



# UT 4 – CSS Avanzado. Grid Layout.



# Contenido

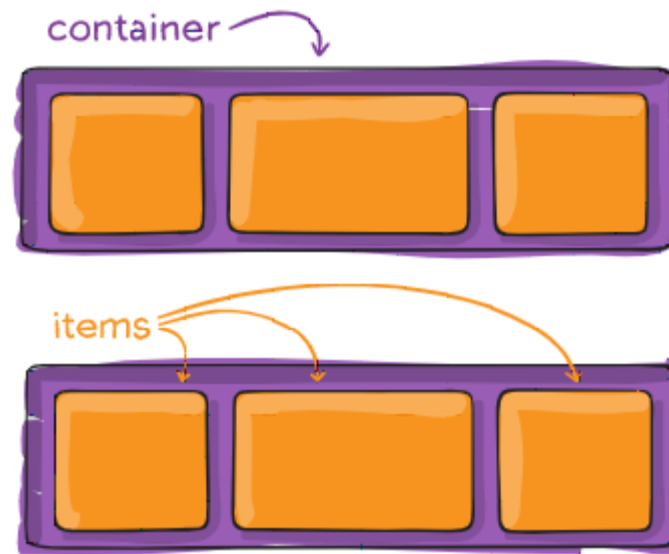
1. ¿Qué es Grid Layout?
2. Funcionamiento.
3. Propiedades contenedor.
4. Propiedades grid ítems.
5. Ejemplo práctico.

# 1. ¿Qué es Grid Layout?

- En los comienzos para distribuir los elementos se utilizaban tablas, luego float o inline-block, pero a estos métodos les faltaban funcionalidades importantes.
- Posteriormente apareció Flexbox, pero está pensado en una sola dimensión.
- **CSS Grid Layout** es el sistema de *layout* más potente disponible en CSS y el primero en ofrecer un sistema de dos dimensiones con funcionalidades completas (puede manejar filas y columnas).

## 2. Funcionamiento

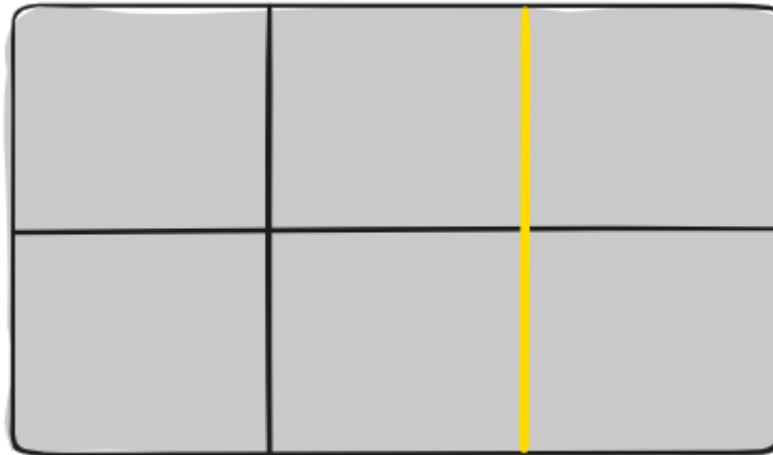
- **Idea principal:** Hay un contenedor de tipo grid (elemento padre) dentro del cual hay ítems (elementos hijos).



- Algunas propiedades es necesario aplicarlas al contenedor y otras a los ítems.

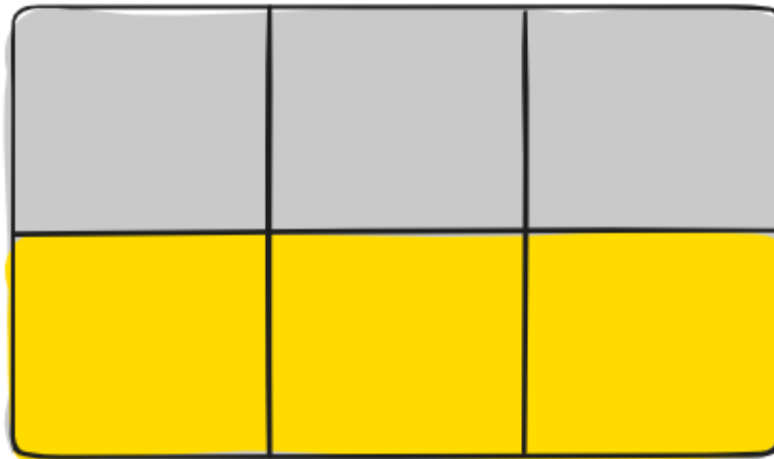
## 2. Funcionamiento

- Línea de la cuadrícula.



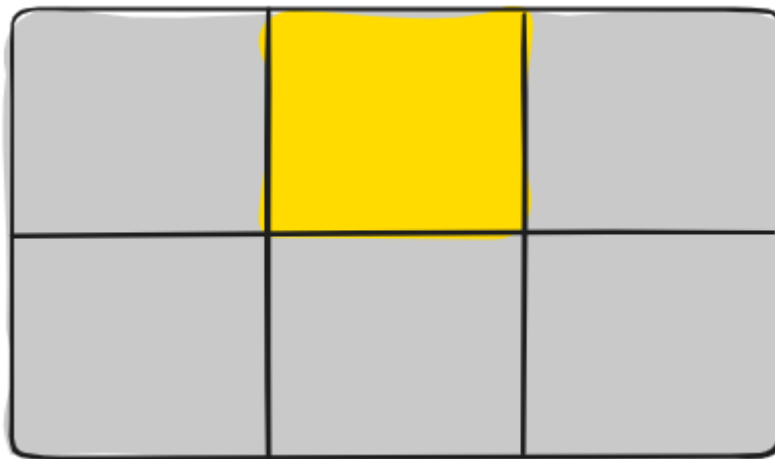
## 2. Funcionamiento

- Vía de la cuadrícula (*track*): Espacio entre dos líneas.



## 2. Funcionamiento

- **Celda de la cuadrícula (cell):** Espacio entre dos líneas de columna y dos líneas de fila adyacentes.



## 2. Funcionamiento

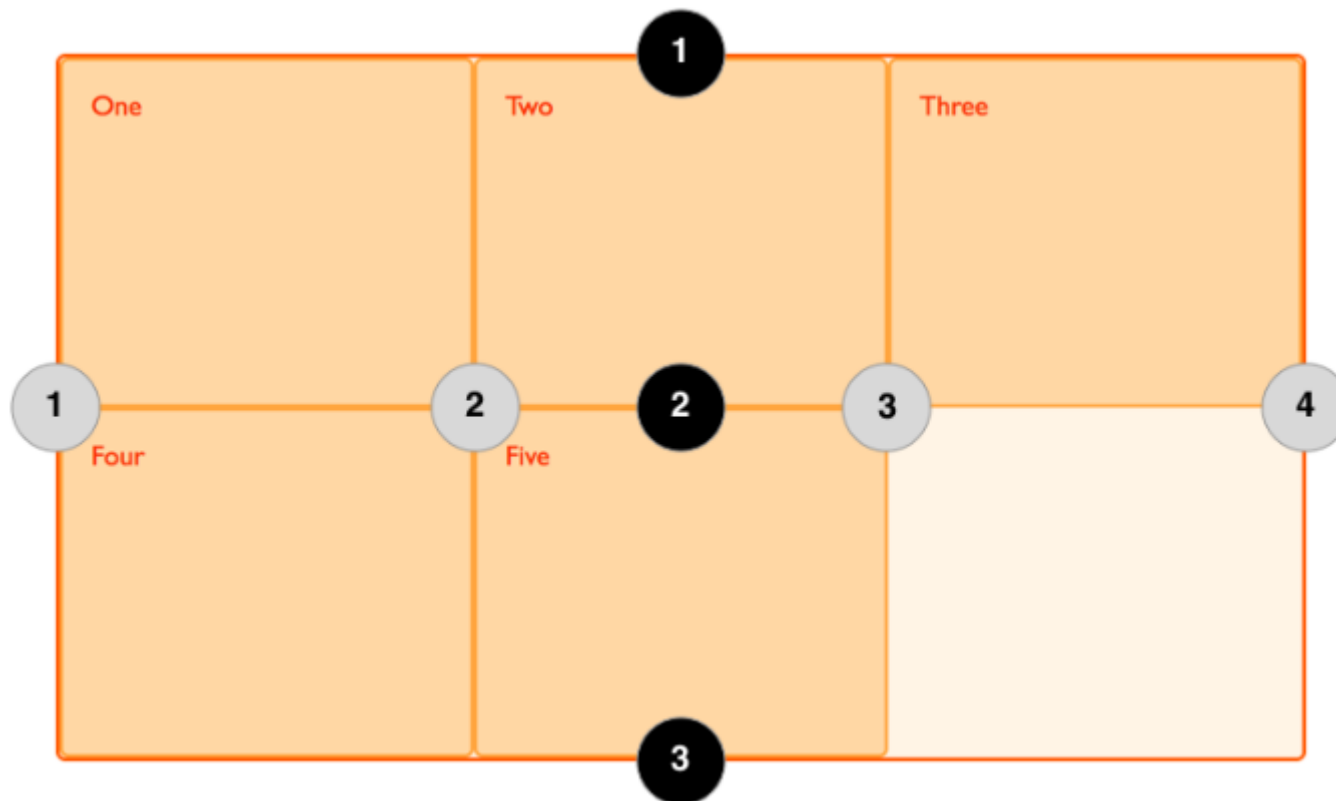
- **Área (*area*):** El espacio total rodeado por 4 líneas de la cuadrícula.





