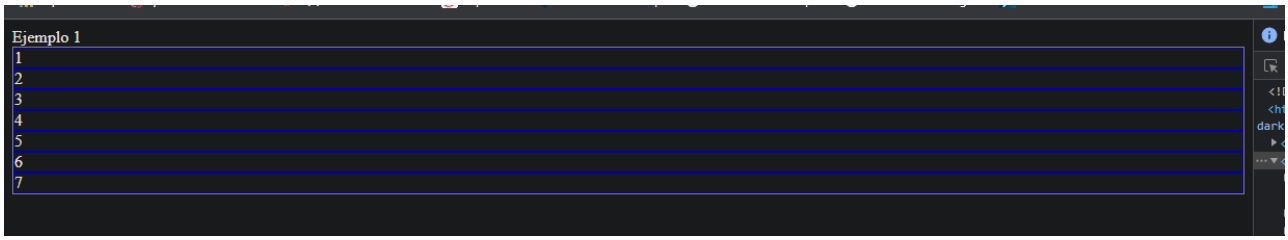
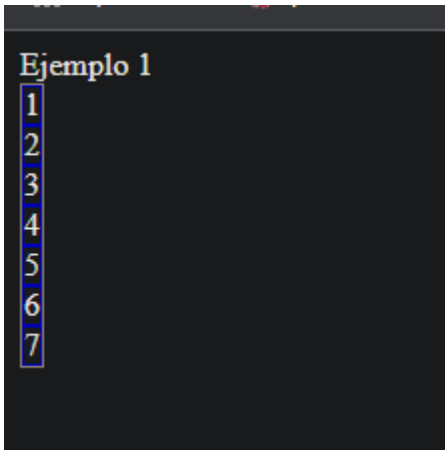
- *display:grid*
  Grid con el ancho 100%

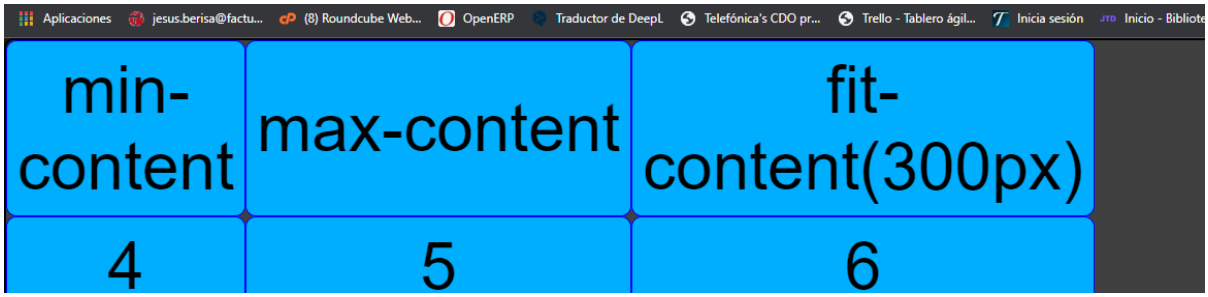


- *display:inline-grid*
  Grid pero con el ancho definido por el contenido



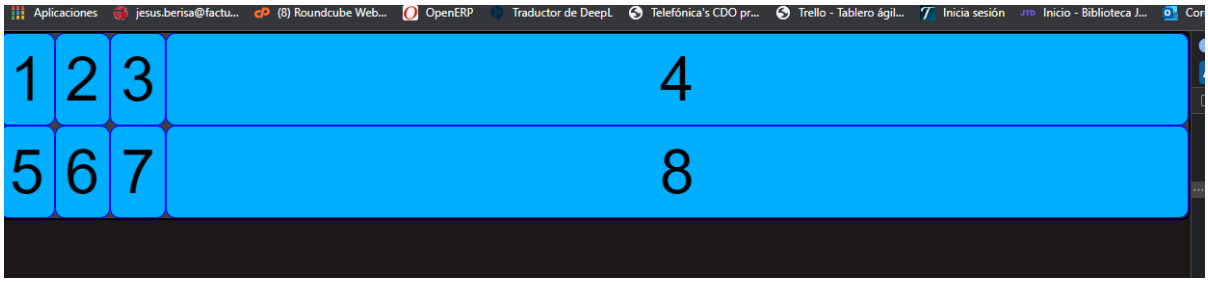
- *grid-template-columns*
  El tamaño o ancho que queremos que tenga cada columna.
  Se puede hacer con pixeles, procentajes.

