INSTITUTO TECNOLOGICO NACIONAL DE MEXICO – INSTITUTO TECNOLOGICO DE LA ZONA OLMECA.

ING. SISTEMAS COMPUTACIONALES - UNIDAD 03 – FERNANDO RUIZ VIDAL

CSS-GRID

# CSS GRID(Concepto).

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.

Pequeños Conceptos

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>

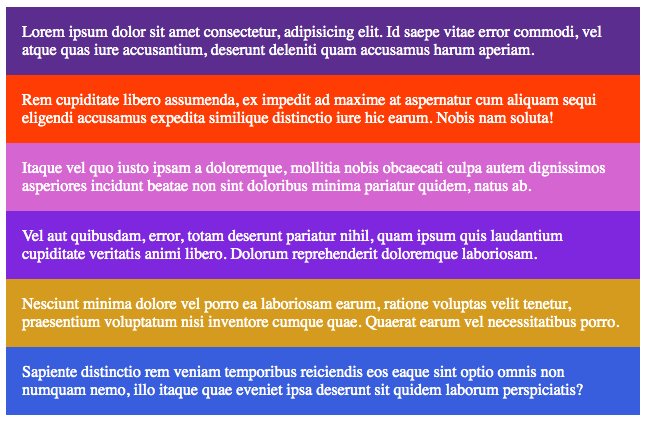
Para conseguir que un contenedor trabaje con el sistema de rejilla, simplemente tenemos que colocar "display: grid" en su CSS.

.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 {

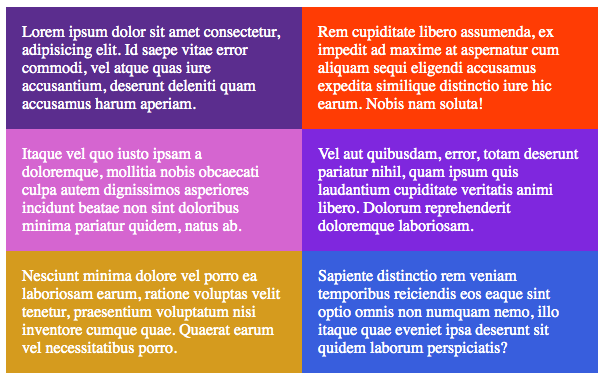
display: grid;

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.

# DESVENTAJAS

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.

# Comparación con Flexbox.(XD)

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.

# ETIQUETAS CON MINI-EJEMPLOS

## 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;

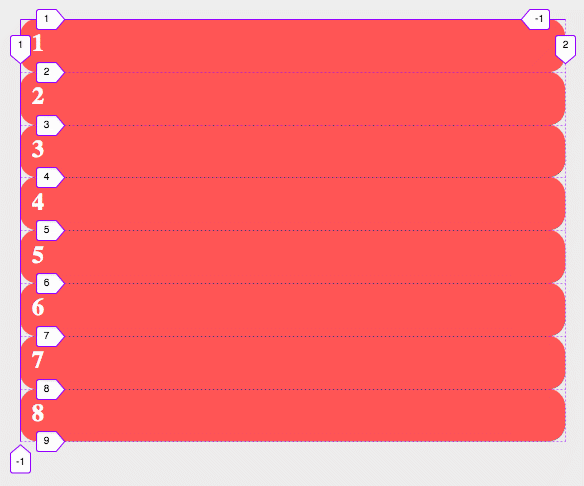
}

## 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.



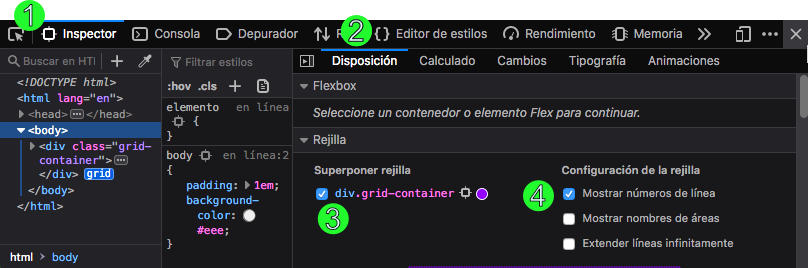
En la anterior imagen podemos ver cómo esta rejilla se mostraría en Firefox Developer Edition, al abrir las herramientas de desarrolladores y activar la inspección de la rejilla.

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.

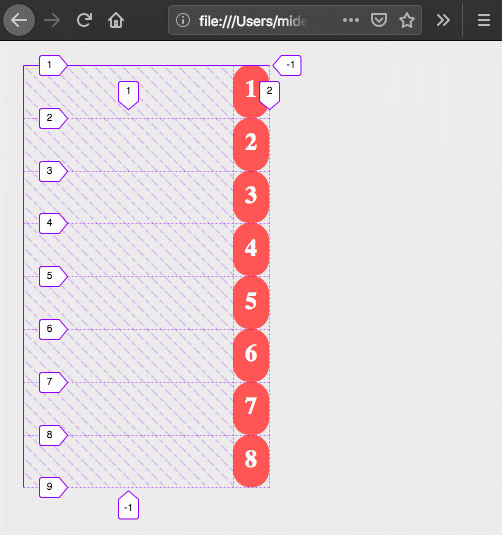
Nosotros ahora vamos a colocar el valor "center", que hará que las celdas se coloquen en el centro de la rejilla y, como consecuencia de haber retirado el valor "stretch", provocará que las cajas ocupen solamente el espacio que marque su contenido.

.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;

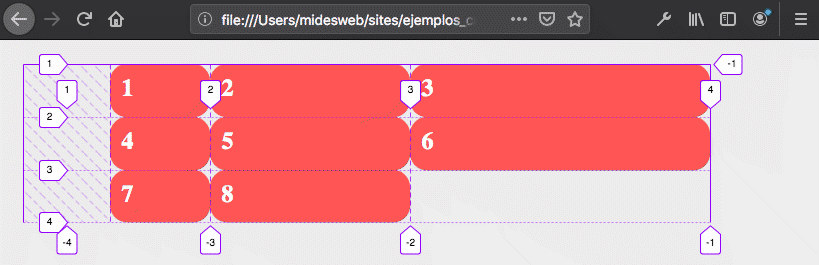
grid-template-columns: 100px 200px 300px;

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!

