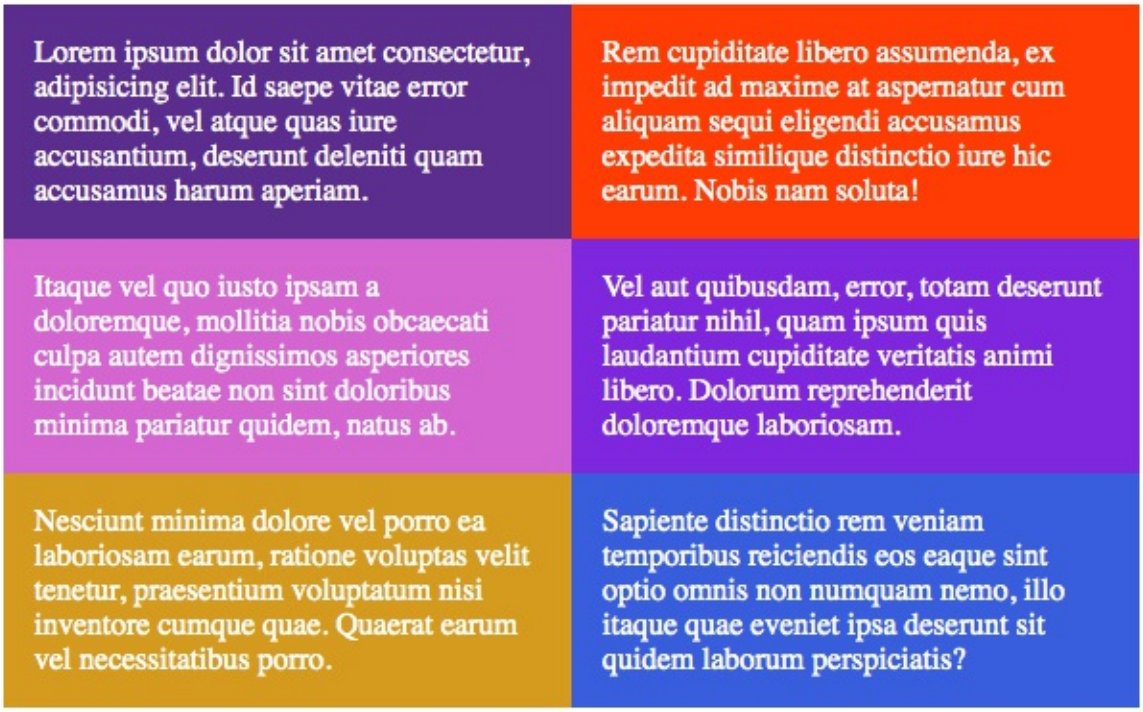
CSS Grid puede variar el número de columnas o de filas de la rejilla, en base a los atributos que tú definas, el número de líneas horizontales o verticales puede cambiar.

Aunque será tarea de estudio de próximos artículos, vamos a ver una de las maneras de definir la forma de nuestra rejilla, con el atributo "grid-template-columns". Este atributo indica la forma de nuestra rejilla atendiendo a cómo serán sus columnas.

```
.migrid {
  display: grid;
  grid-template-columns: 1fr 1fr;
}
```

Este valor indicará que habrá dos columnas y cada una de ellas tendrá 1fr. La unidad "fr" es nueva en CSS Grid y nos indica todo el espacio sobrante disponible. Solo que tenemos repetido dos veces el valor 1fr, por lo que ese espacio se tendrá que dividir a partes iguales entre las dos columnas. No te preocupes si no queda claro puesto que lo trataremos más adelante con detalle.

El caso es que, con nuestra nueva configuración, la rejilla resultante se verá como sigue:



En este caso tenemos dos columnas y, dado que tenemos 6 elementos, han salido 3 filas. Por tanto tendremos 3 líneas verticales y 4 líneas horizontales. No hace falta que las marquemos en la imagen, ya que seguramente las estarás visualizando perfectamente.