## 2. Funcionamiento

- Cuando se define la **cuadrícula** se definen las **vías**, no las líneas (luego Grid nos da las **líneas** numeradas a utilizar al posicionar elementos).



# 3. Propiedades contenedor

- Para definir un elemento como contenedor de cuadrícula (*grid container*) se utiliza la propiedad *display*.
- Se utilizará:
  - **display:grid;** Para que se comporte como un elemento de bloque.
  - **display:inline-grid;** Para que se comporte como un elemento en línea.

CSS

```
.container {  
  display: grid | inline-grid;  
}
```

# 3. Propiedades contenedor

- `grid-template-columns`
- `grid-template-rows`
- Permiten definir las filas y columnas con una lista de valores separados por espacios.
- Se especifica el tamaño de la vía (`track-size`) el cual puede ser dado en una longitud, un porcentaje o una fracción (utilizando la unidad **fr**).
- Al mismo tiempo, también se pueden definir nombres para las líneas.

CSS

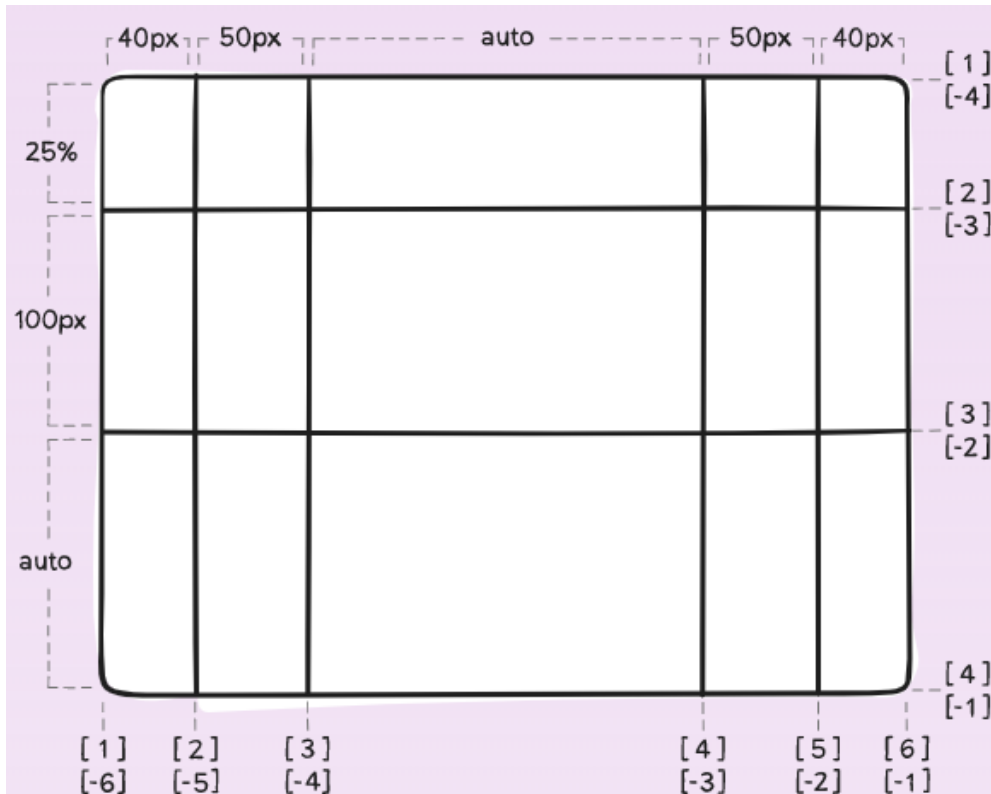
```
.container {  
  grid-template-columns: <track-size> ... | <line-name> <track-size> ...;  
  grid-template-rows: <track-size> ... | <line-name> <track-size> ...;  
}
```

# 3. Propiedades contenedor

## ■ Ejemplo:

CSS

```
.container {  
  grid-template-columns: 40px 50px auto 50px 40px;  
  grid-template-rows: 25% 100px auto;  
}
```

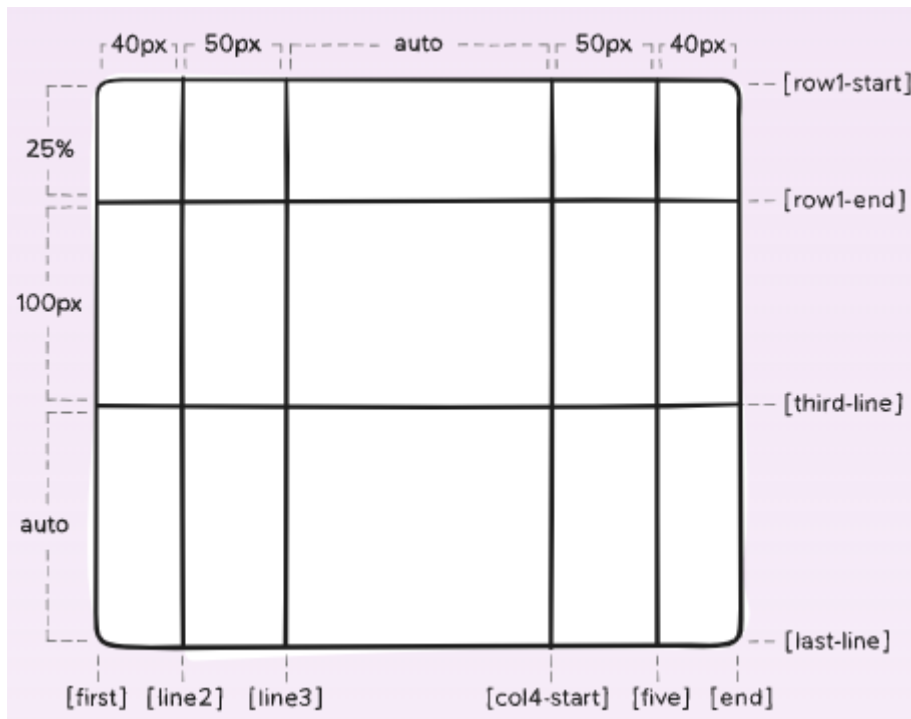


# 3. Propiedades contenedor

## ■ Ejemplo: (nombres)

css

```
.container {  
  grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];  
  grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];  
}
```



# 3. Propiedades contenedor

- **Nombres líneas:** Se pueden definir dos nombres diferentes para la misma línea, separándolos por espacios (ej. row1-end row2-start).

