Universidad de Oviedo

Optativa práctica 2

Informe GRID

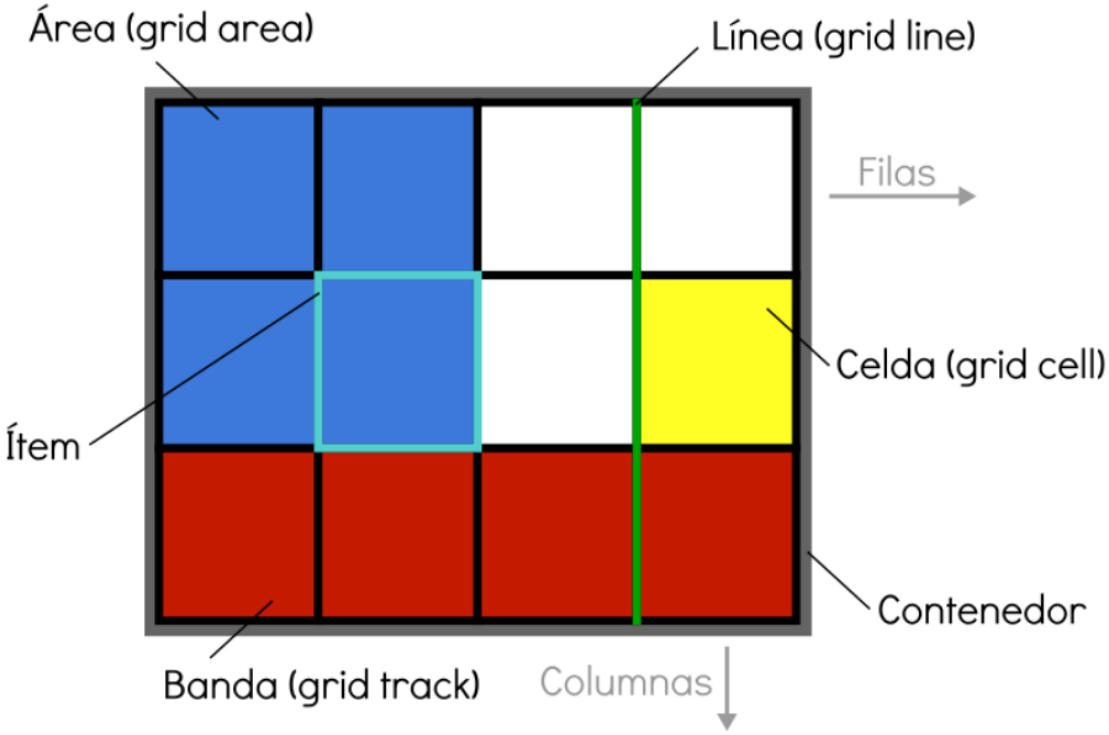
Marcos Matilla González

UO258935

CSS Grid

El proceso de distribuir y colocar los elementos a lo largo de una página suele ser costoso. Los distintos mecanismos como posicionamiento, floats o elementos en bloque suelen insuficientes para crear un layout. De esta necesidad surge **flexbox,** aunque está orientado a sistemas unidimensionales, es por eso por lo que **grid** nace, de esa necesidad que surge con los anteriores mecanismos, además recoge las ventajas de esos sistemas a la par de añadir numerosas mejoras y características. En resumen, es un sistema de maquetación basado en grillas cuyas características son la bidimensionalidad, la independencia de orden del markup y la flexibilidad.

Conceptos



- Contenedor: elemento padre que se encarga de definir la rejilla y contener todos aquellos elementos que formarán parte de nuestro grid.
- Ítem: cada uno de los elementos que formarán parte del contenedor de manera independiente.
- Celda (grid-cell): espacios reservados dentro del propio contenedor para los diferentes ítems.
- Area (grid-template-area): conjunto de celdas del contenedor, las cuales podremos repartir y separar de la manera en que queramos.
- Banda (grid-track): banda horizontal o vertical de celdas.
- Linea (grid-line): separador horizontal o vertical de celdas.

