GRID CSS

1. CUADRÍCULA CSS

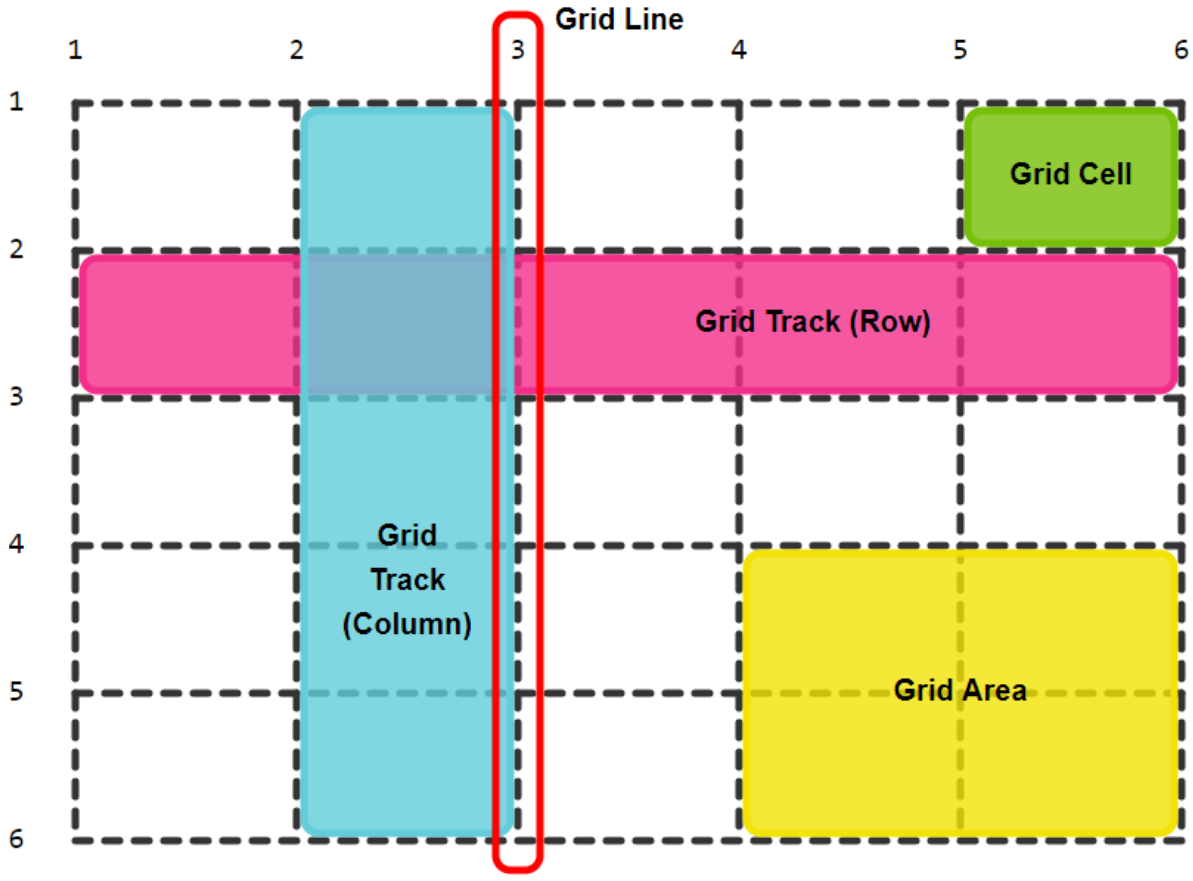
QUE ES UNA CUADRICULA: es un conjunto de líneas horizontales y verticales que se interceptan - un grupo define columnas y las otras filas. Los elementos se pueden colocar en la cuadrícula respetando estas columnas y filas.

CSS GRID: presenta un sistema de cuadrícula bidimensional para CSS. Las cuadrículas se pueden utilizar para posicionar áreas principales de la página o pequeños elementos de la interfaz de usuario.

El diseño de cuadrícula CSS tiene las siguientes características:

Contenedor Padre (***Grid Container***).

- Elementos Hijos (***Grid Items***).
- Líneas de Cuadrícula (***Grid Lines***).
- Pista de Cuadrícula (***Grid Track***).
- Celda de Cuadrícula (***Grid Cell***).
- Área de Cuadrícula (***Grid Area***).



2. GRID EXPLÍCITA

se define de manera directa y precisa usando las propiedades:

grid-template-columns: DEFINE LAS COLUMNAS

grid-template-rows: DEFINE LAS FILAS

grid-template-areas: DEFINE LAS GRID AREAS

grid-template: es un shordhand que engloba esas 3 propiedades

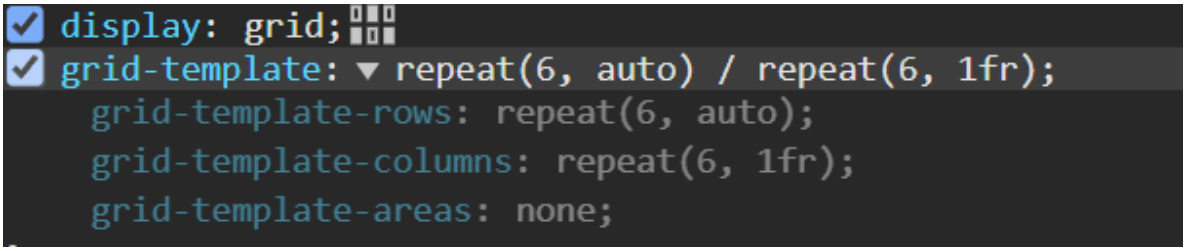
el primer valor corresponde a las filas

el segundo valor corresponde a las columnas

tercer valor corresponde a áreas de la grid.

