**CSS GRID**

NOTA:

<https://platzi.com/blog/flexbox-vs-css-grid-cual-es-la-diferencia/>

Mi resumen:  
Flexbox para componentes  
CSS Grid Layout para estructura

3- Css Grid Layout. (Mis apuntes de la clase).

En css podemos interpretar cualquier cosa como una caja y así darle estilos.

-Grid Container: Nos permite poner las especificaciones ahí, facilitándonoslo de esa manera el trabajo.

-Grid Item: Es donde se guarda el contenido dinámico que se haga dentro de la pagina. Este es hijo directo del Grid Container.

-Grid line: son las lineas divisoras, tanto horizontales como verticales.

-Grid Track: Son las filas y columnas.

-Grid Cell: Es el espacio donde dos lineas horizontales y dos lineas verticales se juntan.

-Grid Area: Es la suma de n grid cells.

:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

## **4- CLASE 1: Definiendo columnas**

Se debe poner el display de grid al contenedor

.container{

display: grid;

}

Para definir columnas debemos ponerle la propiedad de grid-template-columns al contenedor, en esta debemos definir el tamaño de cada una de las columnas, por ejemplo: si en la propiedad ponemos los siguientes valores

.container{

grid-template-columns: 200px200px200px;

}

estamos estableciendo tres columnas, cada una de 200px.

También podemos poner medidas dinámicas, tales como porcentajes, se definen exactamente igual que como en el ejemplo anterior:

.container{

grid-template-columns: 25%50%25%;

}

También podemos combinar las medidas relativas con las medidas absolutas:

.container{

grid-template-columns: 25%200px25%;

}

:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

5- **CLASE 2: Definiendo filas**

Al igual que como definimos las columnas definimos las filas, lo único que cambia es el final de la propiedad, es decir que esta vez ya no será grid-tempate-columns sino que será grid-tempate-rows.

.container{

grid-template-columns: 25%200px25%;

grid-template-rows: 300px300px;

}

\_ Si tenemos el mismo código de la clase pasada aparecen 2 filas más debido al numero de items, la altura de estas es implícita, debido a que no definimos el valor. Nosotros antes definimos explicitamente el tamaño de 3 columnas y 2 filas\_

Podemos definir los tamaños de las filas y las columnas con la misma propiedad, esta propiedad es grid-template, debemos definir primero los tamaños de las filas, luego separandolos con un slash debeos definir los tamaños de las columnas

**grid**-template: 300px 150px / 25%200px 25%;

**Apuntes:**

De la misma forma que al definir columnas, cuando vamos a definir una fila, le añadimos al elemento padre la propiedad grid-template-rows y le agregamos el numero de valores por medida de las filas que vayamos a maquetar explícitamente (grid explicito).

Para realizar un trabajo automático a la hora de designar filas y columnas, usamos la propiedad grid-template, y separamos la configuración de las filas y columnas explicitas con un /.

**Grid explícito:** Es todo aquello a lo que le designamos una regla al tener el display como grid.

**Grid implícito:** Es aquello que se encuentra designado por defecto y no hemos tocado con ninguna regla cuando se tiene el display de grid.

<https://codepen.io/JuanDiegoS/pen/wbLRaw>

IMPORTANTE: Ahora que conocemos grid, ¿cuando es necesario que usemos flex?

Si se necesita un diseño que sus elementos se alineen en una sola dimensión (fila o columna) utiliza **flexbox**. Si por el contrario se necesita un diseño que tus elementos se alineen en 2 dimensiones (filas y columnas) utiliza **Grid** Es decir:

Si tenemos 5 elementos cards y utilizamos **flex** vemos que en la siguiente fila los dos elementos restantes 4 y 5 que no se acomodaron en el viewport de la pantalla pasan a ocupar todo el ancho de la segunda fila:

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

6- **Grids identados y tipos de displays**

Para centrarnos en un elemento especifico, aun cuando todos tengan la misma clase, debemos usar la pseudoclase :nth-of-type(#delElemento).

Para asignarle propiedades de Grid a un subelemento usamos la misma declaración de display: grid y luego copiamos y pegamos todas las reglas que teníamos asignadas del padre (en caso de que queremos que el subelemento herede las reglas).

\*\*Notas: \*\*

* Para que un subelemento herede todas las configuraciones Grid del elemento padre usaríamos para resumir la declaración de display: subgrid, pero aun no existe en los navegadores al día de hoy.
* Para crear un grid en línea usamos la propiedad display: y le asignamos el valor de inline-grid. Esta propiedad tampoco se encuentra disponible, pero lo estará.

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

7- Espaciado entre filas y columnas

**Anotaciones:**

Para lograr un espaciado vertical de la manera tradicional, usamos la propiedad margin-right y le añadimos como valor la medida deseada. De esta misma forma para un espaciado horizontal usamos la propiedad margin-bottom y añadimos el valor deseado.

La mejor forma de hacerlo para un elemento Grid es con la propiedad grid-gap, con lo cual consecuentemente le añadimos el valor deseado de la medida de espaciado, esta propiedad funciona para el espaciado vertical y horizontal. Individualmente usamos grid-column-gap: (espaciado en columnas) y grid-row-gap: (espaciado en filas).

