¿Qué es CSS Grid? (parte I)

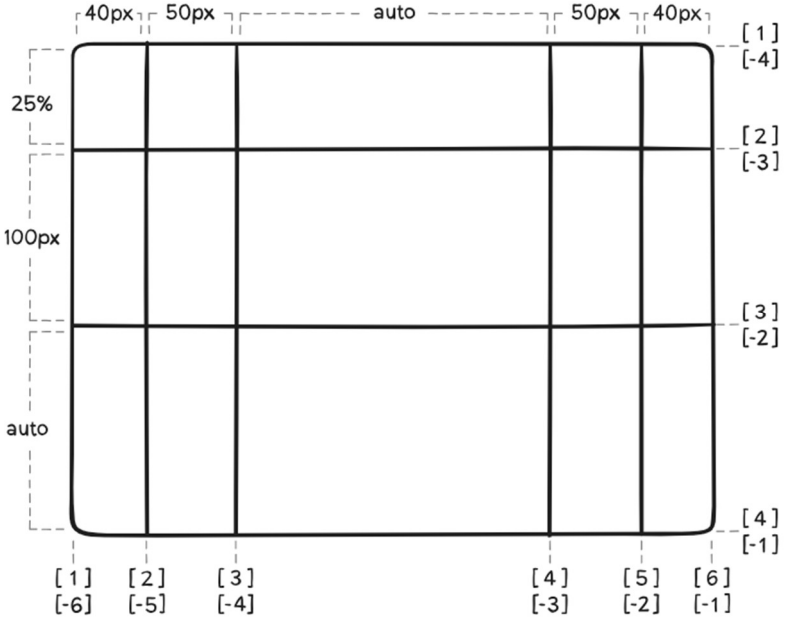
Rita de la Torre

CSS Grid es la técnica más reciente de CSS3 para la disposición en varias dimensiones, de los elementos dentro de un contenedor flexible definido con la propiedad `display: grid;` o `display: inline-grid;`

Inicialmente, tiene la misma filosofía de Flexbox, es decir, un contenedor padre con elementos hijos (llamados **grid items**) que a diferencia de los elementos "flexbox items" se colocan dentro de áreas definidas en la cuadrícula que se establece en el contenedor padre. Esto permite diseñar un layout más estructurado y controlado.

Las áreas de la cuadrícula se pueden nombrar. Estos nombres de las áreas en CSS Grid pueden ser totalmente personalizados.

Al definir áreas dentro de la cuadrícula, es como si se estuviera creando "capas" de contenido organizadas. A esto se suma la alineación de los elementos dentro de cada área, lo que añade otro nivel de control (tres niveles).



Las tres capas de control de GRID

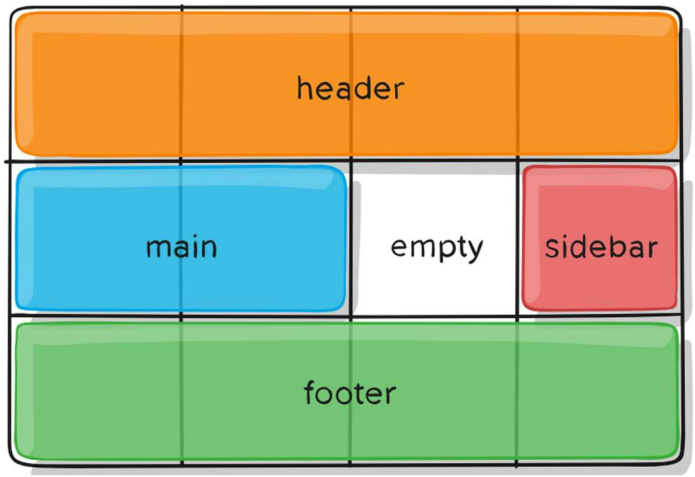
Rita de la Torre

La estructura de CSS Grid es la siguiente:

Container: es el elemento principal, que usa `display: grid;`
La propiedad **`display`** nunca se hereda

Áreas: Nombres asignados con la propiedad `grid-template-areas`, y que luego los **grid-items** pueden usar con la propiedad `grid-area`.

Grid-items: Elementos hijos dentro del contenedor ubicados en un área determinada.



Si lo pensamos en términos de una casa:
El **container** es la casa (la estructura principal).
Las **áreas** son los espacios en la casa (sala, cocina, dormitorio).
Los **grid-items** son los muebles (cada elemento dentro de la casa).

Cada mueble (**grid-item**) se coloca en un espacio de la casa (área).