- - *min-content, max-content, fit-content*
    min-content → El tamaño sera el de la palabra mas ancha del texto
    max-content → Estirar el contenido al maximo para que no haga salto de linea
    fit-content → Como minimo usara el min-content y puedes crecer hasta el numero que le he introducido o hasta que llegue el max-content



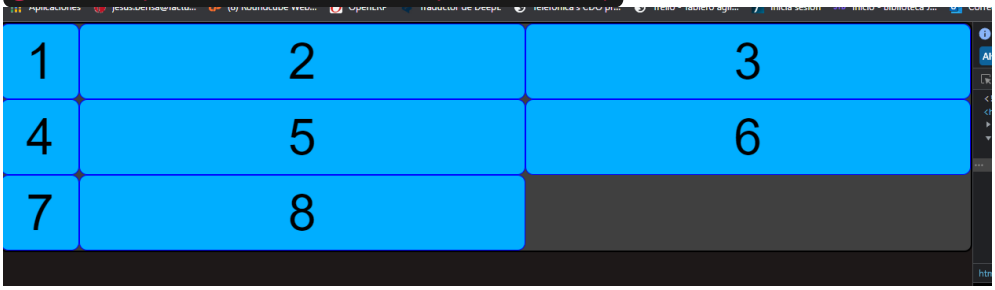
> Si por ejemplo pongo fit-content(400000px) la caja crecera hasta el max-content del contenido y como minimo se aplicara el min-content del contenido

- - *auto (Recomendado usar fracciones antes que esto.)*
  Ancho automatico.



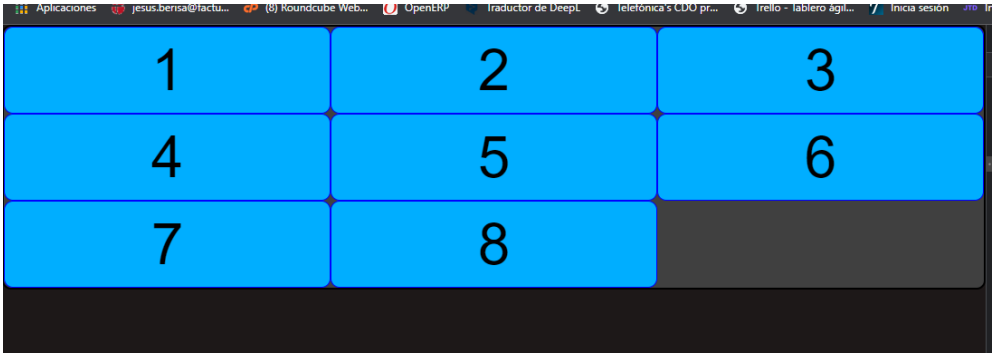

```
grid-template-columns: min-content max-content fit-content(300px) auto;
```

- - *Fracciones*
  Podemos definir cuanto espacio le damos a cada caja del total que hay.

> Ejemplo: De 3 columnas le estoy estoy diciendo que la primera sea de 100px, la segunda y la tercera pillen el espacio restante y lo dividan entre 2
>
> ```
> grid-template-columns: 100px 1fr 1fr;
> ```
>
> 
>

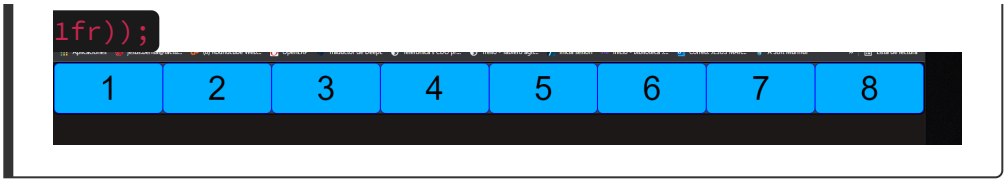
- - *minmax()*
  Le digo que tamaño minimo y maximo puede tener la caja.

> Ejemplo:
> ```
> grid-template-columns: minmax(200px, 1fr) 1fr 1fr;
> ```
> Minimo 200px, maximo 1fr



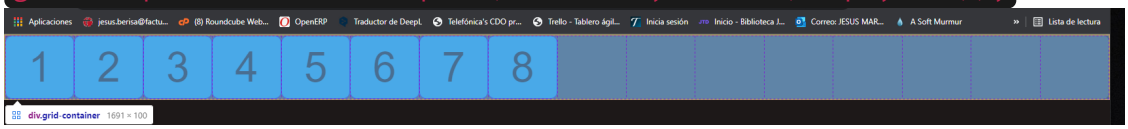
- - *Repeat*
  Para no escribir tanto.
  `grid-template-columns: minmax(200px, 1fr) 1fr 1fr;`
  es lo mismo que
  `grid-template-columns: minmax(200px, 1fr) repeat(2, 1fr);`
- - - *auto-fit*
    Sirve para hacer responsible las cajas.
    Siempre pilla todo el espacio de la linea.

> Ejemplo:
> Lo que estoy diciendole aqui es que haga las con un minimo de 400 pixeles y si hay hueco que meta otra caja mas.
> `grid-template-columns: repeat(auto-fit, minmax(400px,`



- - - *auto-fill*
    Sirve para hacer responsible las cajas.
    La diferencia con la anterior es que si hay hueco en vez de usar todo el espacio, creara cajas invisibles para rellenar el espacio.

```
grid-template-columns: repeat(auto-fill, minmax(400px, 1fr));
```