Ahora, con una tercera configuración de nuestro grid, vamos a cambiar nuevamente nuestro diseño.

```
.migrid {
```

```
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

En este caso tenemos 3 veces "1fr", por lo que serán tres columnas que se dividirán a partes iguales el espacio disponible.

En este caso podrás observar que tenemos 3 columnas y 2 filas. Por tanto tendremos 4 líneas verticales y 3 horizontales.

Habrás podido apreciar lo flexible que es nuestra rejilla y lo mucho que puede cambiar con una única configuración de atributo CSS. Esto es solo el principio. En los próximos artículos nos dedicaremos a hacer prácticas más interesantes que seguramente continúen sorprendiéndote.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado / actualizado en *21/01/2020*
Disponible online en *http://desarrolloweb.com/articulos/conceptos-basicos-css-grid*

Propiedades principales de CSS Grid Layout

Primeros ejemplos de rejilla creada con el estándar CSS Grid Layout, con explicaciones y prácticas de uso de los primeros atributos y valores que debes conocer para comenzar a usar el sistema de rejilla.

En artículos anteriores ya hemos podido conocer las generalidades del sistema de CSS Grid, así como los conceptos principales que debemos de conocer para entender correctamente este estándar de hojas de estilo en cascada.

Ahora vamos a trabajar sobre la práctica, abordando ya los atributos necesarios para trabajar con el sistema de rejilla en una página web.

Atributo display: grid

El primer paso para trabajar con el sistema de rejilla es incorporar el atributo "display: grid" en un contenedor. Desde el momento en que colocamos ese tipo de display en un contenedor, todos sus hijos directos pasan a ser considerados celdas dentro del sistema de rejilla.

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

Además, aunque no se use tanto, tenemos la posibilidad de usar otra variante de declaración del contenedor del grid, con el valor "inline-grid", que hace que la rejilla como un todo se comporte como un elemento inline.

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

El HTML necesario para el sistema de rejilla

El código HTML que necesitamos no tienen misterio alguno. Simplemente vamos a tener un contenedor. Ese contenedor tendrá una serie de hijos (nos referimos a hijos de primer nivel, ya que luego cada hijo podría tener su propio markup con otra serie de etiquetas dentro).

Un posible HTML para comenzar a trabajar sería este:

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

Realmente la clase "grid-container" es la que nos interesa, para aplicarle los estilos de grid, lo que repercute directamente en los hijos.

La clase "grid-element" en realidad no la necesitamos todavía, ya que con definir el CSS del contenedor ya habremos montado la rejilla, sin embargo, vamos a aplicar algunos estilos simplemente para que podamos visualizar convenientemente las celdas formadas por el grid.

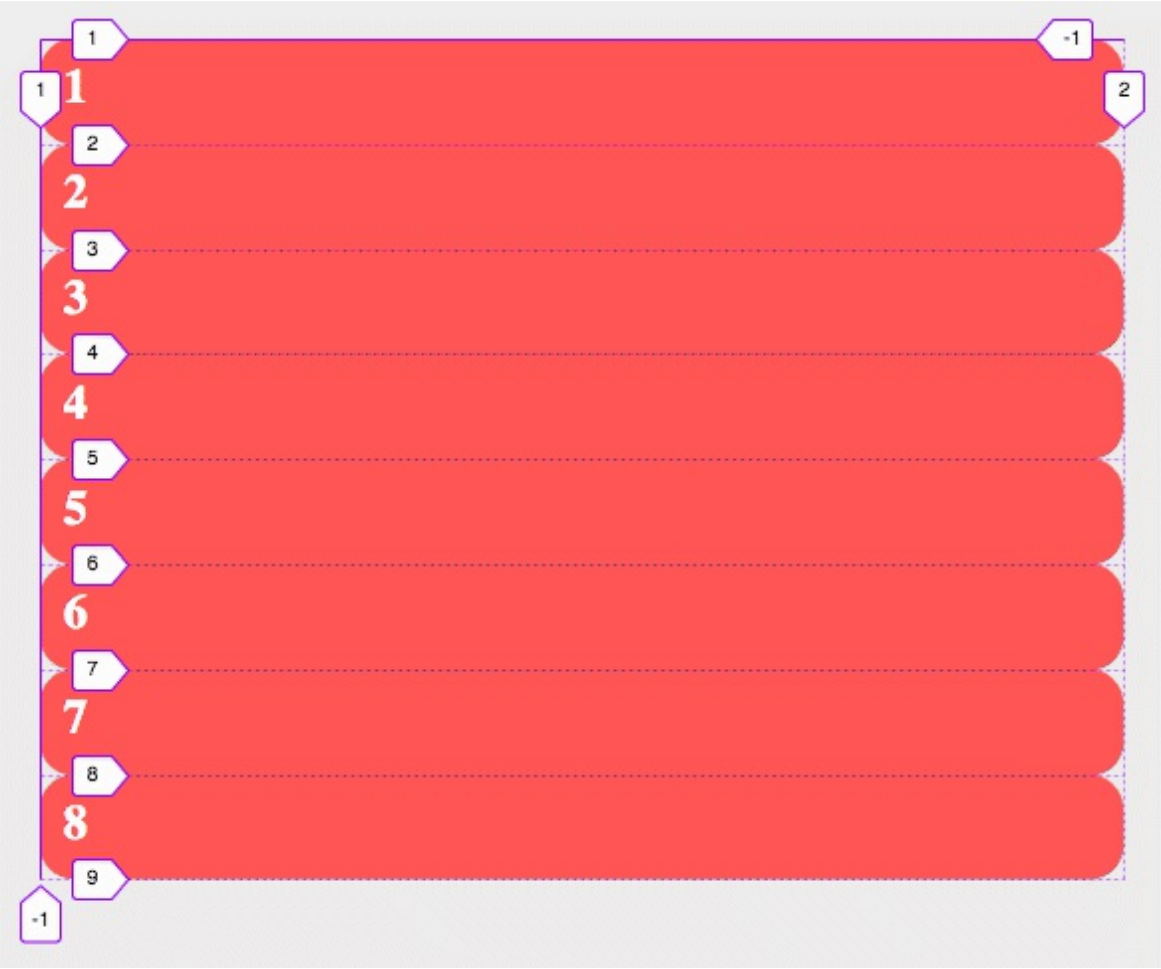
```
.grid-container {
  display: grid;
}