- **Ejemplo:**

(nombres)

CSS

```
.container {  
  grid-template-rows: [row1-start] 25% [row1-end row2-start] 25% [row2-end];  
}
```

# 3. Propiedades contenedor

- **Nombres líneas:** Se puede definir el mismo nombre para varias líneas. En ese caso, se referencian por el nombre y la cuenta. Ej. col-start 2.
- **Ejemplo:**  
(nombres)

```
CSS

.container {
  grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start];
}
```

```
CSS

.item {
  grid-column-start: col-start 2;
}
```

# 3. Propiedades contenedor

- **repeat():** Para repetir un número determinado de veces la misma especificación de vía.

- **Ejemplo:**

```
css
.container {
  grid-template-columns: repeat(3, 20px [col-start]);
}
```

Es equivalente a:

```
css
.container {
  grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start];
}
```

(3 veces lo mismo)



# 3. Propiedades contenedor

- **fr:** Es una unidad que permite especificar el tamaño de las vías como una fracción del espacio libre en el contenedor.
- **Ejemplo:**
  - Cada ítem ocupará 1/3 del ancho del contenedor.

```
CSS
.container {
  grid-template-columns: 1fr 1fr 1fr;
}
```

- El espacio libre se calcula después de cualquier elemento no flexible. Ej.: Se reparte el espacio disponible después de haber ocupado la segunda vía 50px.

```
CSS
.container {
  grid-template-columns: 1fr 50px 1fr 1fr;
}
```

# 3. Propiedades contenedor

- **grid-template-areas:** Define la plantilla referenciando los nombres de las grid áreas las cuales se han especificado con la propiedad grid-area (en cada uno de los ítems). Un punto significa una celda vacía.

```
CSS

.container {
  grid-template-areas:
    "<grid-area-name> | . | none | ..."
    "...";
}
```

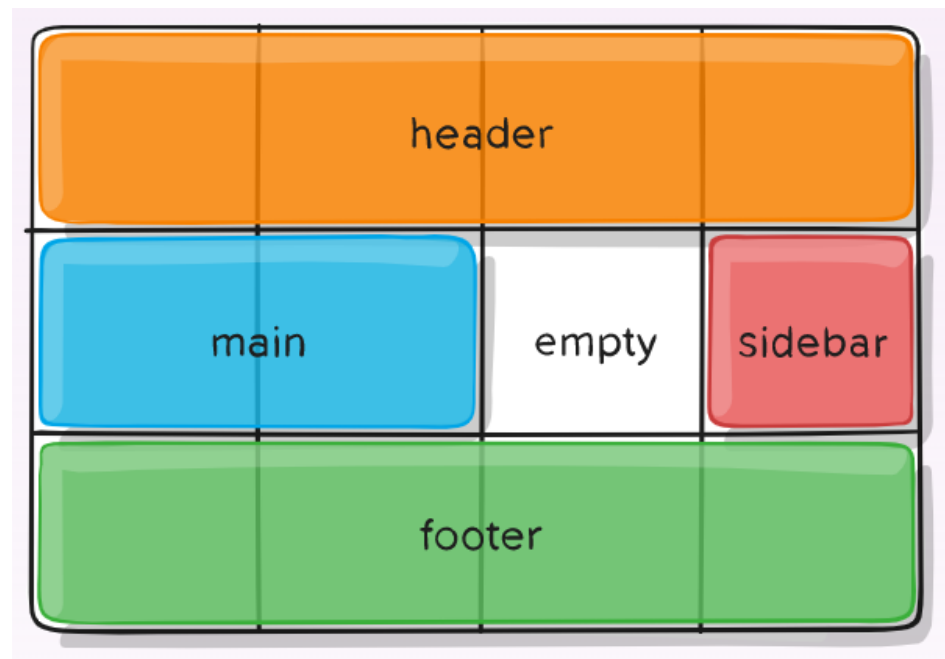
# 3. Propiedades contenedor

## ■ Ejemplo (grid-template-areas):

```
css

.item-a {
  grid-area: header;
}
.item-b {
  grid-area: main;
}
.item-c {
  grid-area: sidebar;
}
.item-d {
  grid-area: footer;
}

.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px;
  grid-template-rows: auto;
  grid-template-areas:
    "header header header header"
    "main main . sidebar"
    "footer footer footer footer";
}
```



# 3. Propiedades contenedor

- **grid-column-gap** y **grid-row-gap**: Especifica el tamaño de las líneas. Se puede pensar en ello como definir el ancho de los canales entre columnas/filas.

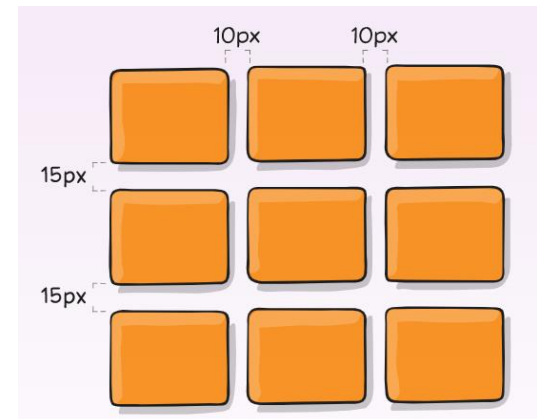
```
CSS

.container {
  grid-column-gap: <line-size>;
  grid-row-gap: <line-size>;
}
```

- Ejemplo:

```
CSS

.container {
  grid-column-gap: 10px;
  grid-row-gap: 15px;
}
```



# 3. Propiedades contenedor

- **justify-items:** Alinea el contenido de los ítems respecto a la dirección horizontal (filas).

```
CSS

.container {
  justify-items: start | end | center | stretch;
}
```

- Valores:
  - **start:** Al comienzo de la celda.
  - **end:** Al final de la celda.
  - **center:** Centro de la celda.
  - **stretch:** Ocupar todo el ancho de la celda.

# 3. Propiedades contenedor

## ■ Ejemplos (justify-items):

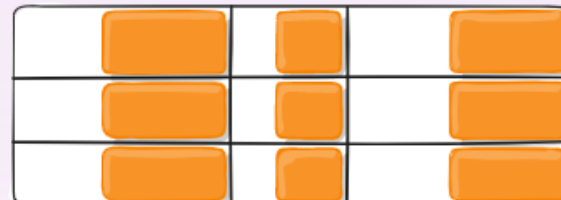
CSS

```
.container {  
  justify-items: start;  
}
```



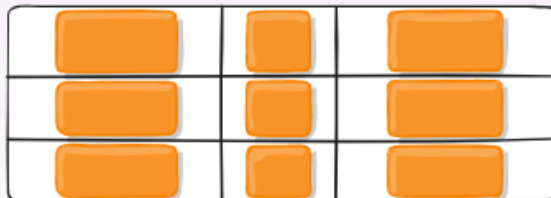
CSS

```
.container {  
  justify-items: end;  
}
```



CSS

```
.container {  
  justify-items: center;  
}
```



CSS

```
.container {  
  justify-items: stretch;  
}
```



# 3. Propiedades contenedor

- **align-items:** Alinea el contenido de los ítems respecto a la dirección vertical (columnas).

```
css
.container {
  align-items: start | end | center | stretch;
}
```

- Valores:
  - **start:** Al comienzo de la celda.
  - **end:** Al final de la celda.
  - **center:** Centro de la celda.
  - **stretch:** Ocupar toda la altura de la celda.

# 3. Propiedades contenedor

## ■ Ejemplos (align-items):

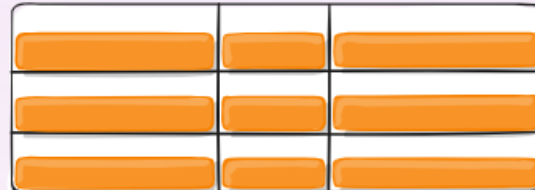
CSS

```
.container {  
  align-items: start;  
}
```



CSS

```
.container {  
  align-items: end;  
}
```



CSS

```
.container {  
  align-items: center;  
}
```



CSS

```
.container {  
  align-items: stretch;  
}
```





# 3. Propiedades contenedor

- **justify-content:** Alinea los ítems respecto al contenedor en la dirección horizontal (row)
  - En caso de que sobre espacio, por ejemplo si se ha definido el tamaño de los elementos en px y es menor que el total del contenedor.
  - Similar a flexbox.