Nota: Cuando se usa esta propiedad se tienen que dar tanto valores en fila como en columnas, el tercer valor es opcional

ejemplo:



Esto significa que se especifica el número de filas, columnas y áreas de la cuadrícula de forma explícita, en lugar de dejar que el navegador las determine automáticamente.

ejercicio:

css

```
.grid-explicit{
   display: grid;
   grid-template-columns: repeat(4, 1fr) ;
   grid-template-rows: repeat(4, 12.5rem);
 }
```

nota: estas propiedades se aplican al contenedor padre para definir las columnas y filas internas

CABE DESTACAR, QUE EL VALOR **repeat(cantidad columnas/filas, tamaño de esas columnas/filas)**, permite optimizar, a la hora de repetir la misma cantidad de filas, columnas y hacer patrones repetitivos.

adicionalmente, tenemos la propiedad

column-gap: espaciado entre columnas

row-gap: espaciado entre filas

gap: es un shordhand, propiedad abreviada donde el primer valor le corresponde a filas y el segundo a columnas y permite el espaciado entre las mismas.

nota: si se coloca un solo valor lo aplica en ambas direcciones columnas y filas

3. POSICIONAMIENTO CON GRID LINES

El posicionamiento de filas y columnas se logra a través de la posición en las **grid lines** (lineas de cuadriculas), dándole el valor de inicio de la línea y fin a través de las siguientes propiedades.

grid-column-start: marca el inicio de la linea de la columna

grid-column-end: marca el fin de la linea de la columna

grid-row-start: marca el inicio de la linea de la fila

grid-row-end: marca el fin de la linea de la fila.

NOTA: ESTAS SON PROPIEDADES DE LOS GRID ITEMS (HIJOS DIRECTOS)

ejemplo:

```
.items:nth-child(14){
  color: white;
  background-color: #cd2626;
  text-align: center;
  align-content: center;
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 2;
  grid-row-end: 3; }
```

Existe un shordhand que abrevia estas dos propiedades

para columna: **grid-column: 1/4;** donde el primer valor es la línea de inicio y el segundo el fin

para fila: **grid-row: 2 / 3;**

nota: si no se le asigna el valor de fin este lo tomara por defecto de la línea siguiente.

ejemplo:

grid-items

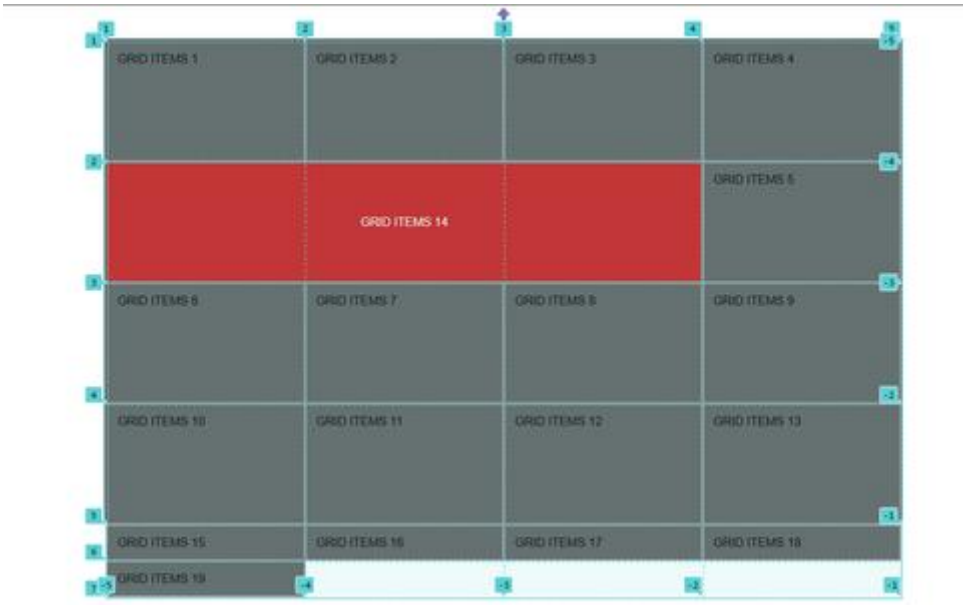
```
.items:nth-child(14){
  color: white;
  background-color: #cd2626;
  text-align: center;
  align-content: center;
  grid-column: 1/4;
  grid-row:2 / 3;  }
```

además, existe una propiedad que en globa estas dos propiedades anteriores que es la cuadricula de área o **grid-area**

grid-area: grid-row-start / grid-column-start / grid-row-end /grid-column-end;

ejemplo:

```
.items:nth-child(14){
  color: white;
  background-color: #cd2626;
  text-align: center;
  align-content: center;
  grid-area: 2 / 1 / 3 / 4;  }
```



también existe el valor "**span**" en las propiedades grid-column y grid-row para definir el número de líneas que un elemento de la cuadrícula debe ocupar.

Básicamente, indica que el elemento se extiende por un cierto número de columnas o filas a través de las grid tracks, a partir de una línea de inicio específica.