.grid-element {
  background-color: #f55;
  color: #fff;
  padding: 0.5em;
  font-weight: bold;
  font-size: 1.5em;
  border-radius: 0.7em;
}
```

Grid predeterminado

Como decíamos, el único atributo que necesitamos para trabajar con CSS Grid Layout es "display: grid". Solamente que esto nos genera un grid predeterminado, que a simple vista parece como si no hubiéramos hecho nada.

Es que el grid predeterminado, por llamarle de alguna manera, tiene una sola columna y tantas filas como elementos hijos tengamos en el contenedor, es decir, sería en la práctica el mismo comportamiento que el display: block.

Sin embargo, sí que hay una diferencia notable, aunque solo podremos percibirlo si inspeccionamos el elemento con las herramientas para desarrolladores de nuestro navegador.



En la anterior imagen podemos ver cómo esta rejilla se mostraría en Firefox Developer Edition, al abrir las herramientas de desarrolladores y activar la inspección de la rejilla.

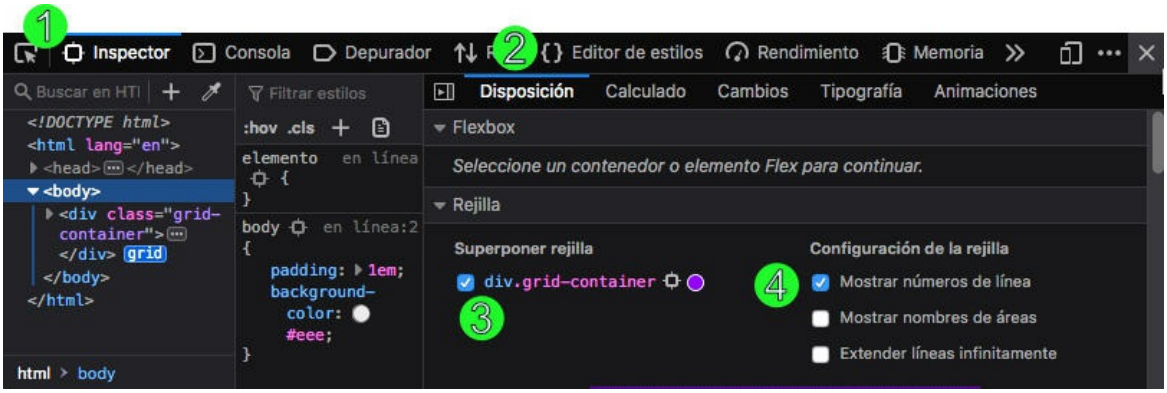
En la imagen podrás apreciar que el navegador me está mostrando los números de las líneas. Como tenemos 8 elementos, y se ponen uno debajo del otro, tenemos un total de 9 (de 1 a 9) líneas en la horizontal y 2 líneas en la vertical (1 a 2).

> **Nota:** deberías ya tener noción de lo que es una línea en el CSS grid, ya que lo explicamos en el anterior artículo sobre Conceptos de CSS Grid.

Para conseguir abrir este inspector de rejilla tenemos que hacerlo del siguiente modo:

- Abrimos las herramientas de desarrollo

- Nos situamos en la pestaña "Inspector"
- Luego pulsamos la sub-pestaña "Disposición"
- Hacemos clic en el checkbox "superponer rejilla"
- Hacemos click en "mostrar números de línea"



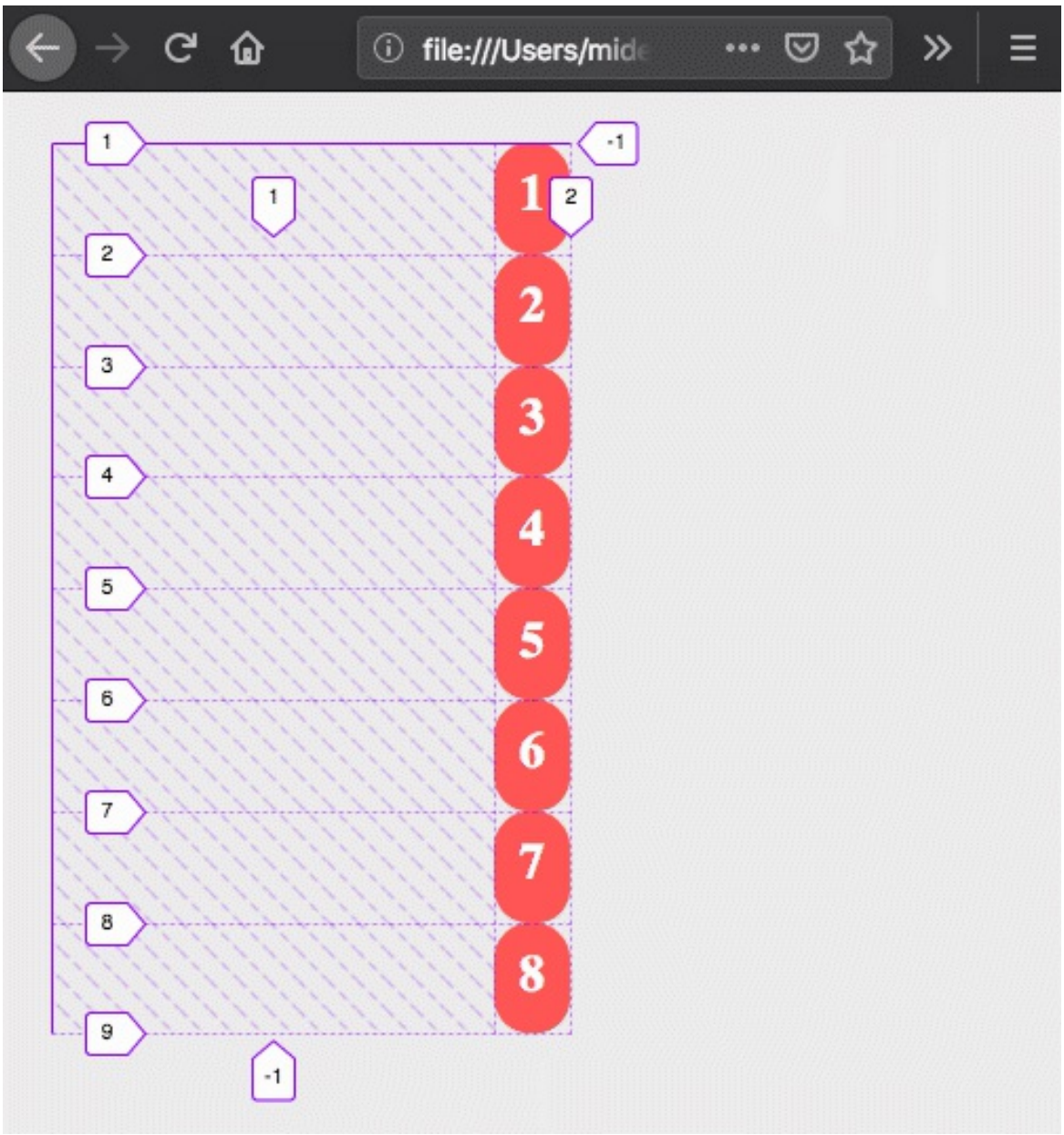
Modificar un poco la rejilla

A fin de ver que realmente está pasando algo gracias a haber definido el display grid, vamos a colocar algunos atributos extra en el contenedor.

Vamos a comenzar modificando el atributo "justify-content". El valor predeterminado que este atributo tiene es "stretch". Eso quiere decir que las cajas de cada una de las celdas se estiran para ocupar todo el espacio disponible.

Nosotros ahora vamos a colocar el valor "center", que hará que las celdas se coloquen en el centro de la rejilla y, como consecuencia de haber retirado el valor "stretch", provocará que las cajas ocupen solamente el espacio que marque su contenido.