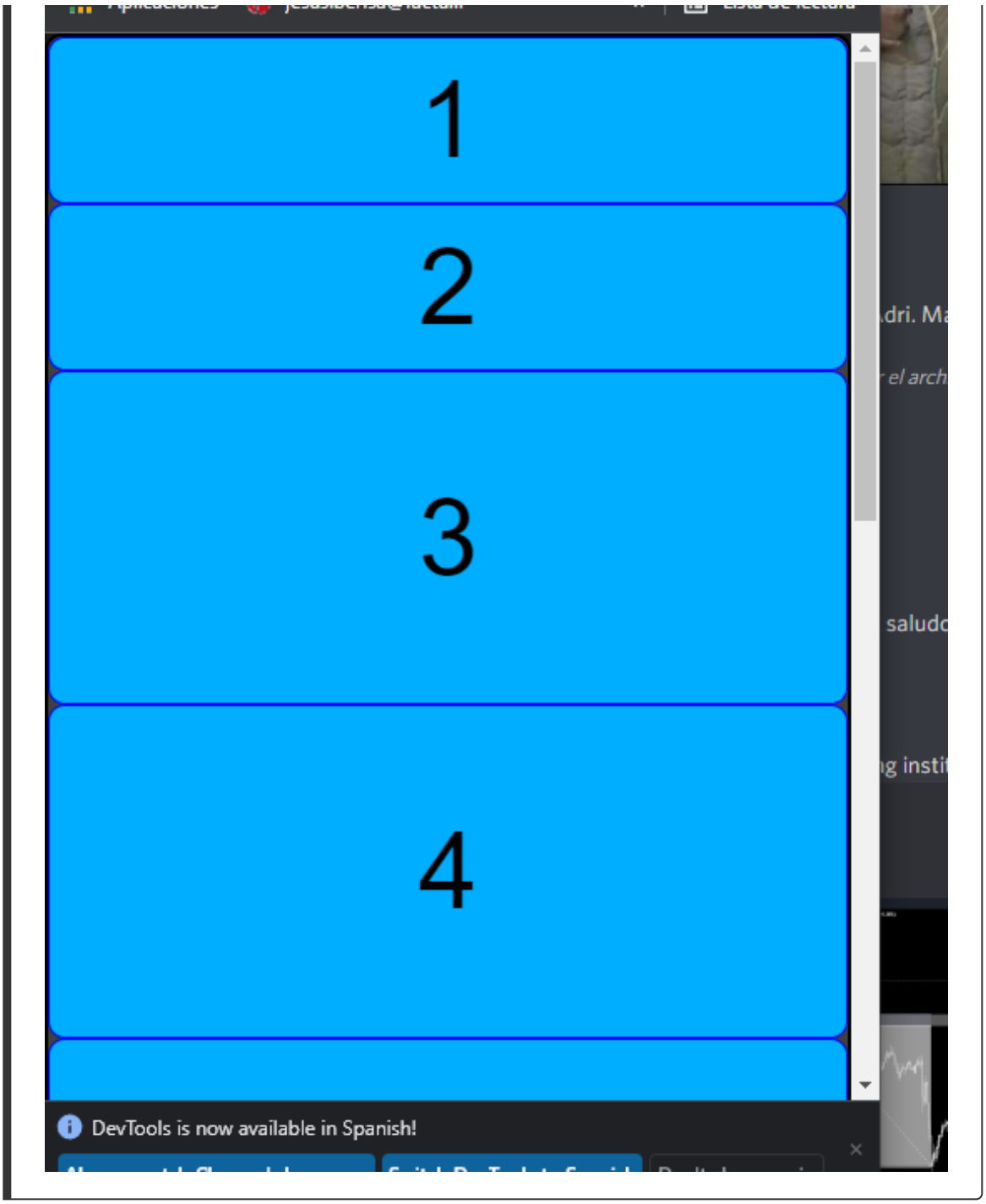


- *grid-template-rows*
  Sirven las mismas propiedades de grid-template-column.

- *grid-auto-columns && grid-auto-rows*
  Sirve para definir las nuevas o existentes columnas/lineas el tamaño.

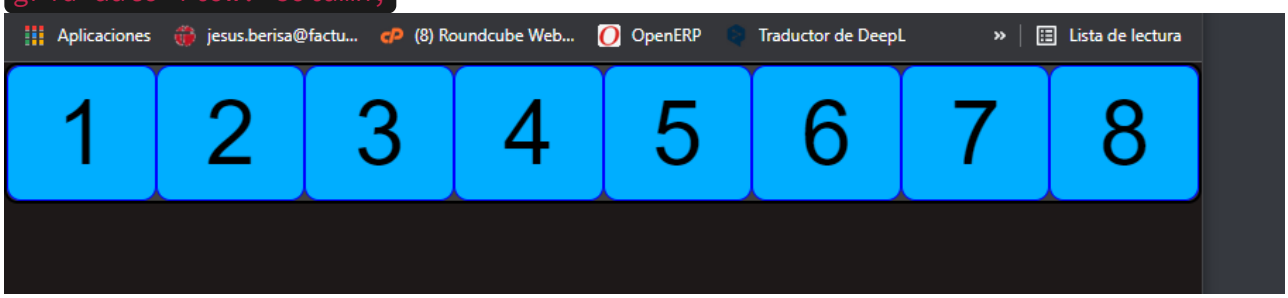
```
grid-template-rows: 100px 100px;
grid-auto-rows: 200px;
```

> Aqui a partir de la tercera row el tamaño que tendran sera de 200px cada nueva row



- *grid-auto-flow*
  Direccion a donde tienen que ir los elementos.