Sintaxis

grid-column: span 5; /* le estamos diciendo expándete 5 columnas desde tu posición de inicio */

grid-row: span 3; /* le estamos diciendo expándete 5 filas desde tu posición de inicio */

de igual forma podemos indicarle desde que posición queremos que empiece ejemplo:

grid-column: 2 / span 5; /* le estamos diciendo expándete 5 columnas desde grid-line column 2 */

grid-row: 3 / span 3; /* le estamos diciendo expándete 5 filas desde la grid-line row 3 */

Ejemplo:

Css

GRID ITEMS

.items:nth-child(14){

grid-area:2/3/3/5; }

.items:nth-child(8){

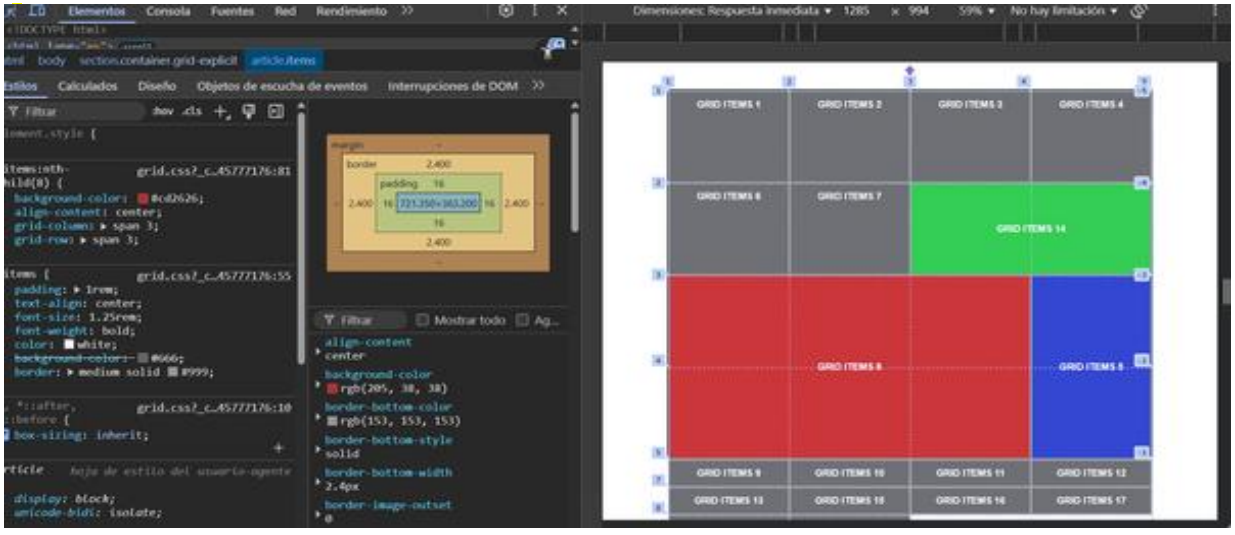
grid-column: span 3; /* expándete 3 grid track column desde tu posicion de inicio

grid-row: span 3; /* expándete 3 grid track row desde tu posicion de inicio }

.items:nth-child(5){

grid-column: 4 / auto; /* empieza desde la grid line column 4 y termina automaticamente en la siguiente*/

grid-row: 3 / span 3; /* expandete 3 grid track row desde la poscion grid line rows 3 de inicio */



4. POSICIONAMIENTO CON NOMBRES DE GRID LINES

utilizando nombres de líneas permite definir dónde debe ubicarse cada elemento dentro de la cuadrícula, sin necesidad de usar números de línea. Esto se logra asignando nombres a las líneas de la cuadrícula y luego usando esos nombres en las propiedades de posicionamiento de los elementos.

La sintaxis es se crea el nombre entre corchetes y después se asigna el tamaño de la cuadricula

Ejemplo

```
GRID CONTAINER
.grid-line-names{
   display: grid;
   grid-template-columns: [columna-a] 1fr [columna-b] 1fr [columna-c] 1fr [columna-d];
   grid-template-rows: [fila-a] 1fr [fila-b] 1fr [fila-c] 1fr [fila-d] ; }
GRID ITEMS
.line-items:nth-child(5){
   grid-column: columna-a / columna-c;
   grid-row: fila-a / fila-b;
   background-color:#cd2626 }
```

nota: los nombres asignados, no funciona con la propiedad grid-area**—ojo**

5. POSICIONAMIENTO CON GRID AREAS

permite definir áreas con nombres específicos en la cuadrícula y luego asignar elementos a esas áreas.

Se utilizan las propiedades **grid-template-areas** para definir un mapa de áreas dentro de la cuadrícula. Cada área se nombra y se asocia a una celda o grupo de celdas en la plantilla de la cuadrícula.

