MAQUETACION CSS CON FLEX Y GRID

Contenidos

¿QUÉ SON FLEX Y GRID?

Son dos nuevos valores posibles para la propiedad `display` añadidos a HTML5, que tienen una serie de características gracias a las cuales la maquetación web va a ser mucho más sencilla que con los métodos tradicionales vistos hasta el momento.

FLEX `display:flex`

Va a trabajar en una sola dirección (horizontal o vertical) y puede alinear, dar tamaño y distribuir los espacios restantes de todos los elementos que contenga.

GRID `display:grid`

Es el sistema de maquetación más potente hasta el momento, ya que permite trabajar de forma simultánea en ambas dimensiones, horizontal y vertical (filas y columnas).

MAQUETACIÓN CON FLEX

Cuando maquetamos con FLEX vamos a tener 2 componentes principales, el **contenedor FLEX** o etiqueta padre; y los **elementos flexibles** que están dentro de este. Las propiedades modificables de los elementos flexibles:

- Altura y anchura
- Orden
- Alineación (horizontal o vertical)
- Distribución a lo largo del padre

PROPIEDADES DEL CONTENEDOR FLEX

En cuanto indicamos en el **contenedor padre** la propiedad `display:flex`, automáticamente organiza los elementos hijo en forma de fila (por defecto), en el orden en que aparecen en el HTML.

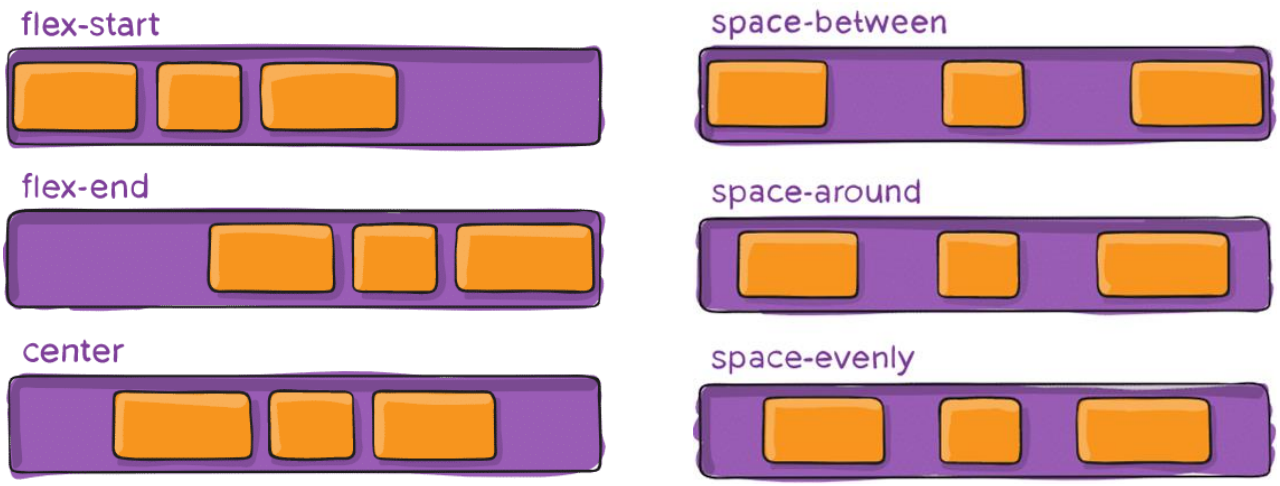
Si queremos modificar la **dirección** en la que se muestran esos elementos, podemos añadir también la propiedad `flex-direction`, con sus valores `row-reverse`, `column`, o `column-reverse`; según lo que busquemos.

Podemos controlar como se **ajustan** los elementos hijos al tamaño del padre, con la propiedad `flex-wrap`, que por defecto viene establecida como `nowrap` (no modifica la posición de los elementos). Pero también tiene los valores `wrap` (se van disponiendo en varias líneas comenzando por arriba) o `wrap-reverse` (se van disponiendo en varias líneas comenzando por abajo).

La dirección y el ajuste **pueden englobarse** en una única propiedad llamada `flex-flow`, por ejemplo: `flex-flow: row-reverse wrap;`

Para personalizar la **alineación horizontal** de los elementos a lo largo del contenedor, usamos la propiedad `justify-content`, con sus valores:

- `flex-start`: (por defecto) los elementos se alinean al principio.
- `flex-end`: los elementos se alinean al final.
- `center`: los elementos se centran en el contendor padre.
- `space-between`: el espacio libre se distribuye igualmente entre los elementos cogiendo todo el espacio del contenedor padre.
- `space-around`: el espacio libre se distribuye igualmente, pero se reserva una parte a repartir entre los extremos del contendor padre.
- `space-evenly`: el espacio libre se distribuye exactamente igual tanto entre cada elemento como la distancia a los extremos del contenedor padre.