CSS

```
.container {  
  justify-content: start | end | center | stretch | space-around | space-between | space-evenly;  
}
```

# 3. Propiedades contenedor

- justify-content (ejemplos)

CSS

```
.container {  
  justify-content: start;  
}
```

grid container



CSS

```
.container {  
  justify-content: end;  
}
```

grid container



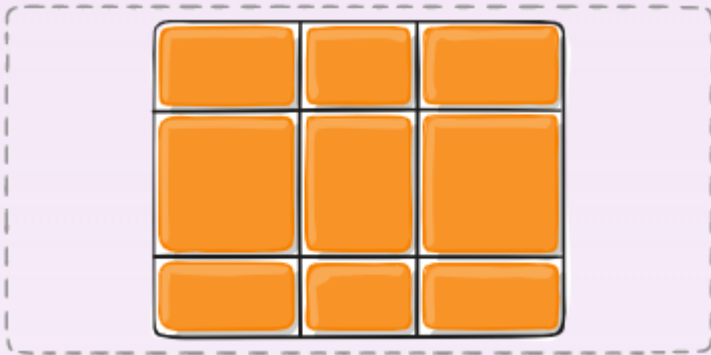
# 3. Propiedades contenedor

- justify-content (ejemplos)

CSS

```
.container {  
  justify-content: center;  
}
```

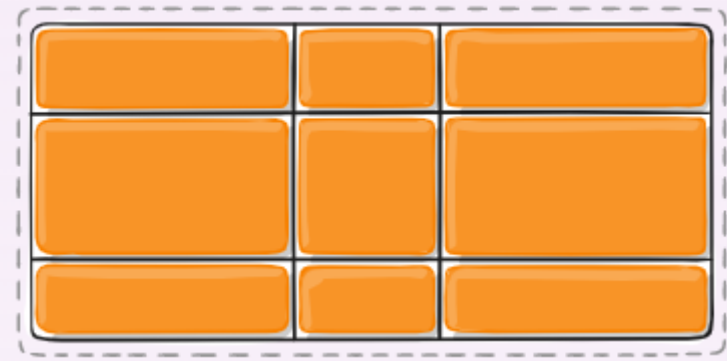
grid container



CSS

```
.container {  
  justify-content: stretch;  
}
```

grid container



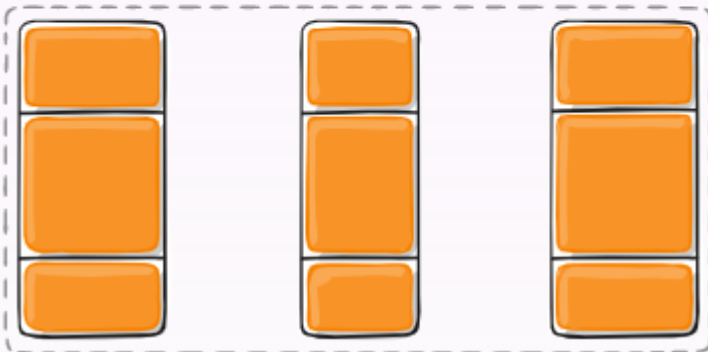
# 3. Propiedades contenedor

## ■ justify-content (ejemplos)

CSS

```
.container {  
  justify-content: space-between;  
}
```

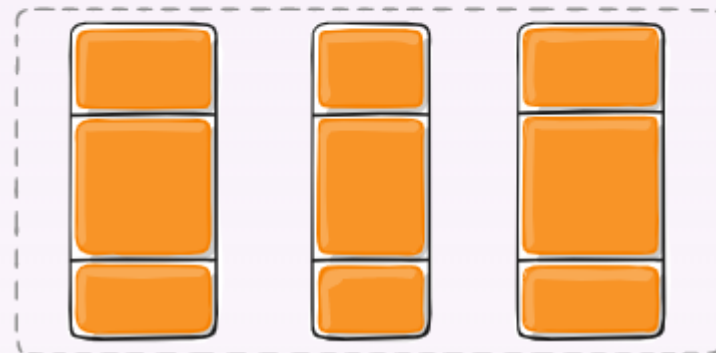
grid container



CSS

```
.container {  
  justify-content: space-around;  
}
```

grid container



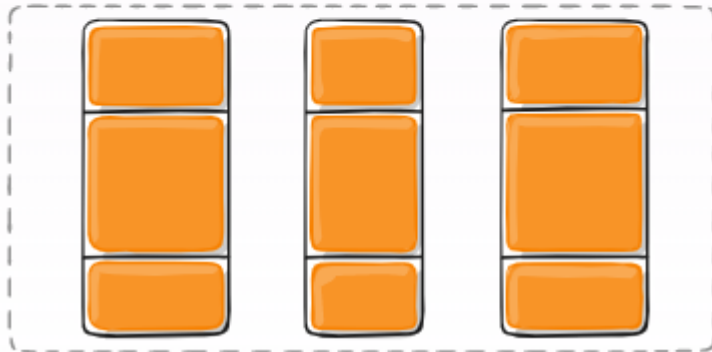
# 3. Propiedades contenedor

- justify-content (ejemplos)

CSS

```
.container {  
  justify-content: space-evenly;  
}
```

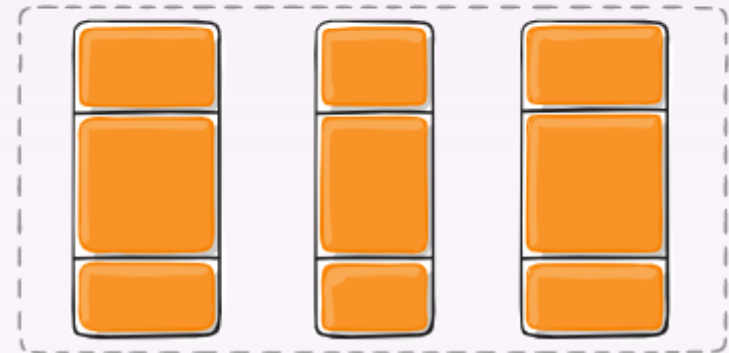
grid container



CSS

```
.container {  
  justify-content: space-around;  
}
```

grid container



# 3. Propiedades contenedor

- **align-content:** Alinea los ítems respecto al contenedor en la dirección vertical (column)
- En caso de que sobre espacio, por ejemplo si se ha definido el tamaño de los elementos en px y es menor que el total del contenedor.
- Similar a flexbox.

CSS

```
.container {  
  align-content: start | end | center | stretch | space-around | space-between | space-evenly;  
}
```

# 3. Propiedades contenedor

- align-content (ejemplos)