Sintaxis:

sí tenemos una grid de 3 columnas y 3 filas en ese mismo sentido asignamos los nombre de las areas"

```
grid-template-areas:
"zona1 zona1 zona2 "
"zona3 zona4 zona4 "
"zona5 zona5 zona5 "
```

nota: si queremos que una grid cell, o celda de cuadricula no tenga nombre, colocamos un punto para que quede vacío sin nombre de zona"

ejemplo

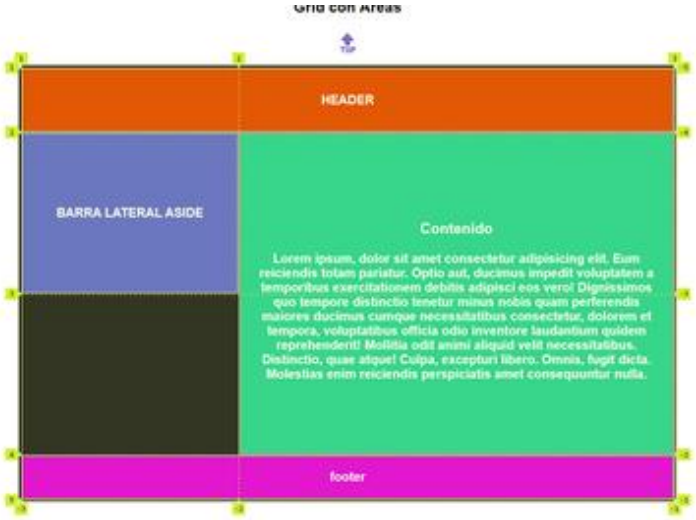
siguiendo con el ejemplo anterior

grid-template-areas:

"zona1 zona1 zona2 "

"zona3 zona4 . " /* la última grid cell de la fila 2, quede sin nombre de posicionamiento*/

"zona5 zona5 zona5 "



ejemplo en css

/* posicionamiento con grid areas */

.grid-areas{

display: grid;

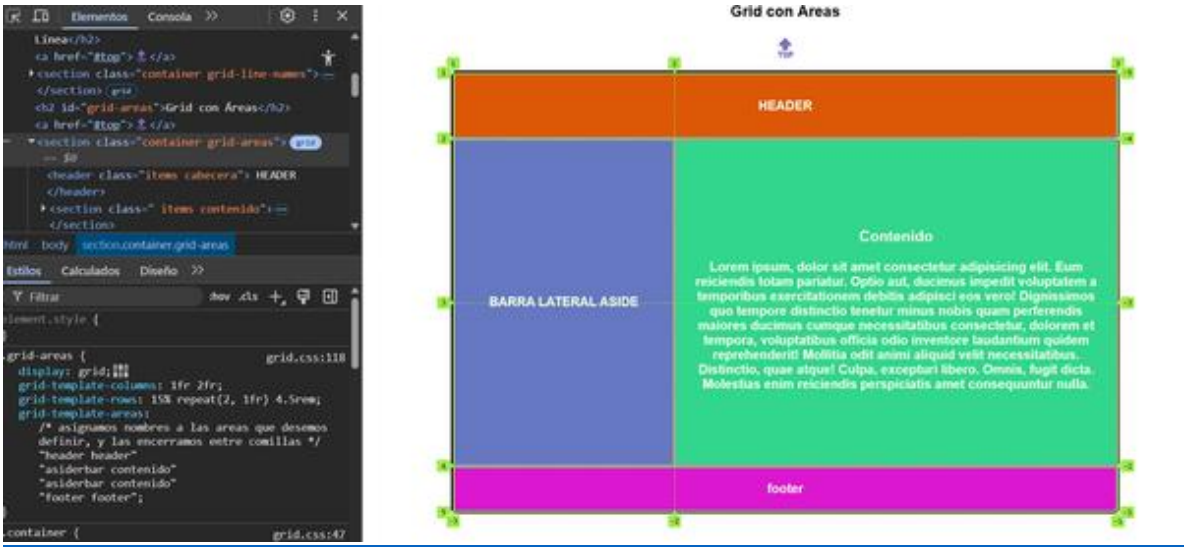
grid-template-columns: 1fr 2fr;

grid-template-rows: 15% repeat(2, 1fr) 4.5rem;

/* una vez realizada la cuadricula explicita podemos asignar nombres a las area de cuadricula definida con grip-template-area */

grid-template-areas:

```
    /* asignamos nombres a las areas que desemos definir, y las encerramos entre comillas */
    "header header"
    "asiderbar contenido"
    "asiderbar contenido"
    "footer footer";
    /* se asigna nombres segun la posicion que queremos que ocupe en nuestra cuadricula grid */
    /* ademas colocamos los nombre asignados ocupando la poscion que deseamos en nuestra grid explicita  */
}

/* a cada elemento asignamos los nombres de nuestras areas definidas con grid-area */
.cabecera{
    background-color: #e34800;
    grid-area: header;
}
.contenido{
    background-color: #27eba3e2;
    grid-area: contenido;
}
.asider{
    background-color: #5f6cce;
    grid-area: asiderbar;
}
.footer{
    background-color: #e300df;
    grid-area: footer;
}
```



6. **GRID IMPLÍCITA. GRIDS DE BLOQUE Y DE LÍNEA**

La cuadrícula implícita es la cuadrícula que se genera automáticamente cuando los elementos de la página se colocan fuera de la cuadrícula explícita que se ha definido.

permite crear contenido flexible y adaptable, ayuda a la gestión de contenido dinámico.

podemos controlar las vías implícitas con las propiedades:

grid-auto-rows, grid-auto-columns

Es decir, si se llegara a formar Grid implícitas, podemos establecer un valor de esas celdas.

ejemplo:

grid-auto-rows: minmax(200px, auto); /* con mimax (min, max) establecemos el minimo y el maximo del valor*/

donde estamos diciendo con el valor minmax, es que si se llegan a crear las gird implicitas como minimo tengan 200px de alto y maximo se ajuste al valor de su contenido con auto)

por otra parte, el grid de bloque y el grid de linea

se establecen con los valores

display: grid; /* bloque*/

display: gird-line; /* linea*/ /* caso que tengamos más de un contendor de bloque y queramos trabajarlo uno al lado de otro aplicando el estilo bidimensional en cuadricula*/

dependiendo los requerimientos utilizaremos contenedores en bloque o líneas en cuadriculas.

