CSS grid: diseño en rejilla con función inteligente

Un sistema de grid o rejilla, nos permite disponer los elementos de una página y que estén alineados. Va a existir una rejilla imaginaria de filas y columnas, a partir de la cual vamos a colocar los elementos de nuestra web. El uso de un sistema de grid tiene sentido si acompaña un diseño que usa también una rejilla.

¿Para qué sirve lo que vamos a ver en esta sesión?

Un sistema de grid nos sirve para posicionar los elementos de la página alineados. Se usa en un montón de webs, por ejemplo, este ejemplo de Google.

Grid de Google Plus

Podemos ver que los elementos están dispuestos en filas y columnas. Hay 4 columnas que se ven claramente, con un elemento que se expande en 2. Aunque a simple vista parecen no estar alineados en filas, todas las cajas tienen una altura proporcional a una base.

Vamos a ver 2 herramientas que nos facilitan crear una composición basada en grid: CSS grid y Bootstrap.

CSS grid

CSS grid es una nueva característica de CSS que permite tener un sistema de grid de forma nativa en CSS. Es una herramienta compleja, así que vamos a ver las bases de cómo poder usarla.

En primer lugar, existen 2 tipos de elementos, el contenedor del grid y los elementos del grid. En este sentido, es similar a algo que ya conocemos: flexbox.

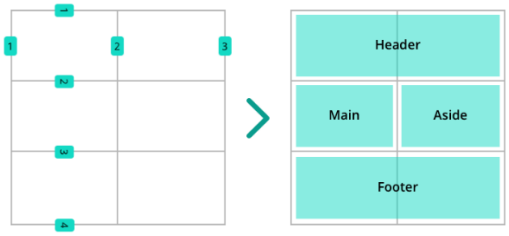
Para comenzar, usaremos en el contenedor la propiedad **display:grid** y definiremos las filas y columnas de nuestro grid con **grid-template-rows** y **grid-template-columns**:

```
.wrapper{
display: grid;
grid-template-columns: 1fr 1fr 1fr 1fr;
grid-template-rows: 40px 200px 40px;
grid-gap: 5px;
}
```

En este grid vamos a tener 4 columnas, cada una de tamaño 1fr, que es una medida sobre el espacio disponible (free space). Por tanto, se divide el espacio disponible en 4 partes para las columnas. Para las filas, tendremos 3 de 40, 200 y 40px respectivamente.
También podemos indicar el tamaño del espaciado de elementos en el contenedor con la propiedad **grid-gap**. Podemos indicar 2 valores si queremos espaciado distinto entre filas y columnas o puedes usar **grid-column-gap** y **grid-row-gap**.

A continuación, indicaremos a los elementos si queremos que ocupen una o varias filas o columnas con las propiedades **grid-column** y **grid-row**.

```
.header {
grid-column-start: 1;
grid-column-end: 3;
grid-row-start: 1;
grid-row-end: 2;
}
```



Esto indica que el item1 se expande desde la primera línea de columna de grid hasta la cuarta, es decir, ocupa las 3 primeras columnas. Hay que tener en cuenta que si tenemos 4 columnas, tendremos 5 líneas de grid. Es decir, siempre vamos a tener una más que el número de filas o columnas. Y desde la línea de fila 2 a la 4, es decir que ocupa la fila 2 y 3.

Podemos escribir lo anterior de una forma simplificada:

```
.item1 {
grid-column: 1 / 4;
grid-row: 2 / 4;
}
```

A la hora de posicionar los elementos en el grid, también podemos usar la palabra span para indicar cuánto se expande desde su posición (fila/columna) actual. Por ejemplo, para un item1 que se expanda desde la fila 1 a la 4 y desde la columna 2 a la 6 podríamos usar:

```
.item1 {
grid-row: span 3;
grid-column: span 4;
}
```

Asignación de áreas

CSS grid layout permite combinar celdas en áreas y nombrarlas. Esto facilita la tarea de dividir los elementos en la rejilla. Los ajustes para esto se hacen en el contenedor. Para ello, usa el comando **grid-template-areas** y escribe los nombres de las áreas deseadas en las celdas línea por línea. Si no quieres asignar una celda y dejarla en blanco, puedes insertar un punto en esta ubicación. Cada fila queda entrecomillada.

```
.grid-container {
  display: grid;
  grid-template-rows: 100px 100px 100px;
  grid-template-columns: 1fr 1fr 1fr 1fr 1fr;
  grid-gap: 5px;
  grid-template-areas:
  "area1 area1 area1 area1 area1"
  "area2 area2 . area3 area3"
  "area2 area2 area4 area4 area4";
}
```

En este ejemplo, hemos definido 4 áreas diferentes. Una celda se ha dejado en blanco. Para definir los elementos, ya no será necesario especificar los valores desde-hasta. Basta con referenciar el área correspondiente con la propiedad **grid-area**.

```
.grid-item1 {
  background: blue;
  text-align: center;
  border: black 5px solid;
  grid-area: area1;
}

.grid-item2 {
  background: red;
  text-align: center;
  border: black 5px solid;
  grid-area: area2;
}
```

```
.grid-item3 {
  background: green;
  text-align: center;
  border: black 5px solid;
  grid-area: area3;
}
.grid-item4 {
  background: yellow;
  text-align: center;
  border: black 5px solid;
  grid-area: area4;
}
```