CSS

```
.container {  
  align-content: start;  
}
```

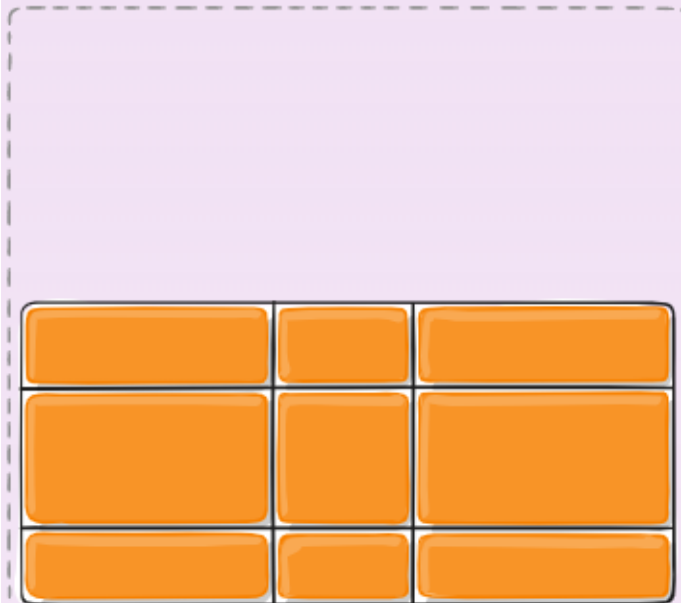
grid container



CSS

```
.container {  
  align-content: end;  
}
```

grid container



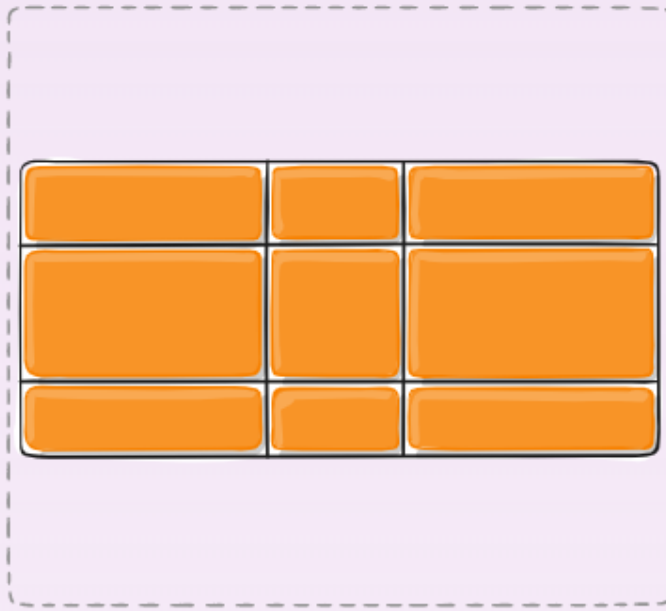
# 3. Propiedades contenedor

- align-content (ejemplos)