7. FLUJO DE LA GRID (GRID FLOW)

El flujo dinámico de la grid está en función de la orientación de la cuadricula, por defecto trabaja en fila rows y las cuadriculas que se generan implícitamente en filas.

para poder cambiar la orientación utilizamos la propiedad **grid-auto-flow,** determina si los elementos se colocan por fila (row) o por columna (column), y en función de este será el flujo dinámico de la grid respetando la grid explicita determinada.

es decir, si el flujo está en row, respetará las columnas explicitas y el flujo se moverá hacia abajo.

Ejemplo:

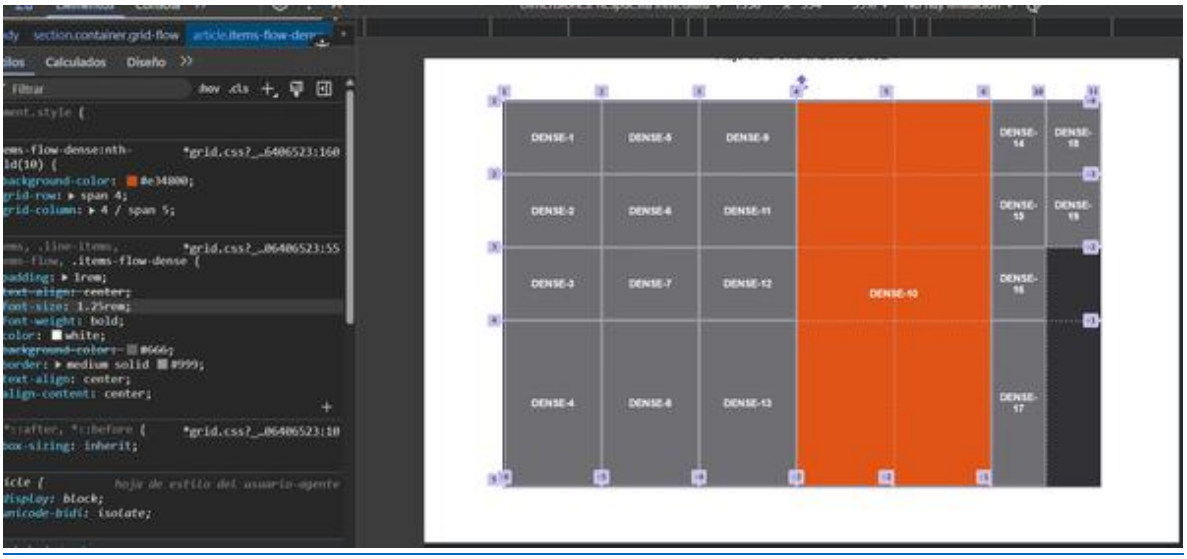
```
.grid-flow{
   display: grid;
   grid-template-columns: repeat(5,1fr);
   grid-template-rows: repeat(3, 12rem);
   grid-auto-flow: column;
   grid-auto-columns: max-content;
}
```

En cambio, sí está en columna, genera implícitamente columnas y el flujo se moverá hacia la derecha en función de las columnas. pero si necesita generar implícitamente filas por la distribución de los elementos de la cuadricula lo generara.

EJEMPLO DE FLUJO DINAMICO DE LA GRID EN COLUMNA

css

```
.grid-flow{
display: grid;
grid-template-columns: repeat(5, 200px);
grid-template-rows: repeat(3, 150px);
grid-auto-flow: column;
}
```



nota: cada destacar, que si los elementos no tienen definito su tamaño dentro de la cuadricula, el valor por defecto que tomara es stretch, y se ajustara de acuerdo a la alineacion dentro de la cuadricula.

8. FLUJO DENSO DE LA GRID (GRID FLOW DENSE)

Si durante la formacion explicita de la grid quedan espacios vacios, podemos utilizar esos expacios con el valor "dense"dentro del grid-auto-flow. Donde se reordena los elementos dentro de la cuadrícula para llenar los espacios vacíos, independientemente del orden original en el HTML. A ESTO SE LE CONOCE COMO FLUJO DENSO DE LA GRID