Del mismo modo podemos controlar la **alineación vertical** de los elementos con la propiedad **`align-items`**, que tiene los siguientes valores:

- `flex-start`: (por defecto) los elementos se alinean arriba.
- `flex-end`: los elementos se alinean abajo.
- `center`: los elementos se centran en el contenedor padre.
- `stretch`: (no debemos indicar altura en los elementos) iguala la altura de cada elementos para que ocupen el alto del padre.
- `baseline`: (esa para elementos con texto) los elementos se centran para que los distintos textos que contienen queden alineados entre ellos.

Para personalizar la **alineación vertical** cuando tenemos **varias líneas** de elementos hijo, deberemos usar la propiedad **`align-content`**, cuyos valores son los mismos que la propiedad anterior, pero sustituyendo `baseline` por `space-between` y `space-around`.

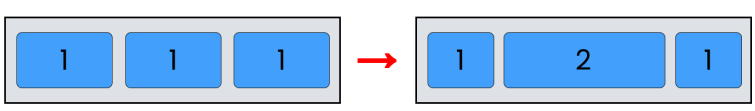
PROPIEDADES DE LOS ELEMENTOS FLEXIBLES

Orden

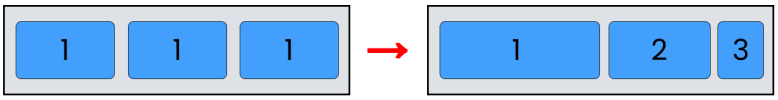
Usando la propiedad **`order`** (**0 por defecto**), podemos personalizar el orden en que se muestran los elementos hijo dentro de su contenedor. Por ejemplo: `order: 3;`

Tamaño

`flex-grow`: (0 por defecto) **Factor de crecimiento** al distribuir espacio libre. Si se modifica debe indicarse en todos los elementos. Por ejemplo: `flex-grow: 2;`



flex-shrink: (1 por defecto) **Factor de contracción** al distribuir la falta de espacio en el contenedor padre. Si se modifica debe indicarse en todos los elementos. Por ejemplo: `flex-shrink: 3;`



flex-basis: (auto por defecto) Es el **tamaño base** de un elemento antes de distribuir el espacio positivo o negativo. Por ejemplo: `flex-basis: 20%;`

Puedo englobar los tres bajo una única propiedad llamada **flex**, especificándolos en el orden que los hemos visto, por ejemplo: `flex: 1 1 20%;`

Alineación individual

Aunque en el contenedor hayamos aplicado una alineación específica para los hijos, podemos hacer que ciertos elementos tengan su propia alineación, con la propiedad **align-self**. Sus valores son los mismos que tenemos en la propiedad align-items. Por ejemplo, `align-self: flex-end;`

MAQUETACIÓN CON GRID

Su filosofía es muy similar a la de FLEX, tendremos un contenedor padre con la propiedad `display:grid`, desde el que controlaremos la disposición y tamaño de los elementos hijos. Solo qué en este caso, además vamos a poder generar estructuras complejas basadas en una rejilla (filas y columnas), para maquetarlos.

PROPIEDADES DEL CONTENEDOR GRID

Estructura de filas y columnas

Generalmente el primer paso es **definir la estructura** que va a tener nuestra **rejilla**, usaremos las siguientes propiedades, pudiendo indicar el ancho y alto de columnas y filas en **diferentes unidades**, como pixeles, porcentajes, auto o repartición por pesos:

- **grid-template-columns** : Número, tamaño y nombre de columnas. Por ejemplo:
  `grid-template-columns: 20% 50% 30%`
  `grid-template-columns: [id] 50px [producto] auto`
- **grid-template-rows:** Número, tamaño y nombre de filas.
  `grid-template-rows: 100px auto 100px auto`
  `grid-template-rows: [uno] 20px [dos] 40 px [tres] auto`
- **grid-row-gap:** seperación entre las filas.
- **grid-column-gap:** separación entre las columnas.
- **repeat**: sirve para replicar valores en columnas o filas. Por ejemplo:
  `grid-template-columns: repeat(3, [mi-columna] 20%) auto`
- **fr**: repartición del espacio por pesos. Por ejemplo:
  `grid-template-columns: 2fr 100px 1fr 2fr`

Alineación Horizontal de Elementos:

Usaremos la propiedad **`justify-items`**, con los valores:

- ↳ `start`: centra los elementos al inicio de cada celda.
- ↳ `end`: centra los elementos al final de cada celda.
- ↳ `center`: centra los elementos al centro de la celda.
- ↳ `strech`: (por defecto) ocupa todo el contenido de la celda.

Alineación Vertical de Elementos:

Usaremos la propiedad **`align-items`**, con los mismos valores que la propiedad anterior.