CSS

```
.container {  
  align-content: center;  
}
```

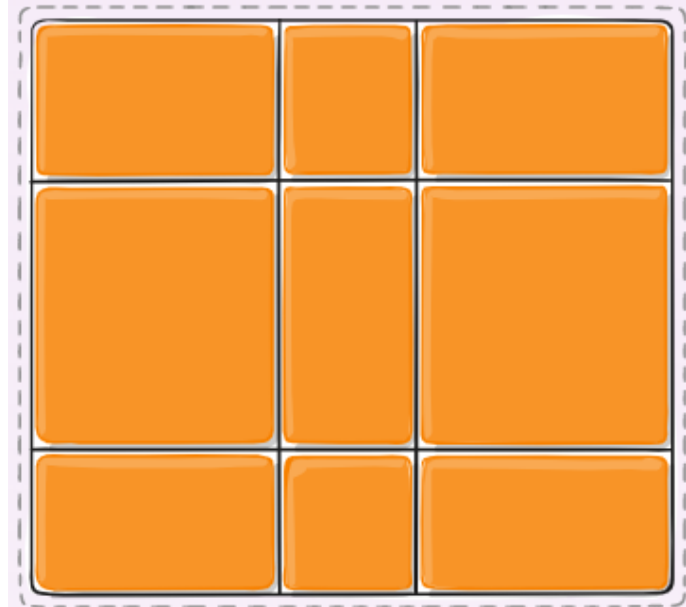
grid container



CSS

```
.container {  
  align-content: stretch;  
}
```

grid container





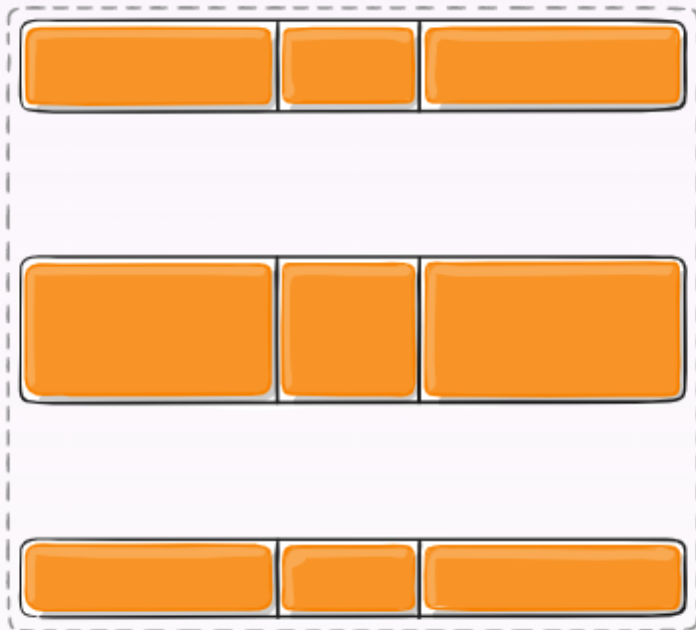
# 3. Propiedades contenedor

## ■ align-content (ejemplos)

CSS

```
.container {  
  align-content: space-between;  
}
```

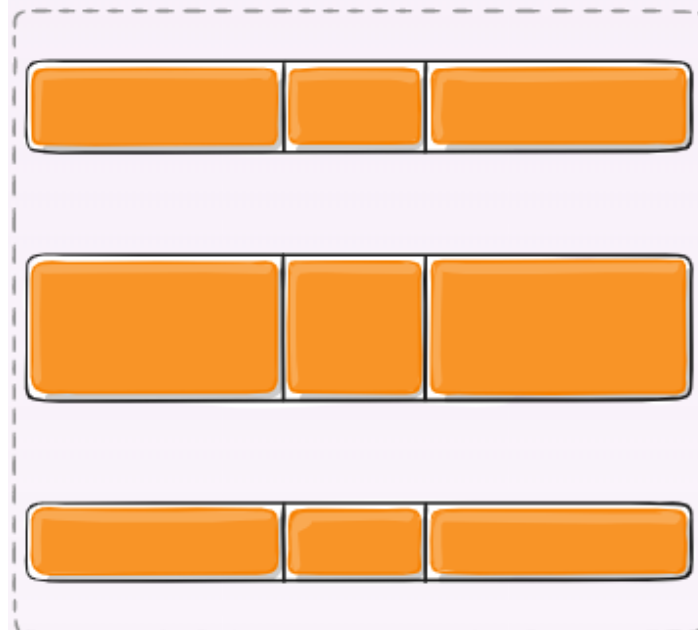
grid container



CSS

```
.container {  
  align-content: space-around;  
}
```

grid container



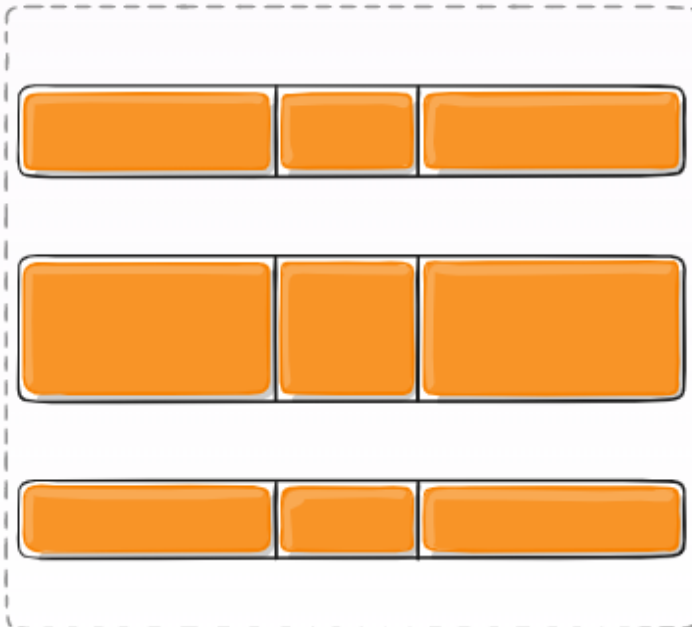
# 3. Propiedades contenedor

## ■ align-content (ejemplos)

CSS

```
.container {  
  align-content: space-evenly;  
}
```

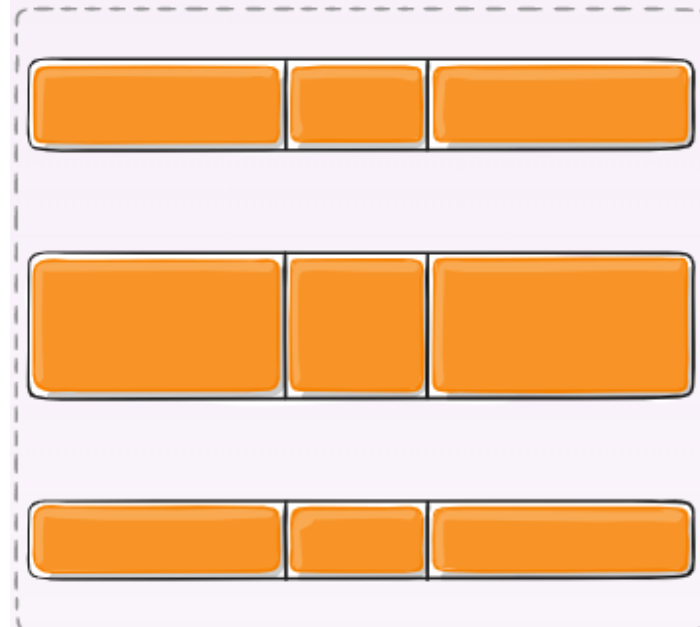
grid container



CSS

```
.container {  
  align-content: space-around;  
}
```

grid container



# 3. Propiedades contenedor

- **grid-auto-columns y grid-auto-rows:** Especifica la plantilla para las vías (columnas o filas) generadas automáticamente (implícitas) cuando hay más ítems que celdas definidas o cuando un ítem se coloca fuera de la rejilla definida (explícita).

CSS

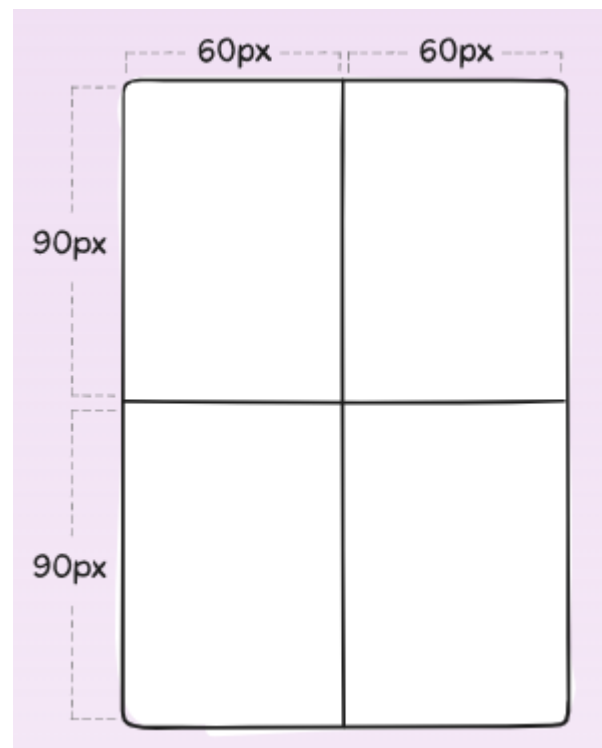
```
.container {  
  grid-auto-columns: <track-size> ...;  
  grid-auto-rows: <track-size> ...;  
}
```

# 3. Propiedades contenedor

- Ejemplo (grid-auto-columns).
- Se define la plantilla explícita, 2x2:

CSS

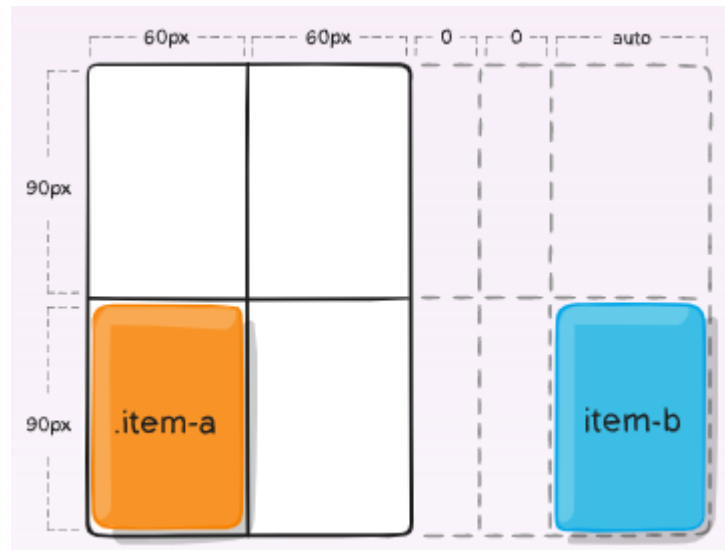
```
.container {  
  grid-template-columns: 60px 60px;  
  grid-template-rows: 90px 90px;  
}
```



# 3. Propiedades contenedor

- Ejemplo (grid-auto-columns).
- Y se ubican los ítems del siguiente modo:

```
css
.item-a {
  grid-column: 1 / 2;
  grid-row: 2 / 3;
}
.item-b {
  grid-column: 5 / 6;
  grid-row: 2 / 3;
}
```

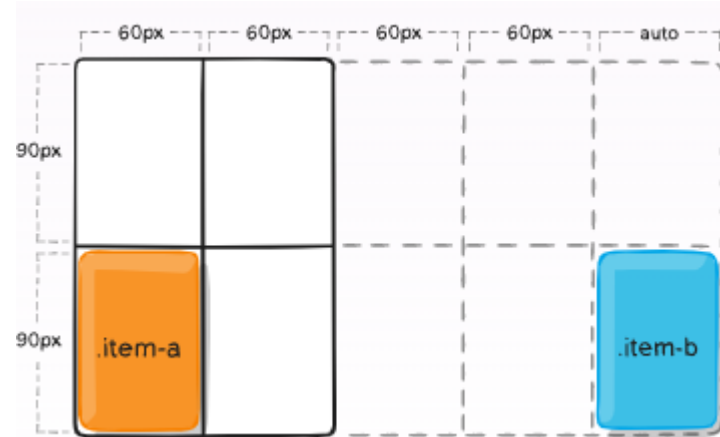


- El ítem b empieza en la línea 5 y finaliza en la línea 6 de las columnas. No existe definida rejilla para esos valores, pero se crean automáticamente (implícitamente) con ancho 0 y auto (para la que contiene el ítem).

# 3. Propiedades contenedor

- Ejemplo (grid-auto-columns).
- Pero se puede definir la anchura de las celdas creadas implícitamente:

```
CSS  
  
.container {  
  grid-auto-columns: 60px;  
}
```



- El ítem b empieza en la línea 5 y finaliza en la línea 6 de las columnas. No existe definida rejilla para esos valores, pero se crean automáticamente (implícitamente) con ancho 60px y auto (para la que contiene el ítem).

# 3. Propiedades contenedor

- **grid-auto-flow**: Si no se especifica una posición para los ítems en la rejilla, estos se colocan siguiendo un algoritmo de *auto-placement*. Con esta propiedad se puede especificar dicho algoritmo.
- Valores:
  - **row**: Se van rellenando las celdas primero de izquierda a derecha (rellenando las filas).
  - **column**: Se van rellenando las celdas primero de arriba abajo (rellenando las columnas).
  - **dense**: Se van rellenando las celdas teniendo en cuenta el espacio libre y el tamaño del elemento a posicionar. No es muy accesible, ya que los elementos pueden quedar desordenados.