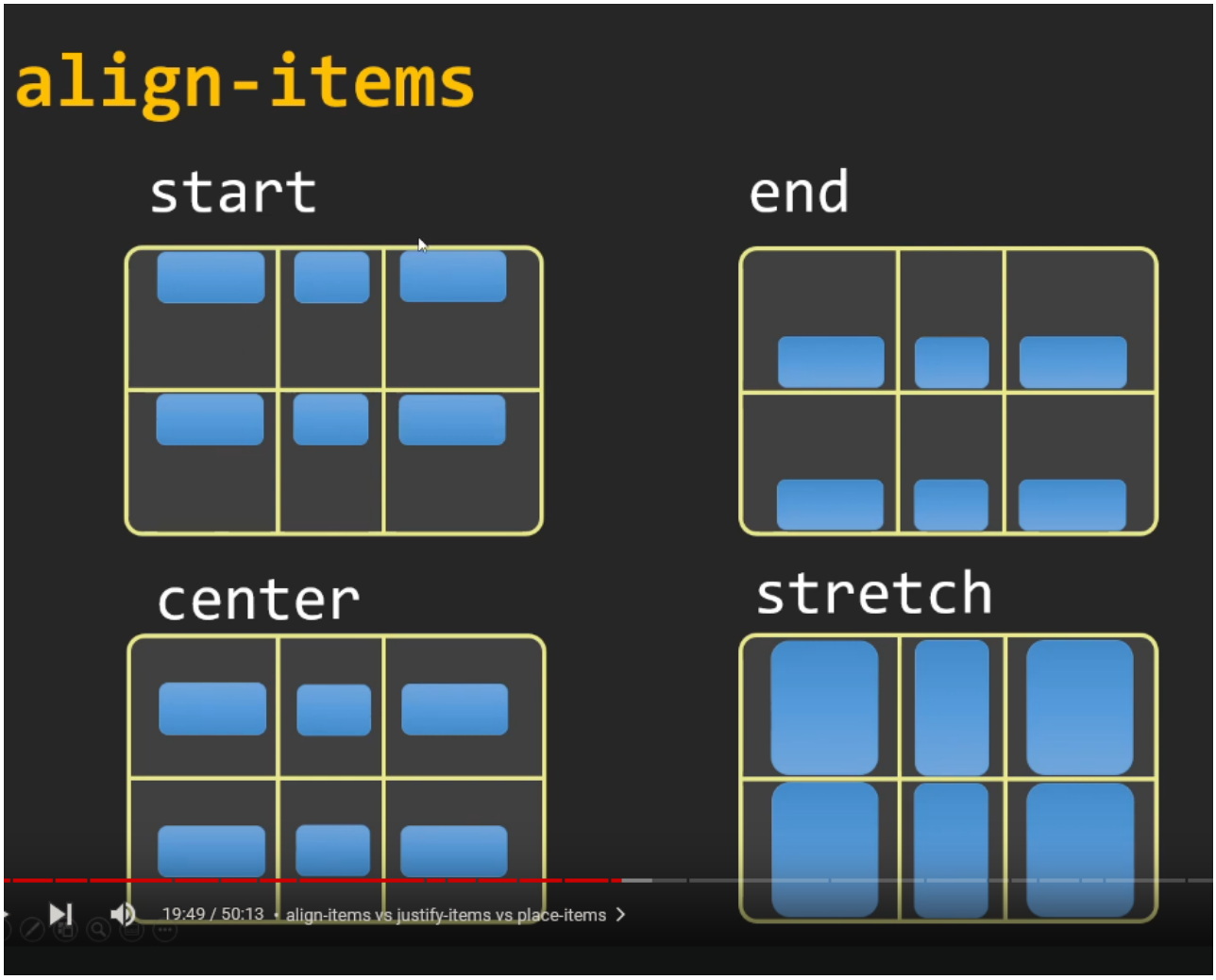
```
grid-template-columns: repeat(1fr);
grid-auto-flow: column;
```

