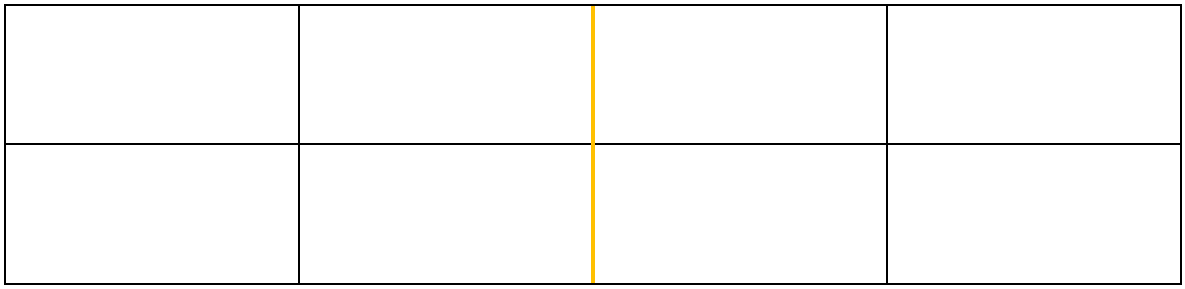
Curso de CSS Grid Layout

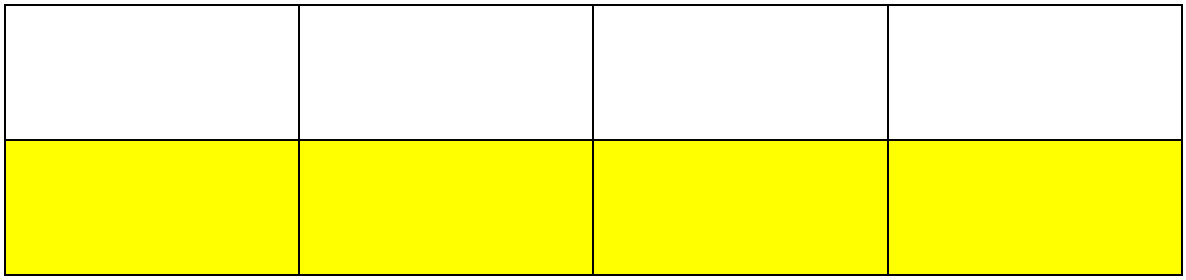
Grid Container: Elemento padre principal donde se asignará un {display: grid}.

Grid Item: Hijos directos de un grid container.

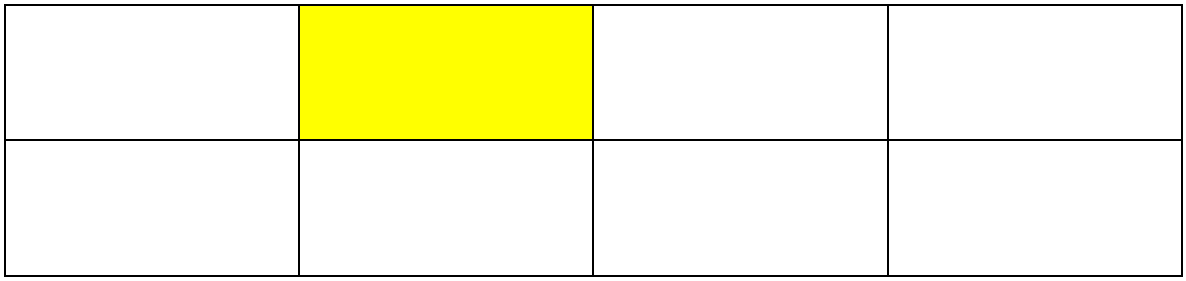
Grid line: Líneas divisoras horizontales y verticales.



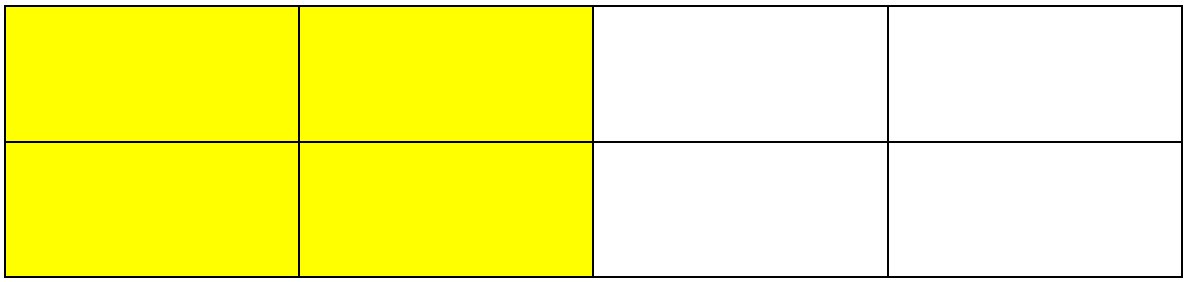
Grid Track: Espacio entre dos líneas adyacentes, filas y columnas.



