

OverFlow

Controla qué sucede si el contenido se desborda del tamaño definido del contenedor.

en HTML

```
<div class="overflow-box">Texto muy largo sin espacios para demostrar overflow...</div>
```

en CSS

```
1 .overflow-box {  
2   width: 200px;  
3   height: 100px;  
4   border: 2px solid black;  
5   overflow: auto; /* otros valores: hidden, scroll, visible */  
6 }
```

- visible: muestra el contenido aunque se salga.
- hidden: oculta lo que desborda.
- scroll: fuerza barra de scroll.
- auto: barra sólo si es necesaria.

White-space

en HTML

```
<p class="nowrap">Este texto es muy largo pero no se va a cortar nunca.</p>
```

en CSS

```
1 .nowrap {  
2   white-space: nowrap;  
3   overflow: hidden;  
4   text-overflow: ellipsis;  
5   width: 200px;  
6   border: 1px solid black;  
7 }
```

Usando fr y repeat()

Reemplazamos px por unidades fluidas:

```
1 .grid {  
2   display: grid;  
3   grid-template-columns: repeat(3, 1fr); /* 3 columnas de igual tamaño */  
4   gap: 20px;  
5 }
```

El fr significa 'fracción'. Es una forma flexible de dividir el espacio. repeat(3, 1fr) es lo mismo que 1fr 1fr 1fr

Controlando filas también:

```
1 .grid {  
2   grid-template-columns: repeat(3, 1fr);  
3   grid-template-rows: 100px 150px;  
4 }
```

Tipos de valores para grid-template-columns

- px, em, rem → fijos
- % → relativo al ancho del contenedor
- fr → fracciones
- auto → se adapta al contenido
- min-content, max-content → se adaptan a contenido mínimo/máximo

¿Por qué posicionar manualmente?

Hasta ahora dejamos que el navegador coloque los ítems en orden. Ahora vamos a decirle nosotros dónde queremos cada cosa. Esto es útil para layouts tipo tablero, revistas, dashboards, portadas, etc

En HTML

```
1 <div class="grid">
2   <div class="item a">A</div>
3   <div class="item b">B</div>
4   <div class="item c">C</div>
5   <div class="item d">D</div>
6 </div>
```

En CSS

```
1 .grid {
2   display: grid;
3   grid-template-columns: repeat(4, 1fr);
4   grid-template-rows: repeat(3, 100px);
5   gap: 10px;
6 }
7
8 .item {
9   background: #3498db;
10  color: white;
11  font-size: 2rem;
12  display: flex;
13  align-items: center;
14  justify-content: center;
15  border-radius: 8px;
16 }
17
18 .a {
19   grid-column: 1 / 3; /* Ocupa columnas 1 y 2 */
20   grid-row: 1 / 2;    /* Solo fila 1 */
21 }
22
23 .b {
24   grid-column: 3 / 5; /* Ocupa columnas 3 y 4 */
25   grid-row: 1 / 3;    /* Fila 1 y 2 */
26 }
27
28 .c {
29   grid-column: 1 / 2;
30   grid-row: 2 / 4;
31 }
32
33 .d {
34   grid-column: 2 / 4;
35   grid-row: 3 / 4;
36 }
```

- grid-column: 1 / 3 → empieza en la columna 1 y termina antes de la 3.
- grid-row: 2 / 4 → ocupa desde la fila 2 hasta la 3 inclusive.

minmax()

Permite definir un tamaño mínimo y máximo para una columna o fila.

```
1 .grid {  
2   grid-template-columns: repeat(3, minmax(200px, 1fr));  
3  
4 }  
5
```

auto-fit = intenta encajar la mayor cantidad posible de columnas que cumplan con el tamaño mínimo (en este caso 250px). Si sobra espacio, lo distribuye.

```
1 .grid {  
2   grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));  
3  
4 }
```

auto-fill = hace lo mismo pero reserva espacios vacíos aunque no haya contenido (ideal para layouts tipo cuadrícula donde todo debe quedar alineado).

```
1 .grid {  
2   grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));  
3 }  
4
```

Ambos generan grillas fluidas automáticas, sin necesidad de media queries para cada tamaño

Responsive con media queries + Grid

```
1 .grid {  
2   display: grid;  
3   grid-template-columns: 1fr 1fr;  
4   gap: 20px;  
5 }  
6  
7 @media (max-width: 768px) {  
8   .grid {  
9     grid-template-columns: 1fr;  
10    }  
11 }
```

Este patrón es super útil para secciones tipo:

- Cards de productos
- Bloques de contenido
- Secciones tipo "sobre nosotros" o testimonios