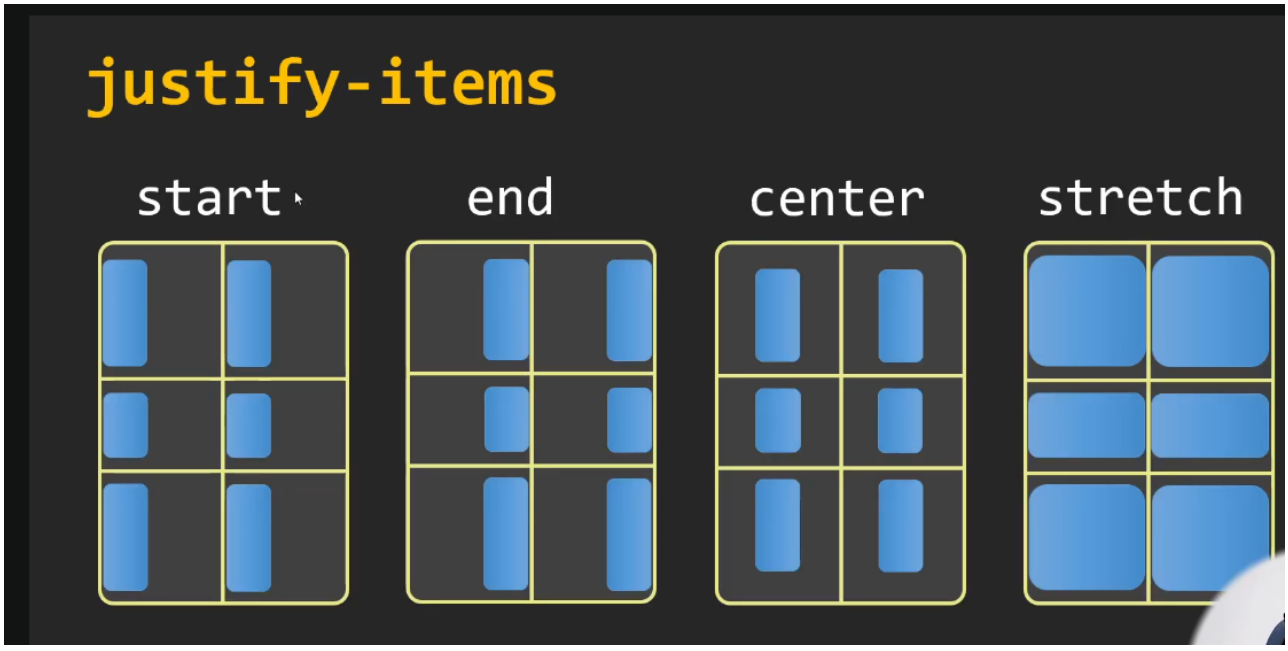


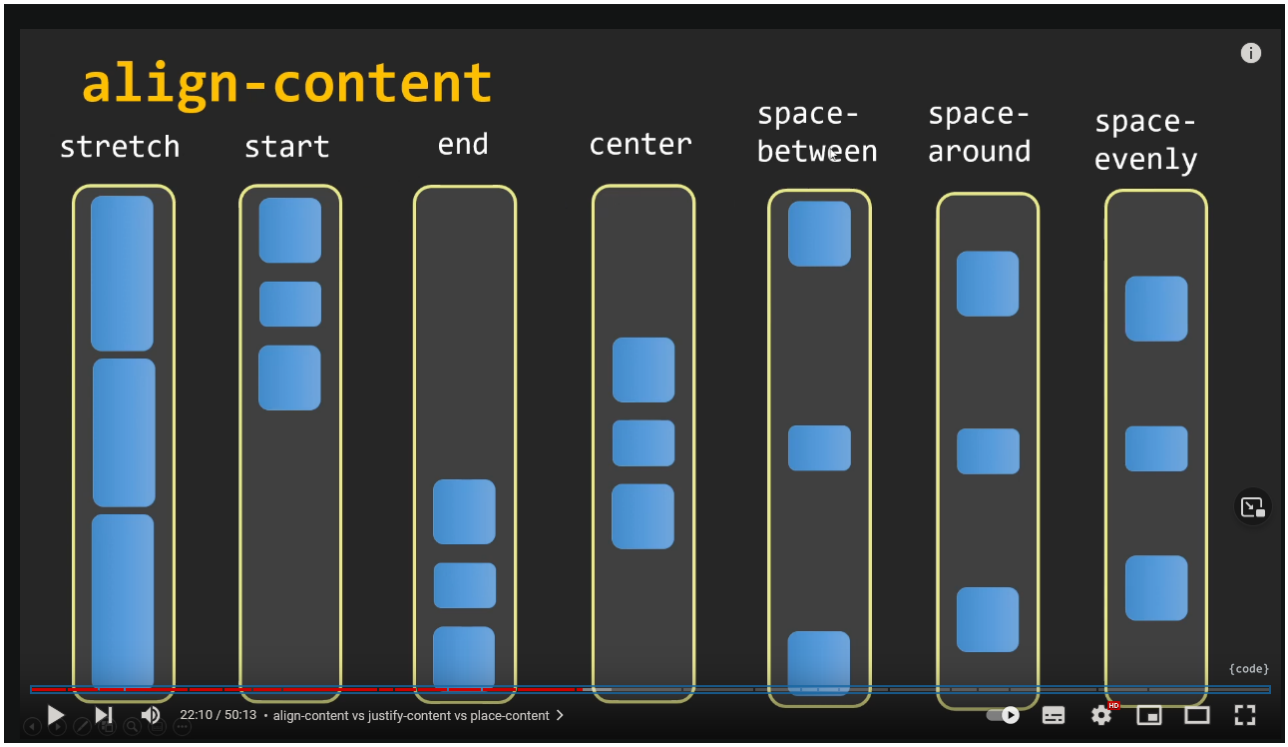
```
grid-template-columns: repeat(1fr);
grid-auto-flow: row;
```