```
.grid-container {
  display: grid;
  justify-content: center;
}
```



En la imagen anterior puedes ver el efecto. Hemos incluido el marco del navegador en la imagen para que veamos que efectivamente las cajas aparecen centradas.

Ahora vamos a probar con otro atributo que ya habíamos usado en el artículo anterior, que es el que nos permite definir cuantas columnas queremos.

```
.grid-container {
  display: grid;
  grid-template-columns: 100px 200px 300px;
  justify-content: center;
}
```

Estamos diciendo que queremos tres columnas en nuestra cuadrícula. La primera de 100 píxeles, la segunda de 200 y la tercera de 300. Las celdas se irán colocando de manera que se respete esta definición.

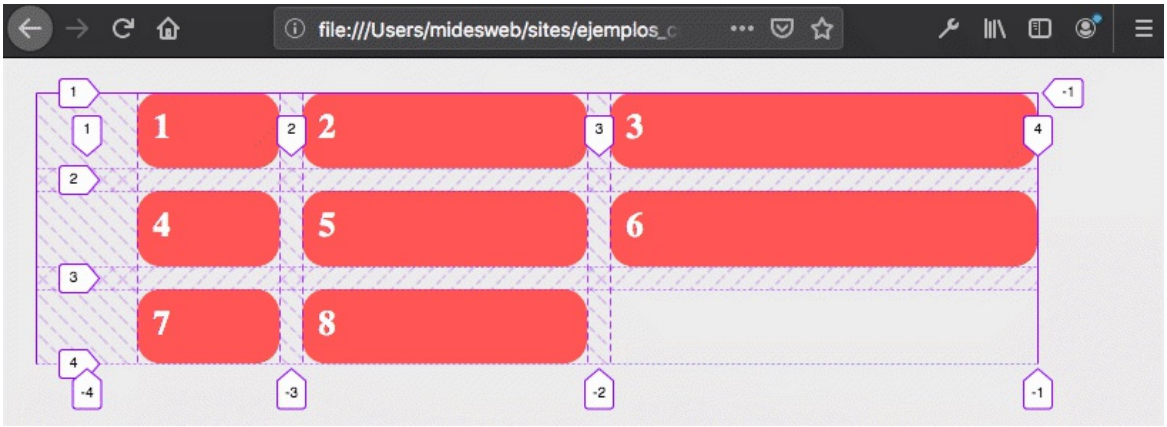
Al definir el tamaño de las columnas, las cajas toman esas dimensiones, independientemente que justify-content no sea "stretch". En este caso "justify-content: center" lo que ha provocado es que la rejilla entera se centre en la página.

Como puedes observar, dado que tenemos 8 elementos hijo, se van llenando las filas y columnas una a una, por lo que en la última fila sólo aparecen dos elementos, ya que el tercero no existe en el código HTML, pues para completarse necesitaríamos 9 elementos hijo.

Por último, vamos a colocar un par de nuevos atributos que no hemos visto todavía, para obtener espacios entre filas y columnas.