<https://www.joomlashack.com/blog/tutorials/grid-auto-flow-dense-property/>

# 3. Propiedades contenedor

## ■ Ejemplo. grid-auto-flow: row

html

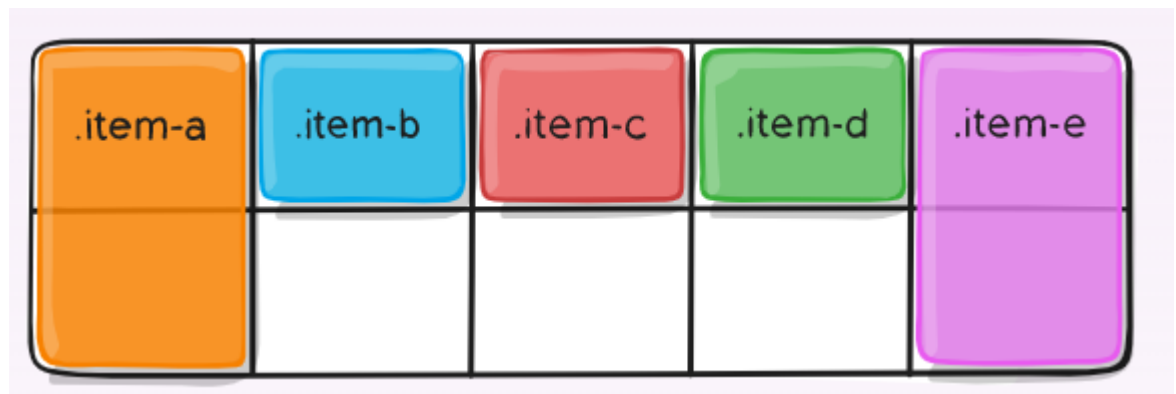
```
<section class="container">
  <div class="item-a">item-a</div>
  <div class="item-b">item-b</div>
  <div class="item-c">item-c</div>
  <div class="item-d">item-d</div>
  <div class="item-e">item-e</div>
</section>
```

css

```
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px 60px 60px;
  grid-template-rows: 30px 30px;
  grid-auto-flow: row;
}
```

css

```
.item-a {
  grid-column: 1;
  grid-row: 1 / 3;
}
.item-e {
  grid-column: 5;
  grid-row: 1 / 3;
}
```





# 3. Propiedades contenedor

## ■ Ejemplo. grid-auto-flow: column

html

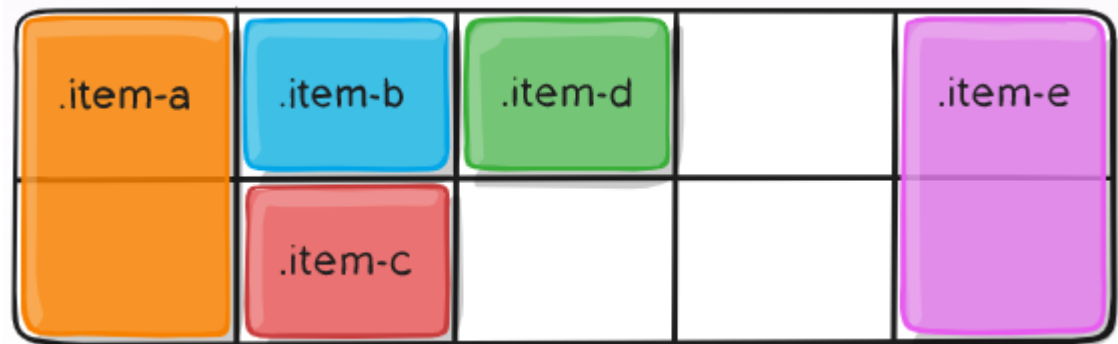
```
<section class="container">
  <div class="item-a">item-a</div>
  <div class="item-b">item-b</div>
  <div class="item-c">item-c</div>
  <div class="item-d">item-d</div>
  <div class="item-e">item-e</div>
</section>
```

css

```
.container {
  display: grid;
  grid-template-columns: 60px 60px 60px 60px 60px;
  grid-template-rows: 30px 30px;
  grid-auto-flow: column;
}
```

css

```
.item-a {
  grid-column: 1;
  grid-row: 1 / 3;
}
.item-e {
  grid-column: 5;
  grid-row: 1 / 3;
}
```



# 3. Propiedades contenedor

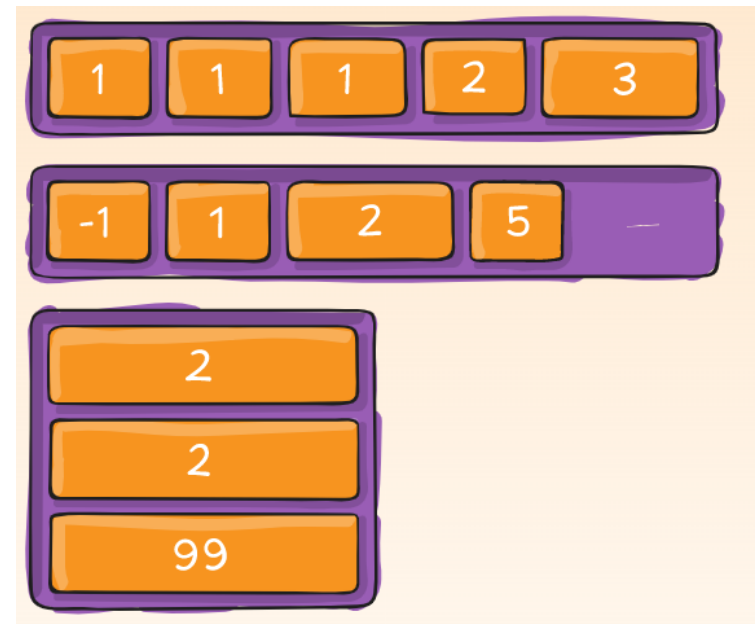
- Propiedades shorthand:
- **grid-template**: Para grid-template-rows, grid-template-columns y grid-template-areas.
- **grid-gap**: Para grid-row-gap y grid-column-gap.
- **place-items**: Para align-items y justify-items.
- **place-content**: Para align-content y justify-content.
- **grid**: Para grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns y grid-auto-flow.

## 4. Propiedades ítems

- Por defecto los ítems se muestran en el orden que aparecen en el código, pero se puede modificar con la propiedad **order**.
- El valor para la propiedad order es un número entero, que indica el orden.
- Funciona igual que en flexbox.

CSS

```
.item {  
  order: <integer>; /* default is 0 */  
}
```



## 4. Propiedades ítems

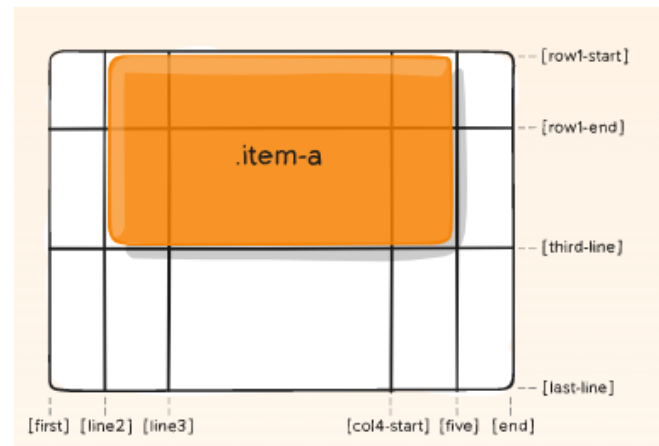
- **grid-column-start, grid-column-end, grid-row-start, grid-row-end.**  
Determina una localización para un ítem en la rejilla refiriéndose a sus líneas.
- Valores:
  - **línea:** Número o nombre de la línea.
  - **span** número: El ítem ocupará ese determinado número de vías (*tracks*).
  - **span** nombre: El ítem ocupará hasta la siguiente línea con ese nombre.
  - **auto:** span automático o por defecto 1 celda.
- Se puede utilizar la propiedad **z-index** en el caso de que se superpongan varios elementos.