- *row-gap*, *column-gap*, *gap*
  Separacion entre columnas, filas

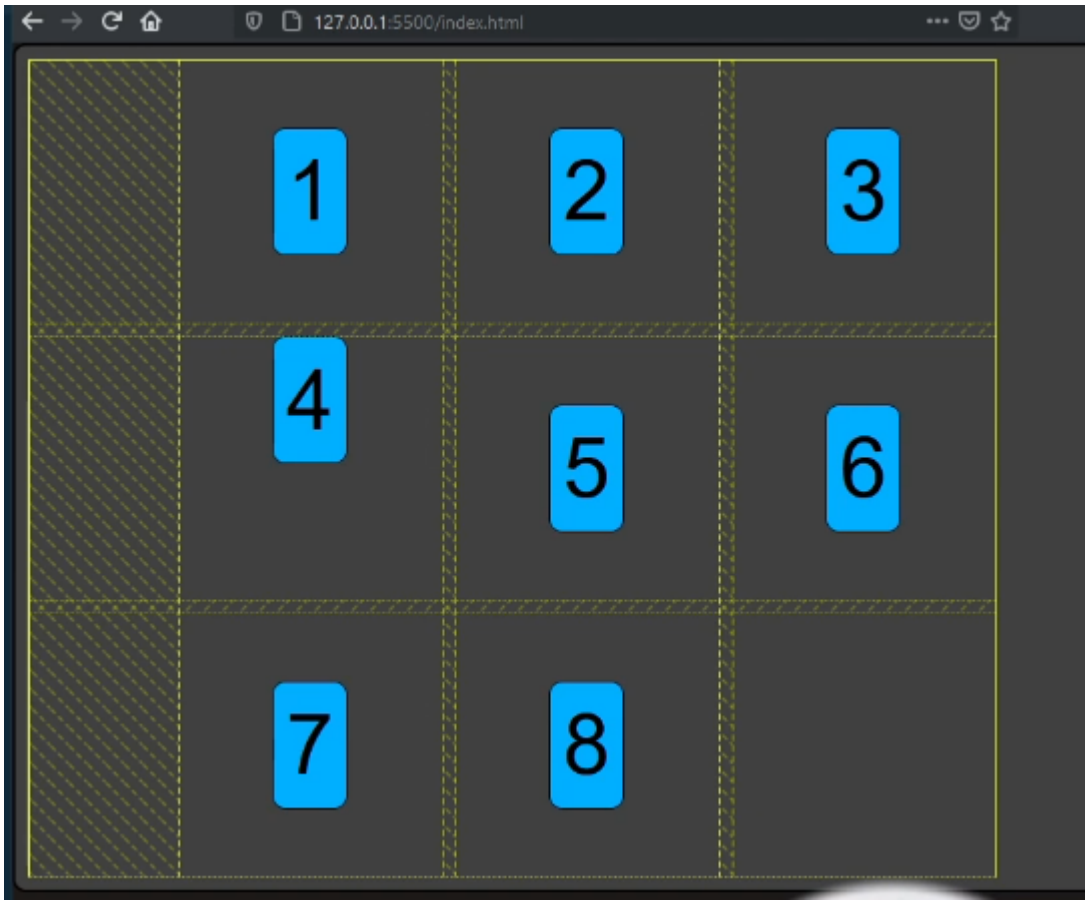
- *align-items & justify-items & align-content & place-items*
  `Place items: align-items justify-items`
  Algo importante es el **stretch** que sirve para que el contenido pille todo el espacio disponible


align-items
start
end
center
stretch
19:49 / 50:13 • align-items vs justify-items vs place-items