```
.grid-container {
  display: grid;
  grid-template-columns: 100px 200px 300px;
  justify-content: center;
  column-gap: 1em;
  row-gap: 1em;
}
```

Como podrás imaginar, column-gap aplica espaciado entre columnas y row-gap lo hace entre las filas. Es mucho más cómodo y preciso que colocarle márgenes a las cajas!



Conclusión a nuestra primera práctica con CSS Grid Layout

En esta primera práctica hemos podido conocer por encima una pequeña lista de los atributos disponibles para los elementos de la rejilla. Pero los hemos visto muy de pasada, solamente

con la intención de comenzar a experimentar un poco.

En el siguiente artículo abordaremos unas explicaciones más detalladas de los atributos disponibles para los contenedores de la rejilla, comenzando por explicar cómo definir filas y columnas, ajustar el espaciado y la adaptabilidad a todas las pantallas con unidades fr.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado / actualizado en *30/01/2020*
Disponible online en *http://desarrolloweb.com/articulos/propiedades-principales-css-grid-layout*

Ajustar las filas y columnas de la rejilla CSS

Cómo manipular la rejilla CSS ajustando la disposición de los elementos display: grid en filas y columnas. Cómo definir de manera exacta o relativa las anchuras y alturas de filas y columnas.

En pasados artículos del Manual de CSS Grid Layout hemos comenzado a usar el sistema de rejilla estándar que nos propone CSS. Hemos podido hacer unas primeras pruebas y hemos conocido de pasada algunos de sus principales atributos. Sin embargo hasta ahora simplemente hemos realizado un acercamiento superficial a este sistema. Ahora queremos comenzar a detallar las posibilidades del estándar, comenzando por la **definición de filas y columnas de la rejilla**.

Existen varios atributos que debemos conocer, así como valores novedosos en CSS, con unas unidades relativas que no existían hasta el momento (fr), que vamos a explicar también con detalle.

Para los ejemplos a lo largo de este artículo voy a estar trabajando siempre con este código HTML, en el que tengo un contenedor, al que le asignaremos el display: grid, que contiene 12 casillas:

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

```
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
  <div>11</div>
  <div>12</div>
</div>
```

Propiedad grid-template-columns

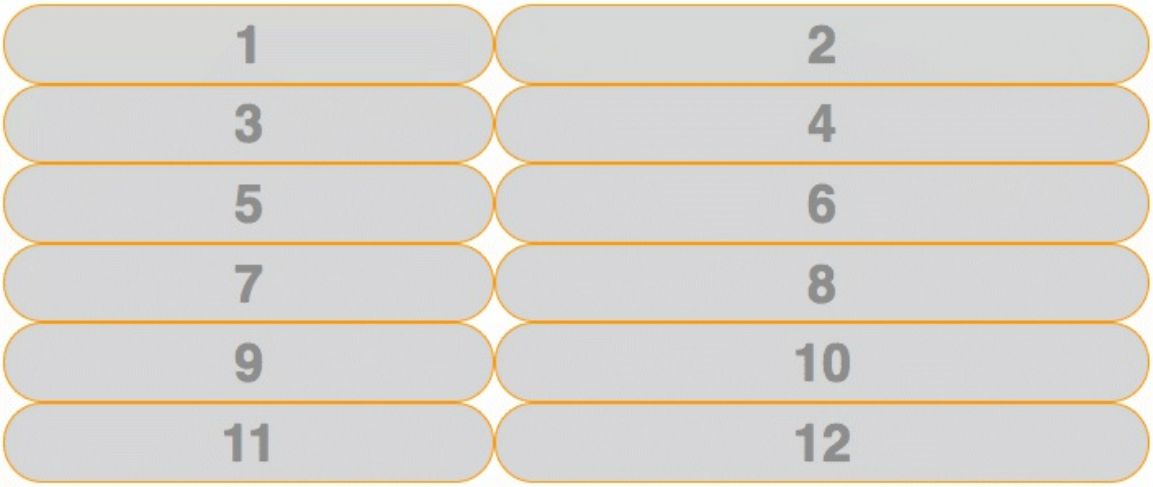
Comenzamos con la propiedad grid-template-columns, que sirve para definir el número de columnas que tendrá nuestra rejilla, así como los tamaños asignados a cada una.

El uso de este atributo es sencillo, pues simplemente tenemos que indicar valores con sus unidades, separados por espacios. Tantos valores como columnas queramos que tenga la rejilla.