# 4. Propiedades ítems

- `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end`.
- Ejemplos:

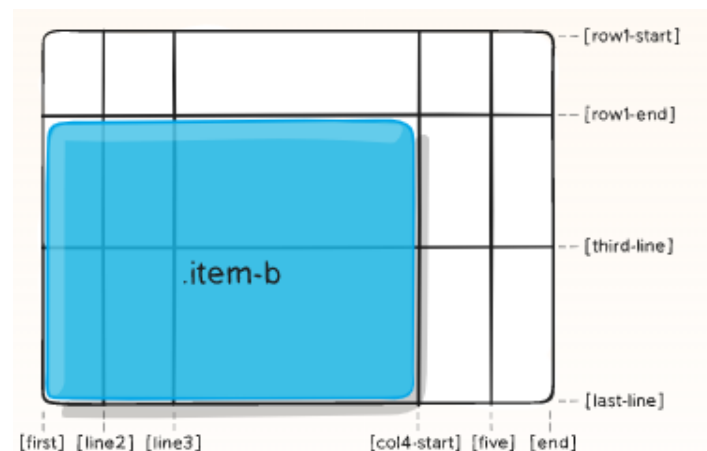
CSS

```
.item-a {  
  grid-column-start: 2;  
  grid-column-end: five;  
  grid-row-start: row1-start;  
  grid-row-end: 3;  
}
```



CSS

```
.item-b {  
  grid-column-start: 1;  
  grid-column-end: span col4-start;  
  grid-row-start: 2;  
  grid-row-end: span 2;  
}
```



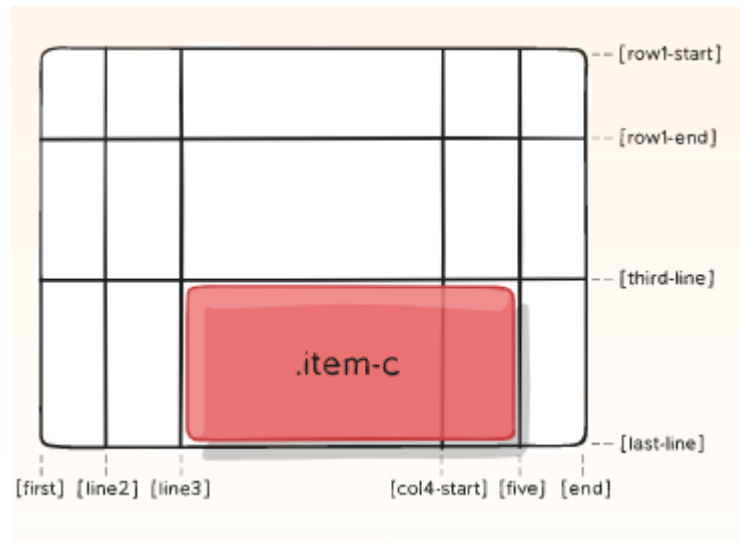
# 4. Propiedades ítems

- `grid-column`, `grid-row`. Propiedades shorthand.
- Ejemplo:

`<start-line> / <end-line>`

```
css

.item-c {
  grid-column: 3 / span 2;
  grid-row: third-line / 4;
}
```

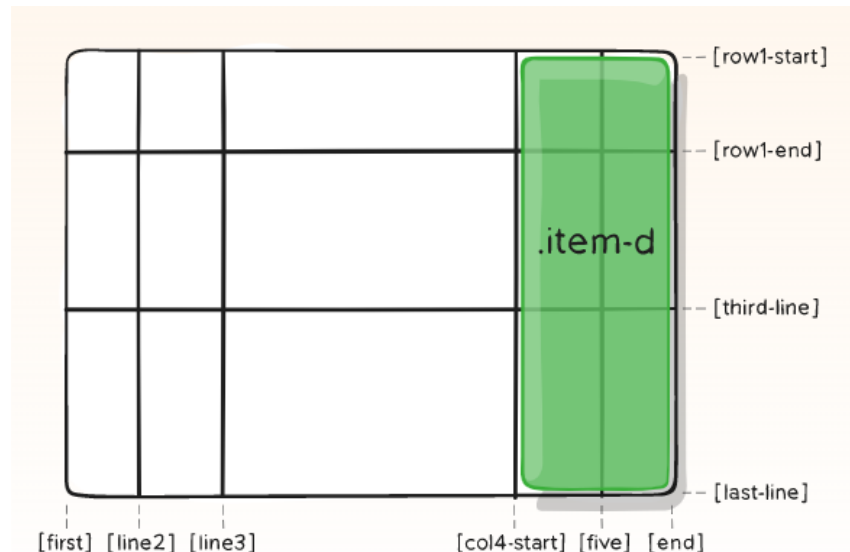


## 4. Propiedades ítems

- **grid-area**. Además de utilizarse como se explicó junto con **grid-template-areas** puede ser también una shorthand para **grid-column-start**, **grid-column-end**, **grid-row-start**, **grid-row-end**.
- Ejemplo:

```
css

.item-d {
  grid-area: 1 / col4-start / last-line / 6;
}
```

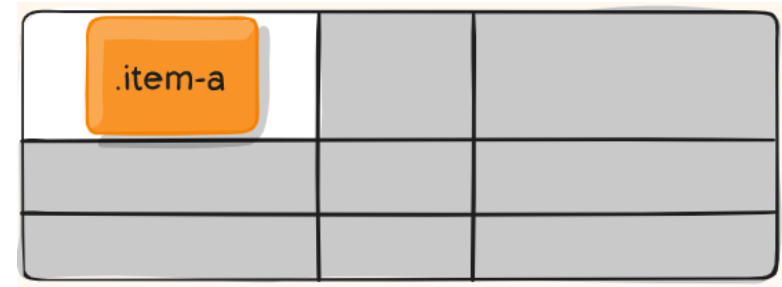


## 4. Propiedades ítems

- **justify-self.** Se puede utilizar para alinear un ítem en el eje horizontal (similar a justify-items).

- Ejemplo:

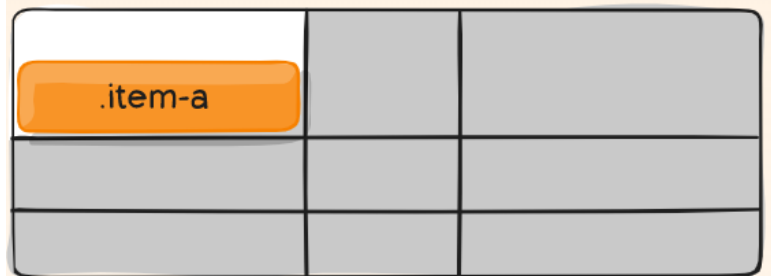
```
css
.item-a {
  justify-self: center;
}
```



- **align-self.** Se puede utilizar para alinear un ítem en el eje vertical (similar a align-items).

- Ejemplo:

```
css
.item-a {
  align-self: end;
}
```



- **place-self:** Propiedad shorthand para justify-self y align-self.



# 5. Ejemplo práctico

- <https://cssgridgarden.com/#es>

## GRID GARDEN


◀ Nivel 1 de 28 ▶

¡Bienvenido a Grid Garden, donde escribirás tu código CSS para cultivar tu jardín de zanahorias! Riega solo las áreas que tienen zanahorias usando la propiedad `grid-column-start`.

Por ejemplo, `grid-column-start: 3;` regará el área comenzando por la tercera línea vertical, que es otra manera de decir el 3er borde vertical contando desde la izquierda de la cuadrícula.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20%
4   20% 20%;
5   grid-template-rows: 20% 20% 20% 20%
6   20%;
7 }
8
9 #water {
10   
11 }
12
13
14
```

Siguiente



# Referencias

- CSS-TRICKS. *A complete Guide to Grid.*

<https://css-tricks.com/snippets/css/complete-guide-grid/>

- Rachel Andrew Grid by Example.

<https://gridbyexample.com/>

- MDN, Conceptos Básicos de Grid Layout.

[https://developer.mozilla.org/es/docs/Web/CSS/CSS\\_Grid\\_Layout/Conceptos\\_B%C3%A1sicos\\_del\\_Posicionamiento\\_con\\_Rejillas](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Grid_Layout/Conceptos_B%C3%A1sicos_del_Posicionamiento_con_Rejillas)

- Casos de uso típicos de Grid Layout.

[https://developer.mozilla.org/es/docs/Web/CSS/CSS\\_Grid\\_Layout/Realizing\\_common\\_layouts\\_using\\_CSS\\_Grid\\_Layout](https://developer.mozilla.org/es/docs/Web/CSS/CSS_Grid_Layout/Realizing_common_layouts_using_CSS_Grid_Layout)

- W3C. CSS Grid Layout Module.

<https://www.w3.org/TR/css-grid-1/>