9. GRID LAYERS: CELDAS EN CAPAS (SUPERPOSICIÓN)

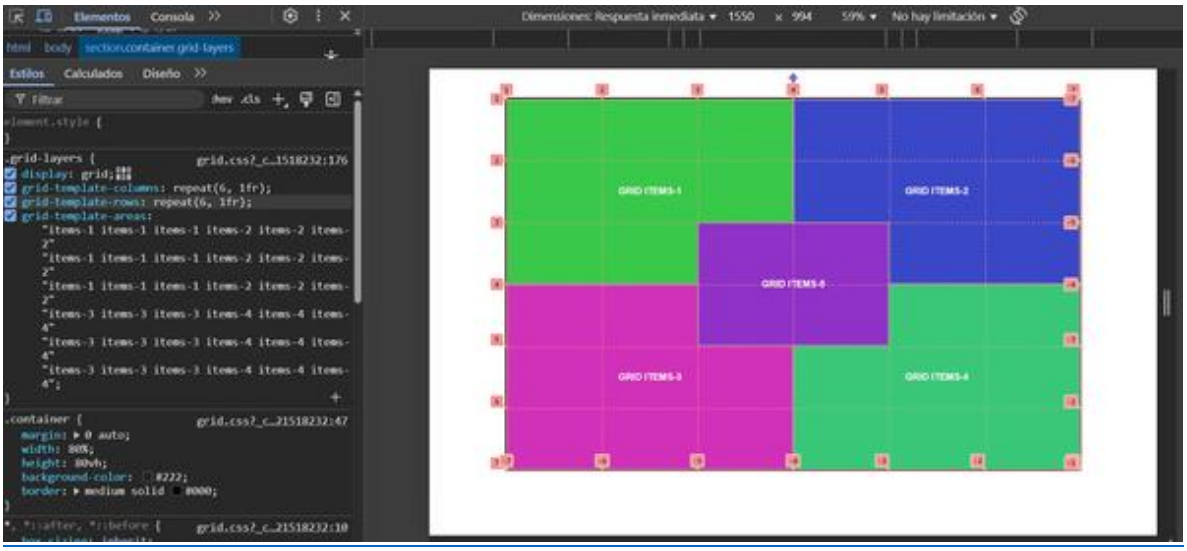
se utiliza para colocar elementos uno encima de otro dentro de una misma celda de la cuadrícula. Se logra definiendo las áreas de cuadrícula de los elementos de manera que coincidan y luego, si es necesario, controlando el orden de las capas con la propiedad z-index

ejemplo:

```
.grid-layers{
   display: grid;
   grid-template-columns: repeat(6, 1fr);
   grid-template-rows: repeat(6, 1fr);
   grid-template-areas:
   "items-1 items-1 items-1 items-2 items-2 items-2"
   "items-1 items-1 items-1 items-2 items-2 items-2"
```

```
    "items-1 items-1 items-1 items-2 items-2 items-2"
    "items-3 items-3 items-3 items-4 items-4 items-4"
    "items-3 items-3 items-3 items-4 items-4 items-4"
    "items-3 items-3 items-3 items-4 items-4 items-4";
}

.items-grid-layers:first-child{
    background-color: #26cd42;
    /* grid-column: 1/ 3 ;
    grid-row: 1 / 3 ; */
    grid-area: items-1;
}

.items-grid-layers:nth-child(2){
    background-color: #263fcd;
    /* grid-column: 3 / 5 ;
    grid-row:  1 /  3; */
    grid-area: items-2;
}

.items-grid-layers:nth-child(3){
    background-color: #cd26bc;
    /* grid-column: 1 / 3 ;
    grid-row:  3 / 5; */
    grid-area: items-3;

}

.items-grid-layers:nth-child(4){
```

```
    background-color: #26cd74;
    /* grid-column: 3 / 5 ;
    grid-row: 3 / 5 ; */
    grid-area: items-4;

}
.items-grid-layers:last-child{
    background-color: #8526cd;
    grid-column: 3 /  5 ;
    grid-row:  3 /  5;
}
```



en este ejemplo se utiliza la propiedad grid-area (y las propiedades abreviadas individuales shordhand grid-row, grid-column) para asignar a los elementos una posición dentro de la cuadrícula.

10. order y alinacion items GRID

order: es la propiedad que especifica el orden de los elementos dentro de la grid, el valor por defecto de todos los elementos es 0, y admite valores negativos y positivos.

ejemplo:

```
.grid-order{
   display:grid;
   grid-template-columns: repeat(5, 1fr);
   grid-template-rows:repeat(2, 1fr);
}
.items-order:last-child{
   order:-2;
}
```

en cuanto a la alineacion de elementos dentro de la grid tenemos las propiedades:

justify-items: main axis(x) /* valores: stretch, start, center, end, basilene*/

align-items: cross axis(y) /* valores: stretch, start, center, end, basilene*/

place-items: es un shordhand que engloba estas dos propiedades, donde el primer valor es para align-items y el segundo para justify-items, separados por espacio, si solo se coloca un valor lo tomara en ambos ejes**--ojo**

en cuanto a las propiedades de los hijos si queremos moverlos independientemente usamos las propiedades:

justify-self

align-self

place-self: es un shordhand que engloba las propiedades align-self y justify-self en ese mismo orden se aplica

funciona exactamente igual con los mismo valores, pero la diferencia es que se aplica al mismo elemento hijo.

ejemplo:

```
.grid-align{
   display: grid;
   grid-template-columns: repeat(3, 250px);
   grid-template-rows: repeat(2, 200px);
```

```
    justify-items: end;
    align-items: end;
}

.align-grid:last-child{
    background-color: #26cd42;
    justify-self: center;
    align-self: center;
}
```

10. ORDER Y ALINACION ITEMS GRID

order: es la propiedad que especifica el orden de los elementos dentro de la grid, el valor por defecto de todos los elementos es 0, y admite valores negativos y positivos.

ejemplo:

```
.grid-order{
    display:grid;
    grid-template-columns: repeat(5, 1fr);
    grid-template-rows:repeat(2, 1fr);
}
.items-order:last-child{
    order:-2;
}
```

en cuanto a la alineacion de elementos dentro de la grid tenemos las propiedades:

justify-items: main axis(x) /* valores: stretch, start, center, end, basilene*/

align-items: cross axis(y) /* valores: stretch, start, center, end, basilene*/

place-items: es un shordhand que engloba estas dos propiedades, donde el primer valor es para align-items y el segundo para justify-items, separados por espacio, si solo se coloca un valor lo tomara en ambos ejes**--ojo**

en cuanto a las propiedades de los hijos si queremos moverlos independientemente usamos las propiedades:

justify-self

align-self

place-self: es un shordhand que engloba las propiedades align-self y justify-self en ese mismo orden se aplica

funciona exactamente igual con los mismo valores, pero la diferencia es que se aplica al mismo elemento hijo.