justify-items
start
end
center
stretch



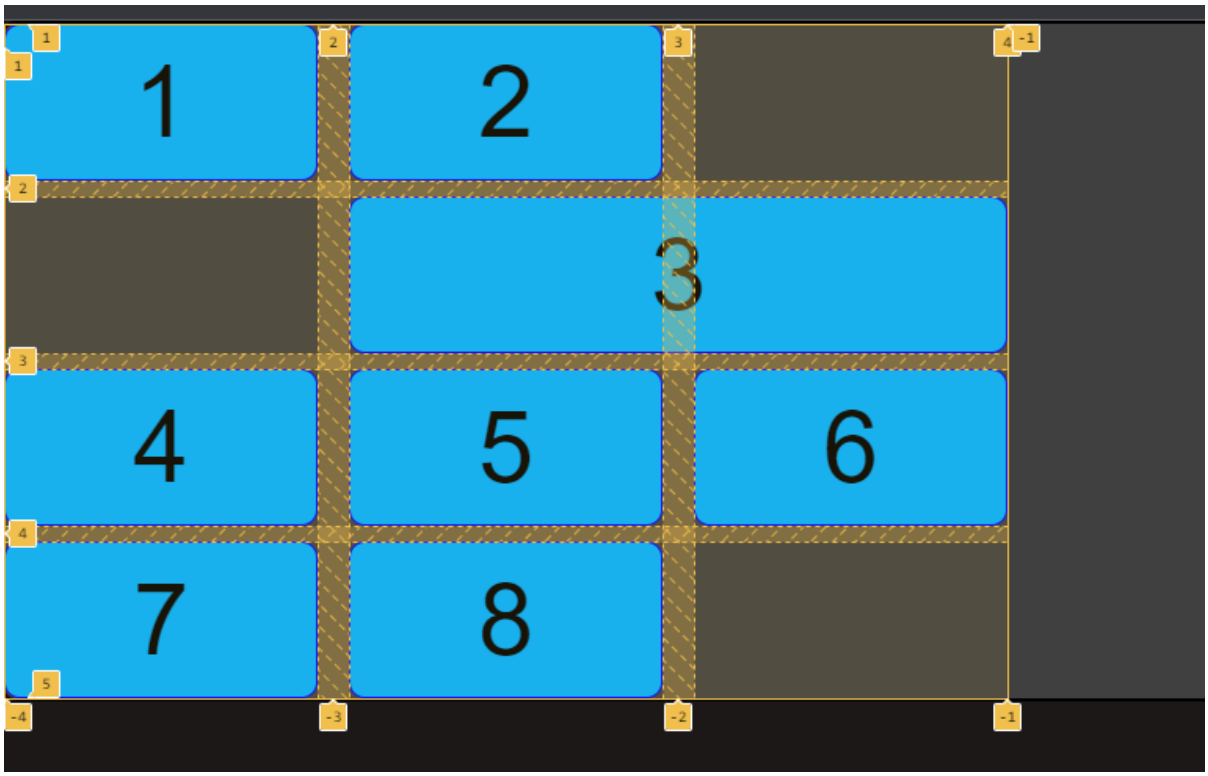
- *align-self & justify-self & place-self*
  Sirve para alinear de forma distinta a lo que dice el padre



- *Como posicionar objetos de forma manual*

- *grid-column-start & grid-column-end & grid-column*
  Quiero que la caja 3 empieze en la columna 2 y que acabe en la 4

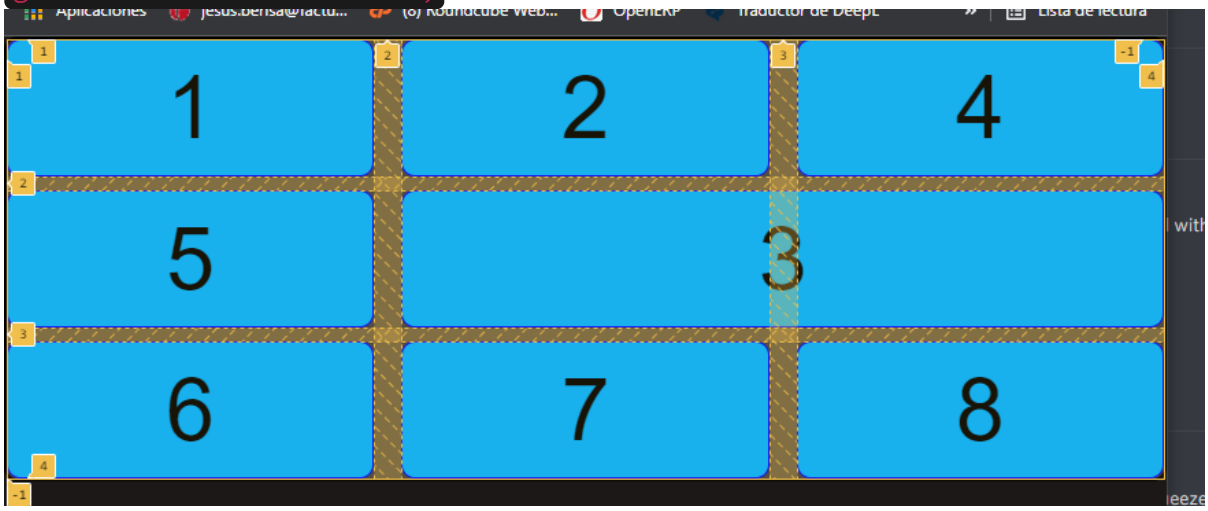
```css
.different{
    grid-column-start: 2;
    grid-column-end: 4;
}
```



> Lo mismo para las rows
> Si quiero pillar toda la linea otra forma seria tal que asi `grid-column: 1/-1`