```
.rejilla {
  display: grid;
  grid-template-columns: 300px 400px;
}
```

Con el anterior código estamos indicando que queremos dos columnas, la primera que mide 300 píxeles de ancho y la segunda 400 píxeles de ancho.

Con esta sencilla definición el navegador hará todo lo posible por ajustar el contenido de la rejilla a nuestra especificación. Su trabajo en este caso está bien claro, ir situando cada elemento, según le llegue, en la casilla siguiente. Empezando de izquierda a derecha y de arriba a abajo. Nos quedaría de esta manera:



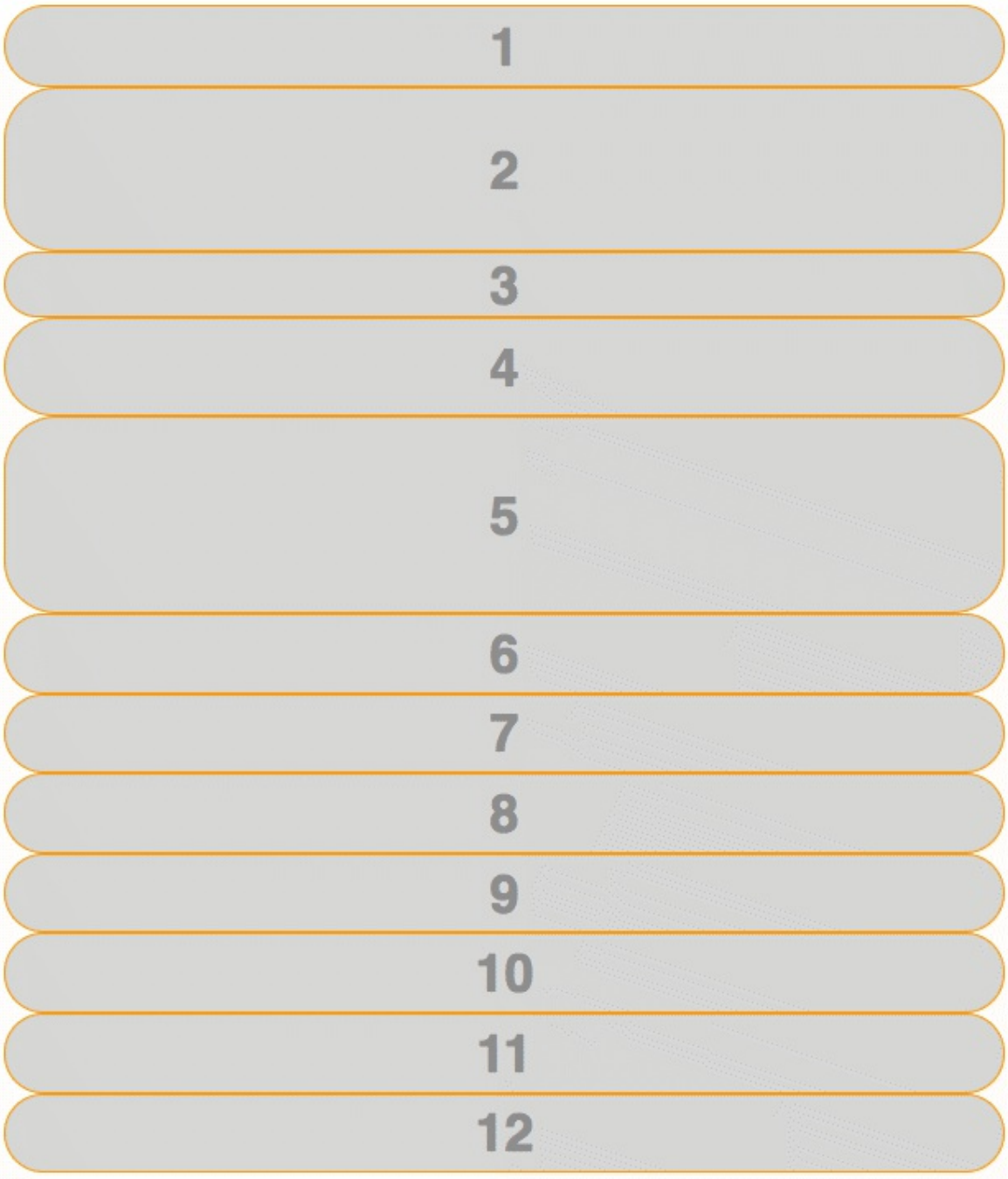
Propiedad grid-template-rows

Esta segunda propiedad sirve para definir los tamaños de las filas. Funciona de manera similar a grid-template-columns. Tenemos que poner simplemente los valores de las filas que

queramos que se vayan generando.