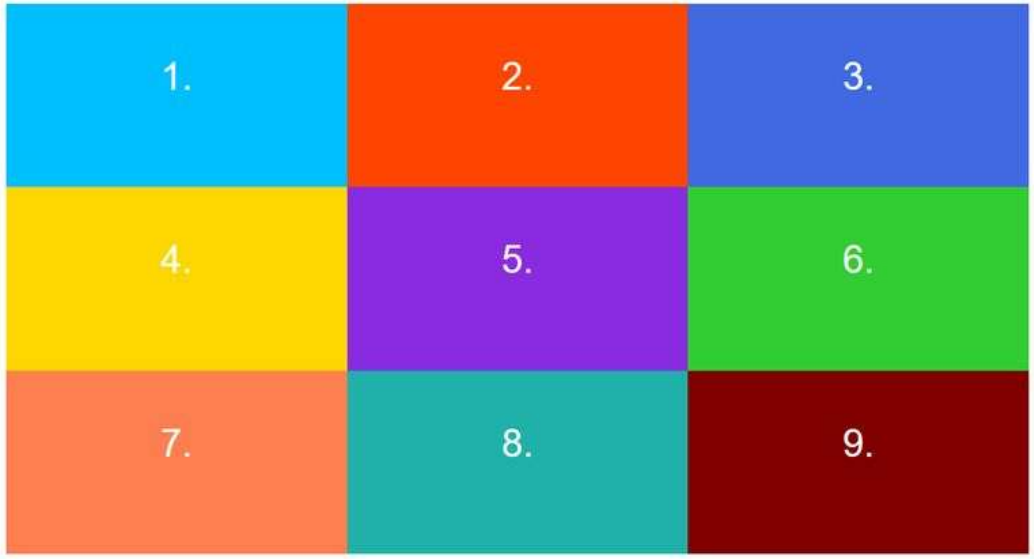
Sin embargo, no es obligatorio nombrar las áreas en CSS Grid. Se puede estructurar el diseño sin usar `grid-template-areas` y, en su lugar, posicionar los **grid-items** directamente con propiedades como `grid-column` y `grid-row`.

Pasos antes de definir la cuadrícula en CSS Grid

Rita de la Torre

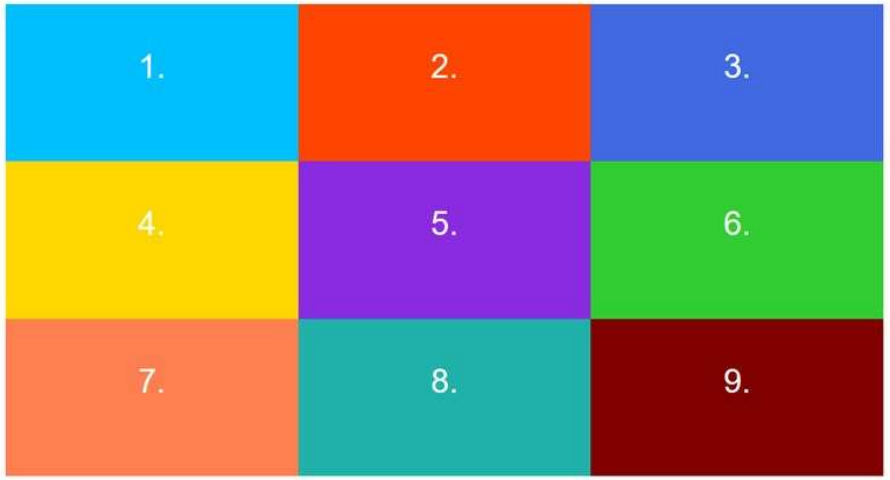
Antes de empezar con CSS Grid, lo mejor es diseñar la estructura general de la página para distribuir los elementos correctamente. Pasos:

1. **Analizar el diseño**: ¿Qué estructura se necesita? ¿Será un layout con sidebar, encabezado y contenido?
2. **Dibujar la cuadrícula**: Ya sea en papel o en una herramienta digital, hay que estar claro en cuántas **filas** y **columnas** serán necesarias.
3. **Decidir si usar nombres de áreas o posicionamiento manual.** ¿Prefieres definir secciones como "header", "contenido", "footer", o trabajar con grid-column y grid-row?
4. **Definir grid-template-columns y grid-template-rows** para establecer el tamaño de cada celda en la cuadrícula.



Definiendo las propiedades del contenedor Grid

Rita de la Torre



El contenedor grid es el elemento que define la cuadrícula.
Las propiedades que permiten definir la cuadrícula son:

```
.container {
  height: 90vh;
  margin: 2rem;
  display: grid; /* Activa la cuadrícula  */
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 1fr 1fr;


  grid-template-areas:
    "header header header"
    "aside main main"
    "footer footer footer";
}
```

OPCIONAL
Se usa para posicionar por áreas

`grid`: Es un atajo que combina `grid-template-rows` y `grid-template-columns`

Funciones para optimizar el grid template

Rita de la Torre

`repeat()` Permite repetir columnas o filas en `grid-template-columns` y `grid-template-rows`.