- *dense*
  Si se quedan huecos vacios el dense sirve para rellenarlos

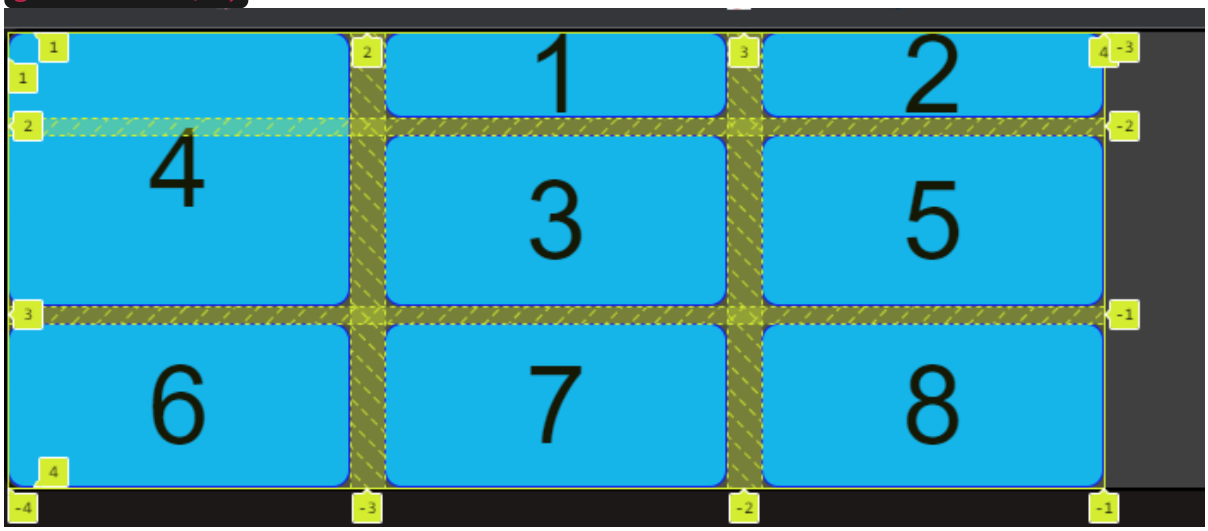
```
grid-auto-flow: dense row;
```



- *alias*

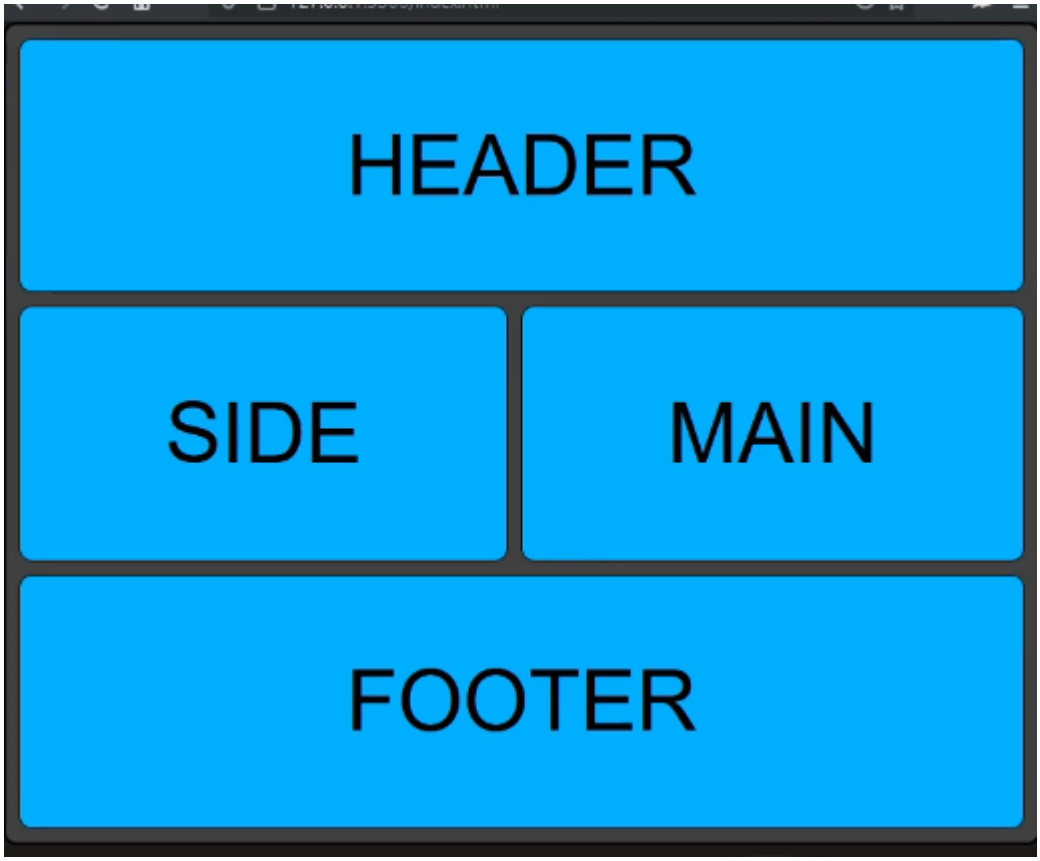
  Ponerle alias a las filas/columnas para poder posicionarlas mejor.

```
grid-template-rows: [a] 50px [b] 100px [c];
grid-row: a/c;
```



- grid-template-areas*

  Se quiere llegar a este resultado




```
    grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer"
    ;
}
.header{
    grid-area: header;
}
.sidebar{
    grid-area: sidebar;
}
.main{
    grid-area:main;
}
.footer{
    grid-area:footer;
}
```

> Curiosidad
> Para dejar un hueco en blanco hay que poner un . en el lugar correspondiente

```
grid-template-areas:
". header"
"sidebar main"
"footer footer"
;
```