```
.rejilla {
  display: grid;
  grid-template-rows: 50px 100px 40px 60px 120px;
}
```

Para suponer como CSS Grid creará la rejilla en este segundo ejemplo tenemos que ser un poco más imaginativos, pues en realidad sólo hemos indicado los valores de grid-template-rows para 6 filas y nuestra división tenía 12 elementos hijo. ¿Qué pasará con los siguientes para los que no hemos definido nada? La respuesta la tienes en esta imagen:



Realmente nuestro caso era un poco más especial, porque al no haber definido grid-template-columns tendremos una sola columna. Como, si no indicamos nada más, los elementos se colocan uno debajo de otro, los hijos para los que no hemos definido dimensiones simplemente adquieren el tamaño que les toque, dependiendo del contenido y el padding asignado que tenga el elemento. Así pues, los últimos 6 elementos tendrán todos el mismo tamaño, como has podido ver.

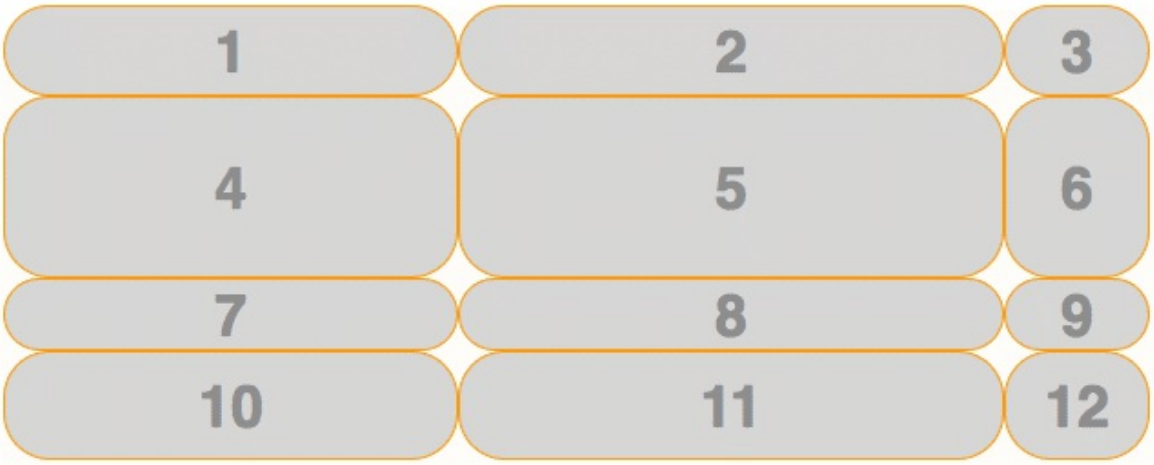
Combinar grid-template-columns con grid-template-rows

Es normal que en una rejilla queramos combinar ambas propiedades, para que podamos definir los tamaños de las filas y columnas a la vez.

Por ejemplo, en este nuevo código CSS tenemos ambas declaraciones indicadas al mismo tiempo:

```
.rejilla {
  display: grid;
  grid-template-columns: 250px 300px 80px;
  grid-template-rows: 50px 100px 40px 60px;
}
```

El resultado se puede ver en la siguiente imagen:



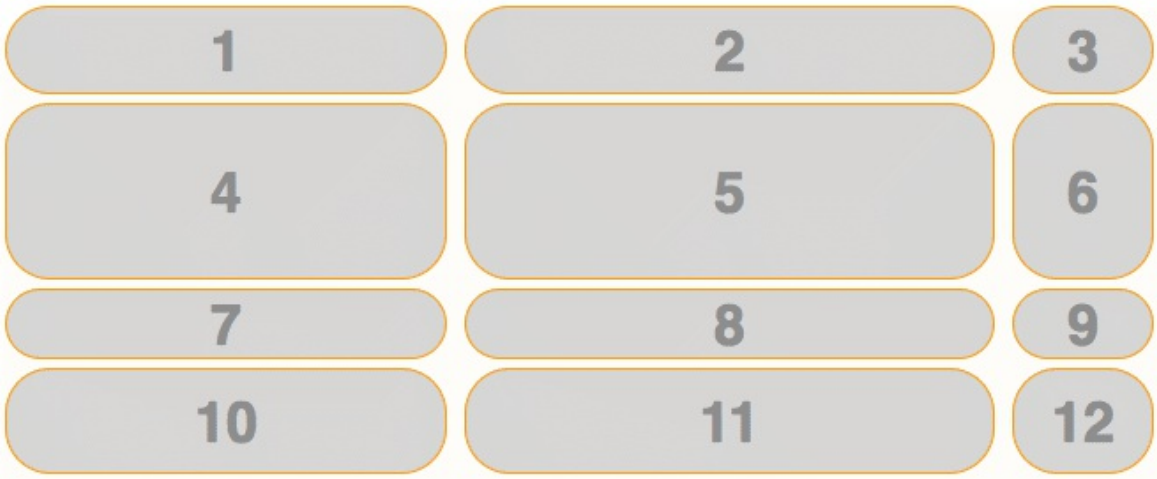
Cómo separar las filas y columnas con grid-row-gap y grid-column-gap

Seguramente te pase como a mi. Ver todas esas celdas tan pegadas te causa un poco de malestar visual. Si has trabajado con CSS durante bastante tiempo, la primera idea que se vendrá a tu cabeza es usar la propiedad margin para las casillas. Podrías usarla, pero CSS Grid Layout tiene una manera mucho más cómoda de establecer el espaciado entre filas y columnas.

Lo podemos conseguir con las propiedades grid-row-gap y grid-column-gap, que son mucho mejores que poner un margin, ya que generalmente no quieres que haya margen a la izquierda del primer elemento y tampoco margen a la derecha del último. Gracias a estos atributos nos evitamos tener que jugar con selectores del tipo de :first-child y similares, evitando complicaciones en el código CSS.