ejemplo:

```
.grid-align{
   display: grid;
   grid-template-columns: repeat(3, 250px);
   grid-template-rows: repeat(2, 200px);
   justify-items: end;
   align-items: end;
}


.align-grid:last-child{
   background-color: #26cd42;
   justify-self: center;
   align-self: center;
}
```

11. ALINEACIÓN DE GRID TRACKS

Se utiliza para la alineación completa de la grid (todo el contendió dentro de un contenedor), siempre y cuando tenga espacio disponible.

las propiedades que se utilizan son:

justify-content: eje x

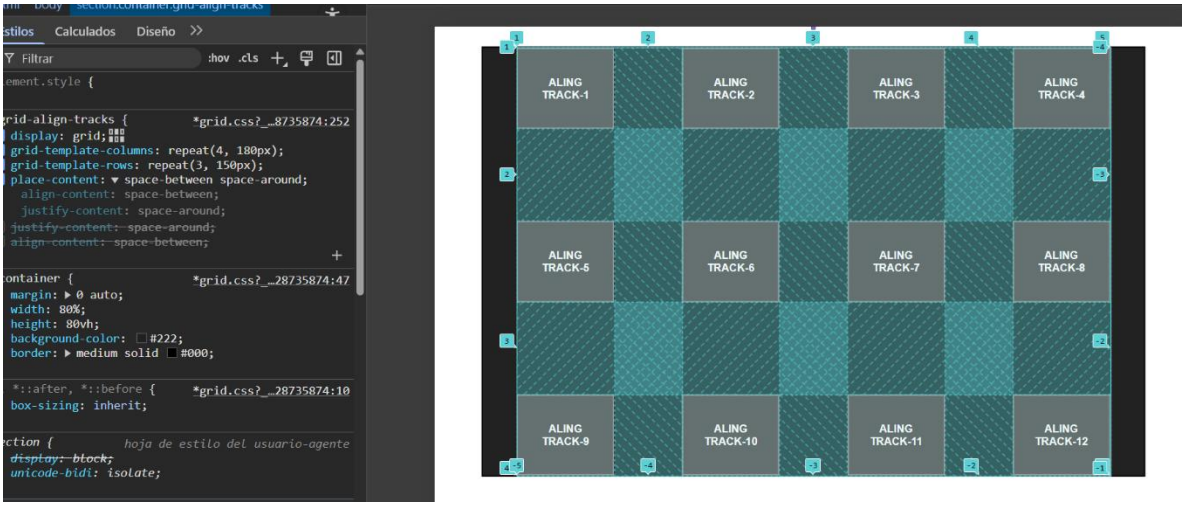
align-content: eje y

place-content: es un shordhan que que agrupa estas dos propiedades, donde el primer valor es align-content y el segundo valor es justify-content separado por espacios, si solo se coloca un valor lo tomara para ambos ejes--**ojo**

los valores que utiliza son:

start, center, end, space around, space-between, space-evelyn

ejemplo:



12. Tamaños Máximos y Mínimos de Grid Tracks

Existe una funcion donde podemos definir el valor minimo y maximo dentro de la grid

minmax (valor minimo, valor maximo); estos valores pueden ser medidas absolutas o relativas.

o podemos utilizar los valores:

min-content: que representan el minimo contenido

max-content: que representan el maximo contenido

estos nos permite adaptar las columnas y filas a los diferentes contenidos

ejemplo:

```
.grid-min-max{
   display:grid;
   grid-template-columns: repeat(4, minmax(min-content, max-content));
 /* Donde establecimos lorem ipsum y se establecido que se adaptara a su minimo y su maximo contendo */
}
```

13. Grids con Patrones Repetitivos

Con la funcion **repeat()** podemos crear patrones repetivos definiendo, cuantas veces quiero que repite el patron explicito!

ejemplo:

```
.grid-repeat{
   display: grid;
   grid-template-columns: repeat(2, 12.5rem 10rem 12.5rem 8rem ); /* donde repetira el patron de 4 columnas 2 veces*/
}
```

14. Grids Dinámicas. Responsive sin Media Queries

para crear grid dinamicas existen dos palabras claves en las propiedades en la propiedad grid-template-columns o grid-template-rows estas palabras claves son: auto-fit y auto-fill

auto-fill: Llenado automatico, se utiliza para indicar que se deben crear tantas columnas o filas como sea necesario para rellenar el espacio disponible dentro de la cuadrícula.

auto-fit: se utiliza para ajustar automáticamente el número de columnas o filas en función del contenido. En lugar de definir explícitamente el número de columnas o filas, auto-fit permite que la cuadrícula se ajuste al espacio disponible, creando columnas o filas adicionales según sea necesario.

ejemplo:

```
.grid-dynamics{
   display: grid;
   grid-template-columns: repeat(auto-fit, 12.5rem)/* le estamos diciendo crea automaticamente las columnas segun tu contendio y que sean de 200px y ajustate a eso*/
}
```

caso auto-fill

```
.grid-dynamics{
   display: grid;
   grid-template-columns: repeat(auto-fill, 8em)/* le estamos diciendo crea automaticamente las columnas hasta llenar el espacio disponible del contenedor */
}
```

responsive practice

```
/*  RESPONSIVAS  */
.grid-responsive{
   display: grid;
   grid-template-columns:repeat(auto-fill, minmax(min-content, 12.5rem));
```