Creando un grid

Mediante las propiedades *grid-template-columns* y *grid-template-rows* es posible crear cuadrículas de un tamaño deseado.

Por ejemplo, supongamos que queremos diseñar una página web en la cual deseamos que el titulo ocupe toda la parte superior, en el centro a la izquierda nos muestre una barra de navegación para poder ir moviéndonos entre páginas, a la derecha nos muestre la información respectiva a la página y que, en la parte de abajo, se nos muestren una serie de vínculos a páginas relacionadas que nos aportarán más información a nuestra página.

1. Declaramos en la CSS el display del layout.

```
body {
    display: grid;
}
```

2. Indicamos como queremos las columnas dentro del body.

```
grid-template-columns: 1fr 3fr;
```

3. Definimos las áreas de la siguiente manera dentro del body.

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

De esta manera ya tendremos diseñado como será nuestra pantalla, ahora tan solo debemos indicarle al grid que parte de nuestro html ocupará cada una de esas áreas.

4. Declaramos las áreas.

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

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

Posición en el grid

Existen una serie de propiedades que nos permiten posicionar los elementos en nuestra cuadrícula, con estas podemos distribuir los elementos de una manera más sencilla. Estas son *justify-items* la cual distribuye los elementos en el eje horizontal y *align-items* la cual los distribuye en el eje vertical.

Ambas propiedades han de ser aplicadas sobre el contendor padre, aunque afecten a los hijos. En caso de que queramos que alguno de los hijos no herede del padre, deberemos sobrescribir su distribución.

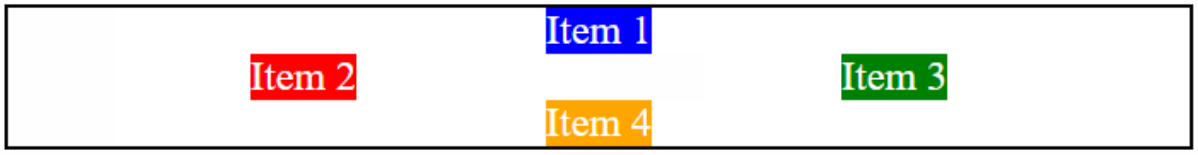
También podemos utilizar las propiedades *justify-content* o *align-content* para modificar la distribución de todo el contenido y no solo de los ítems por separado.

De esta manera controlamos todo el posicionamiento directamente desde el contenedor padre.

```
    justify-items:stretch;
    height:200px;
    align-items:stretch;

    justify-content:center;
    align-content:stretch;
}

.a { grid-area: head; background:blue }
.b { grid-area: menu; background:red }
.c { grid-area: main; background:green }
.d { grid-area: foot; background:orange }
```

```
    justify-items:stretch;
    height:200px;
    align-items:stretch;

    justify-content:stretch;
    align-content:stretch;
  }

.a { grid-area: head; background:blue }
.b { grid-area: menu; background:red }
.c { grid-area: main; background:green }
.d { grid-area: foot; background:orange }
```


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

    justify-content:stretch;
    align-content:stretch;
  }

.a { grid-area: head; background:blue }
.b { grid-area: menu; background:red }
.c { grid-area: main; background:green }
.d { grid-area: foot; background:orange }
```



Propiedades

Ajuste automático de celdas

grid-auto-columns: indica el tamaño automático de ancho que tendrán las columnas.

grid-auto-rows: indica el tamaño automático de alto que tendrán las filas.

grid-auto-flow: utiliza un algoritmo de auto colocación.

Ítems hijos

justify-self: altera la justificación del ítem hijo en el eje horizontal.

align-self: altera la alineación del ítem hijo en el eje vertical.

grid-area: indica un nombre al área especificada.

grid-column-start: Indica en que columna empezará el ítem.

grid-column-end: Indica en que columna acabará el ítem.

grid-row-start: Indica en que fila empezará el ítem.

grid-row-end: Indica en que fila acabará el ítem.

Grid con huecos

grid-column-gap: establece el tamaño de los huecos entre columnas.

grid-row-gap: establece el tamaño de los huecos entre filas.