```
.rejilla {
  display: grid;
  grid-template-columns: 250px 300px 80px;
  grid-template-rows: 50px 100px 40px 60px;
  grid-row-gap: 5px;
  grid-column-gap: 10px;
}
```

Ya está. Hemos solucionado el tema del espaciado entre las celdas de la rejilla, de una manera rápida y cómoda. Como has podido comprobar nos facilita mucho la vida. El resultado lo podemos ver en la siguiente imagen:



Nota: Estoy colocando todas las unidades como px, pero como te imaginarás puedes trabajar con cualquier unidad de CSS de las conocidas.

Unidades fr

CSS Grid Layout introduce una nueva unidad CSS que también nos ayudará mucho, en este caso para crear diseños adaptables a cualquier dimensión de pantalla.

Como habrás observado, colocando tamaños con "px" el diseño nos queda bastante rígido. Lo podríamos solucionar de la manera que ya conoces, con unidades de porcentaje por ejemplo, pero esto nos obligaría a que todas las casillas tuvieran porcentajes como tamaño para que se adaptasen a todas las posibles medidas. Aún así, tendríamos problemas al aplicar los espaciados de las columnas y todavía sería más difícil combinar tamaños fijos con porcentajes, que nos permitan estar seguros que el diseño se adapta bien a todas las anchuras de pantalla.

Por eso CSS Grid Layout nos ofrece la nueva unidad "fr". Esta unidad indica "todo el espacio restante disponible". En principio usarla es tan sencillo como esto:

```
.rejilla {
```

```
  display: grid;
  grid-template-columns: 350px 1fr;
}
```

Tal como acabamos de definir grid-template-columns en combinación con la unidad fr, nuestra rejilla se adaptará a todas las anchuras posibles. Tendremos dos columnas, una a la izquierda que mide 350 píxeles y otra a la derecha cuya anchura será variable, pues toma todo el espacio restante de la página.

Con esta declaración puedes probar a redimensionar la ventana del navegador y comprobar cómo el tamaño de la columna de 350px permanece siempre igual, siendo el tamaño de la segunda columna variable.

Esta herramienta, las unidades fr que está disponible solamente en los elementos de display grid, ha venido a salvarnos la vida y a facilitarnos el diseño responsive de una manera increíble!!

Pero la unidad fr es todavía más potente. Podemos combinar estas unidades entre sí de muchas maneras.

Por ejemplo, en este caso vamos a crear tres columnas. La del medio tendrá 300 píxeles, y las dos de los laterales ocuparán resto del espacio disponible, pero dividido a partes iguales.

```
.rejilla {
  display: grid;
  grid-template-columns: 1fr 300px 1fr;
}
```

Ahora veamos que los valores de fr pueden variar para ajustar mejor las proporciones. Observa el siguiente ejemplo:

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

Estamos indicando que el espacio disponible se divida entre las dos columnas, de manera que la primera ocupará el doble que la segunda.

Las posibilidades de las unidades "fr" son muy amplias. Seguro que a partir de ahora las usarás constantemente.

Conclusión sobre la definición de filas y columnas en CSS Grid Layout

Hemos podido aprender y usar una buena cantidad de atributos CSS para ajustar de manera muy específica, versátil y adaptable el tamaño de las filas y columnas de la rejilla. Hemos encontrado ya algunos de los aspectos más destacables del estándar CSS Grid Layout, que

seguro que te habrán abierto los ojos y sorprendido gratamente.

Pero aún nos queda mucho que aprender. En próximos artículos veremos nuevas unidades, facilidades para crear rejillas adaptables al contenido y atributos diversos que todavía te ofrecerán más posibilidades y variantes para conseguir cualquier maquetación que necesites.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado / actualizado en *13/02/2020*
Disponible online en *http://desarrolloweb.com/articulos/ajustar-filas-columnas-rejilla-css-grid*