}

otro ejemplo:

.grid-repeat{

display: grid;

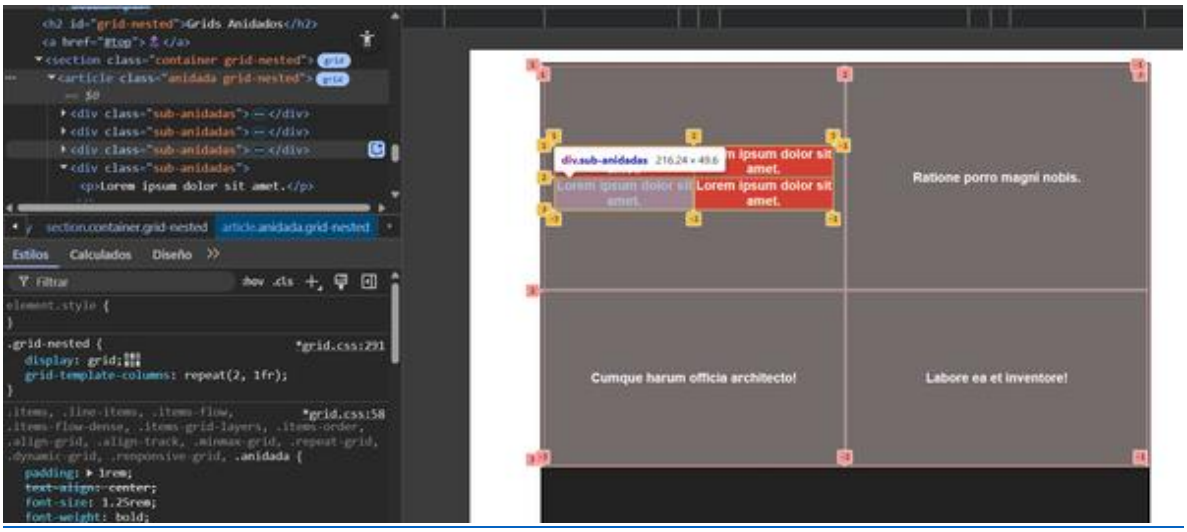
grid-template-columns: repeat(4, min-content, max-content);/* donde repetira 2 columnas con contenido minimos y otro contenido maximo*/

}

15. Grids Anidadas & Subgrids

es una técnica que permite organizar elementos dentro de una estructura de cuadrícula, donde una cuadrícula principal contiene otras cuadrículas más pequeñas, creando una estructura jerárquica. Esto facilita la creación de diseños web complejos y dinámicos, permitiendo organizar contenido de manera flexible y adaptable.

Ejemplo:



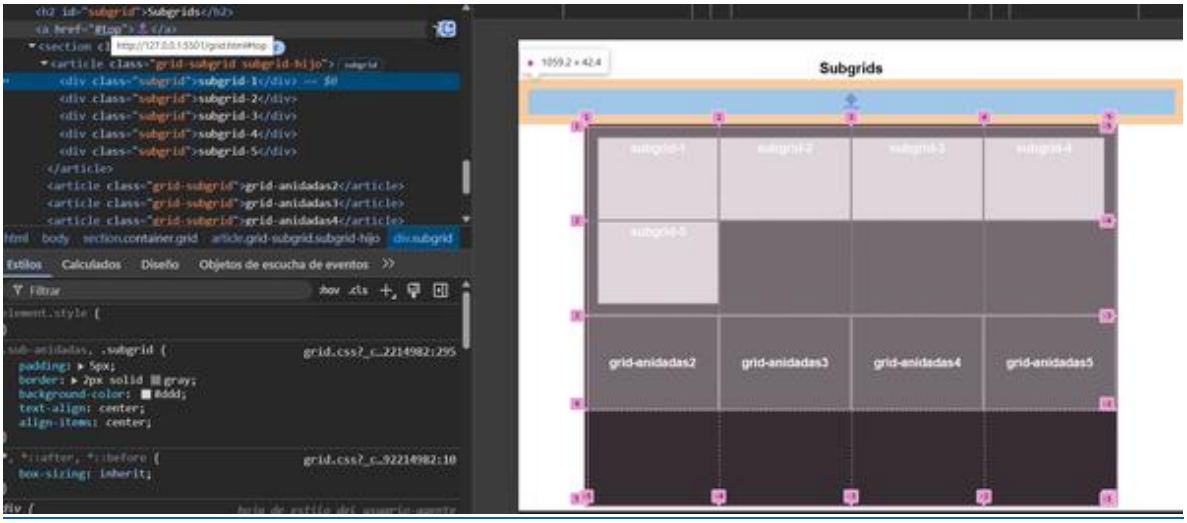
nota: para que funcione la propiedad subgrid es necesario que se aplique los siguientes pasos:

1. definir explícitamente el tamaño que ocupara elemento contenedor (columns y rows)
2. aplicar grid al elemento que sera subgrid

3. aplicar el valor subgrid a las propiedades grid-template-columns o grid-template-rows dependiendo lo que necesites definir

ejemplo:

```
.subgrid{
   display: grid;
   grid-template-columns: subgrid;
   grid-template-rows: subgrid;
}
```



nota: tambien se le puede aplicar a los subnietos