Grid Cell: Espacio entre dos filas adyacentes y 2 columnas adyacentes.



Grid Area: Espacio que rodeado por 4 grid líneas.



Básicamente funciona en todos los navegadores.

Cuando se usa display:grid; Chrome lo interpreta en el inspector como líneas.

Grid-template-columns: 200px 200px 200px; Con esto se marca las columnas y el tamaño que tendrá cada una. También se puede hacer con porcentajes.

Grid-template-rows: 300px 300px; Con esta propiedad se manipula el alto las filas, las filas que no se le indique un tamaño tomaran un tamaño implícito.

Grid-template: filas / columnas; Esta propiedad es la versión resumida de los datos anteriores. Primero se col9ocan las filas y después las columnas.

.item: nth-of-type(4){} Es una seudo clase que nos permite utilizar el elemento que nosotros querramos.

Overflow: auto; Te permite dibujar barras de desplazamiento o mostrar el contenido excedente de un elemento a nivel de bloque.

Display: subgrid; Hereda lo especificado en el grid-template // Nota: Aun no está disponible.

Grid-column-gap: 3px; Genera un hueco o espacio entre columnas.

Grid-row-gap: 3px; Genera un hueco o espacio entre filas.

Grid-gap: filas columnas; Es la forma resumida de los valores de arriba, primero se coloca el valor de las filas y después el de las columnas.

Los grid-template se pueden dividir en fracciones.

Grid-template: 300px 100px 100px / 1fr 1fr 1fr; Al colocar 1fr les indicas que son partes iguales y dependiendo la cantidad de fracciones que se pongan es la cantidad de veces que será dividido.

Si se utiliza auto en vez de 1fr, va a distribuir el tamaño dependiendo del contenido.

Repet(3, 1fr); Esta es una función de Css. De esta forma no se tiene que escribir de manera individual. El primer valor indica la cantidad de veces que se va a repetir, el segundo indica el tamaño que tendrá.

Para evitar que se deforme el contenido se puede utilizar la función **minmax();** donde el primer valor seria el tamaño mínimo que se le permite para que no se deforme y max seria el tamaño máximo permitido.

Se pueden usar estas funciones de manera combinada.

Repet(3, minmax(200px, 1fr));

Grid-template-areas: “Header Header”

“left contenido”

“footer footer”;

Cada cosa que se ve dentro de las comillas va a corresponder a las áreas que tenga esa fila. Especifica nombres para cada grid áreas.

Estas áreas no están asociadas a ningún elemento particular de la cuadrícula, pero pueden referenciarse desde otras propiedades de posicionamiento en la cuadricula.

Para poder aplicar esta propiedad se tiene que generar clases y de esta manera relacionarlas.

.header{grid-area:header} El valor después del grid área es uno de los que se asignaron en grid-template-areas.

Grid-column-start:1;

Grid-column-end:3;

Sirven para limitar una columna, decirle donde inicia y donde termina. Se cuentan las líneas no las columnas.

Grid-column: 2 / 4; Es la forma resumida, el primer digito es donde inicia y el segundo es donde termina.

Usando **span** se cuentan las columnas.

Grid-auto-flow: row; Cuando nos sobran cosas, este elemento organiza lo que sobra, por defecto está en row, nosotros lo podemos cambiar a columna y va a hacer que los elementos desplazados por el grid se vuelvan columnas.

Grid-auto-columns: 200px 200px 200px 200px; Con esta propiedad se le da el tamaño a los valores sobrantes.

Justify-items. Nos ayuda con un alineado de forma horizontal.

Align-items. Nos ayuda con un alineado de forma vertical.

Align-self:

Justify-self:

Les esta diciendo que se alineen a si mismos que no respeten los valores generales.

Justify-content:

Align-content:

Realizan funciones similares, pero sobre todo el contenido, este no modifica el tamaño de los elementos.