`minmax()` Define un tamaño mínimo y máximo para las columnas/filas en `grid-template-columns` y `grid-template-rows`.

```css
.container {
  height: 90vh;
  margin: 2rem;
  display: grid; /* Activa la cuadrícula  */
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 1fr 1fr 1fr;
 }
```

`fr` en **CSS Grid** significa **fracción** y es una unidad especial que se usa para dividir el espacio disponible de una cuadrícula de manera proporcional.

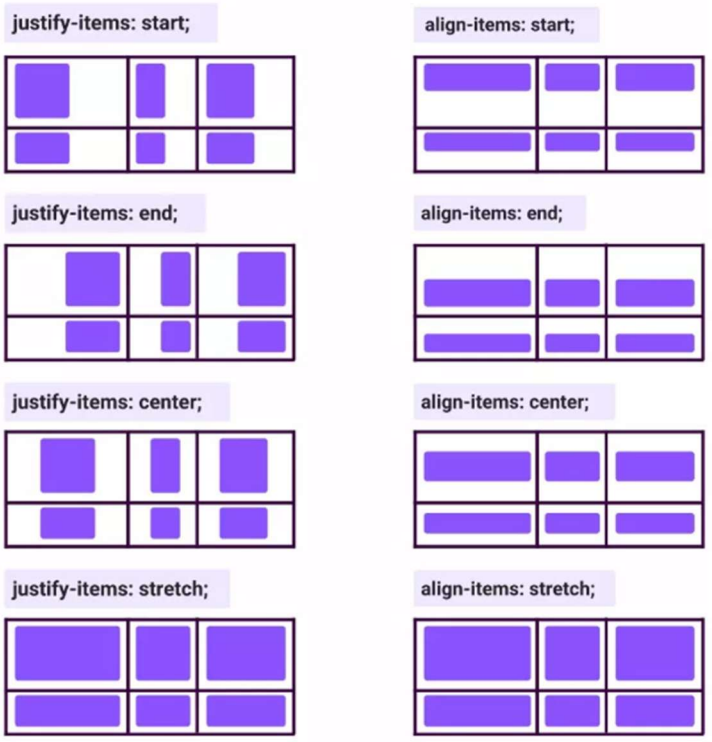
```css
.container {
  height: 90vh;
  margin: 2rem;
  display: grid;
  grid-template-columns: repeat(3, minmax(150px, 1fr))
  grid-template-rows:    repeat(3, auto);
 }
```

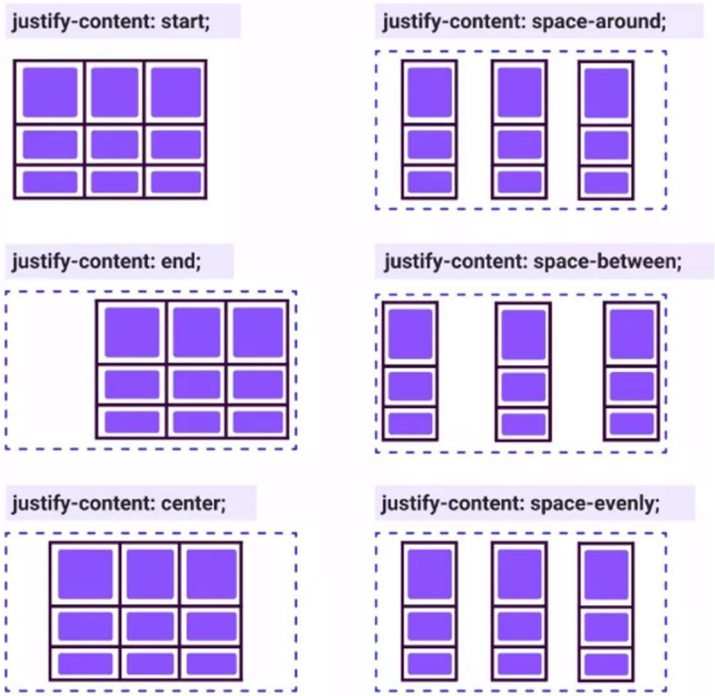
Las columnas tendrán como mínimo si se reduce el ancho de la pantalla 150px y como máximo podrán ocupar hasta 1 fracción del espacio disponible.

Propiedades del contenedor Grid

`justify-items` alineamiento horizontal y `align-items` alineamiento vertical alinean los grid-items dentro de sus celdas.