Podemos **juntar ambas propiedades** en una única, llamada **`place-item`**, indicando primero el valor vertical y luego el valor horizontal. Por ejemplo: `place-item: start center;`

Distribución horizontal del GRID dentro del contenedor:

Cuando queda espacio libre en el contenedor, podemos usar la propiedad **`justify-content`**, para indicar como se distribuyen las columnas a lo largo de este. Los valores posibles ya los hemos visto en propiedades de FLEX, y son:

- ✓ `start`
- ✓ `end`
- ✓ `center`
- ✓ `stretch`
- ✓ `space-between`
- ✓ `space-around`
- ✓ `space-evenly`

Distribución vertical del GRID dentro del contenedor:

Del mimo modo, usaremos la propiedad **`align-content`**, para indicar como se distribuyen las filas a lo largo del contendor. Con los mismos valores que la propiedad anterior.

De nuevo, es posible **unir ambas propiedades** en una sola, llamada **`place-content`**, indicando primero el valor vertical y luego el valor horizontal. Por ejemplo: `place-item: center space-around;`

Colocación implícita

Es lo que sucede cuando colocamos **elementos fuera de la rejilla** definida o cuando **no indicamos posición**. Para ello, tenemos las propiedades:

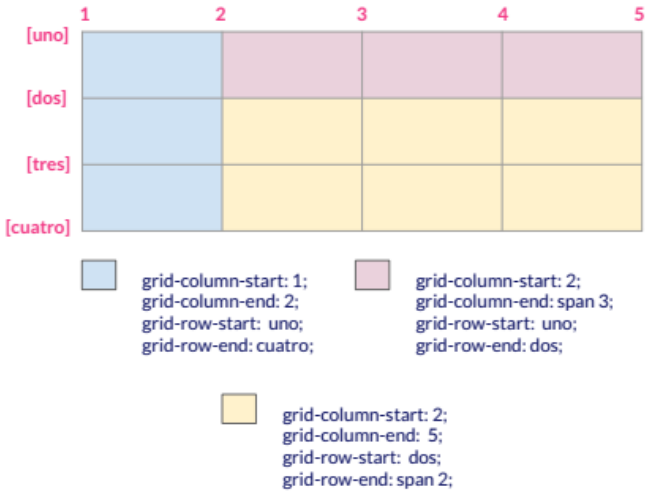
- ↳ **`grid-auto-columns`**: (0 por defecto) ancho de las columnas que sobresalen de mi rejilla
- ↳ **`grid-auto-rows`**: (0 por defecto) alto de las filas que sobresalen de mi rejilla
- ↳ **`grid-auto-flow`**: propiedad para ubicar los elementos que no se les define colocación, sus valores pueden ser:
  - `row`: (por defecto) rellena las filas primero.
  - `column`: rellena las columnas primero.
  - `dense`: intenta rellenar primero los huecos. Según el tamaño de los elementos, puede cambiar su orden, por lo que hay que usarlo con prudencia.

PROPIEDADES DE LOS ELEMENTOS DENTRO DE UN GRID

A los elementos dentro de un GRID, no va a afectarles las propiedades: `float, display:inline-block, display-table-cell, vertical-align` o `column-*`. Y podremos personalizar las siguientes características:

Área ocupada

- ✓ `grid-column-start`
- ✓ `grid-column-end`
- ✓ `grid-row-start`
- ✓ `grid-row-end`



Estas propiedades **pueden juntarse** de diferentes formas, que son:

- ✓ `grid-column: start / end`
- ✓ `grid-row: start / end`
- ✓ `grid: row-start column-start row-end column-end`

Alineación horizontal

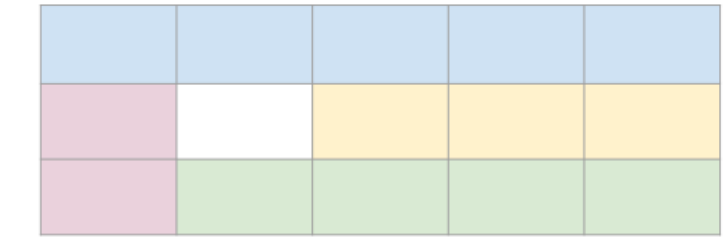
Usaremos la propiedad **`justify-itself`**, con los mismos valores que la propiedad justify-items.

Alineación vertical

Usaremos la propiedad **`align-itself`**, con los mismos valores que la propiedad align-items.

DEFINICIÓN DE AREAS EN GRID

Si no queremos especificar el tamaño para cada elemento del GRID podemos darles nombre con la propiedad `grid-area` y usar esos nombres para posicionarlos con la propiedad `grid-template-area`. Tal y como vemos en el siguiente ejemplo:



```
.container {
 grid-template-columns: repeat(5,20%);
 grid-template-rows: repeat(3,100px);
 grid-template-areas:
   "cab cab cab cab cab"
   "menu . main main main"
   "menu pie pie pie pie";
}
```

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

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

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

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

Hueco

Es posible juntar todas las propiedades de área en una sola, denominada **`grid-template`** e incluso podemos juntar todas las propiedades de nuestra rejilla en una única llamada **`grid`**.

Sin embargo, no suele ser muy recomendable, ya que quedan valores demasiado largos y difíciles de leer.