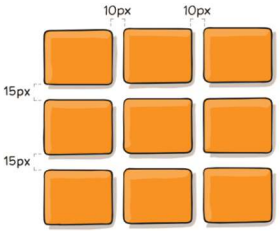
`place-items` es un atajo que combina `align-items` y `justify-items` en una sola propiedad.





`justify-content` alineamiento horizontal y `align-content` alineamiento vertical alinean los grid-items dentro de sus celdas.

`place-content` es un atajo que combina `align-content` y `justify-content` en una sola propiedad.



`gap` es el espacio entre filas y columnas (también `row-gap` y `column-gap`).

Definiendo el posicionamiento de los grid-items (manual)

Rita de la Torre

Para definir los elementos sin usar grid-template-areas, se reemplaza `grid-area` con `grid-column` y `grid-row`, asignando manualmente su ubicación en la cuadrícula.

```
header {
  grid-column: 1 / 4;        /* Ocupa desde la columna 1 hasta la 3 */
  grid-row: 1;               /* Está en la primera fila */
}

aside {
  grid-column: 1;    /* Ocupa la primera columna */
  grid-row: 2;       /* Está en la segunda fila */
}

main {
  grid-column: 2 / span 2;   /* Ocupa 2 columnas */
  grid-row: 2;               /* Está en la segunda fila */
}

footer {
  grid-column: 1 / 4;        /* Ocupa desde la columna 1 hasta la 3 */
  grid-row: 3;               /* Está en la tercera fila */
}
```

Se coloca 1 / 4
(el 4 es el límite superior que no se incluye)

Definiendo el posicionamiento de los grid-items (áreas)

Rita de la Torre

Usar nombres de áreas es ideal para:

- Diseños con estructura clara y bien definida (como una página web con secciones fijas).
- Diseños en los que cada elemento tiene un propósito y posición lógica en la cuadrícula.

Ventaja: Más fácil de leer y modificar.

Desventaja: Puede volverse rígido si se necesita cambiar posiciones dinámicamente.

```
.container {
  ...
  grid-template-areas:
    "header header header"
    "aside main main"
    "footer footer footer";
}
```


```
header {
  grid-area: header;
}

aside {
  grid-area: aside;
}

main {
  grid-area: main;
}

footer {
  grid-area: footer;
}
```

Propiedades de los contenedores hijos: **grid-items**

Rita de la Torre

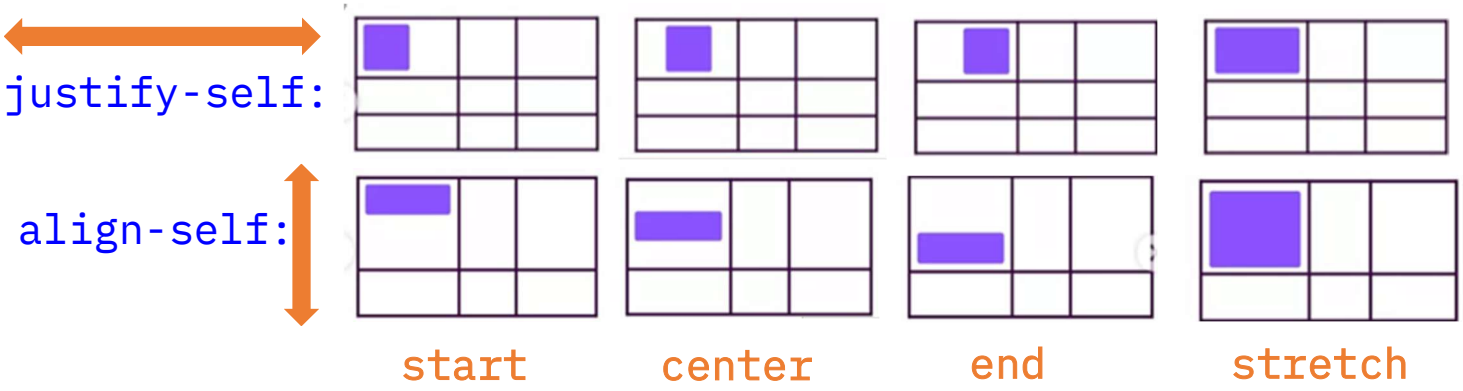
Posicionamiento manual ➔

`grid-column`,
`grid-row`,
`span:` debe ir dentro de `grid-column` o `grid-row`, nunca como una propiedad separada.

Posicionamiento por área ➔

`grid-area`

Alineación interna



`place-self:` Atajo que combina `align-self` y `justify-self` en una sola propiedad.

Copia el ejemplo, de
https://ritadelatorre.github.io/grid/
e intenta cambiar el **`grid-template`**
hasta que entiendas cómo funciona esta técnica.

https://gridbyexample.com/examples/