UT3: IMPLANTACIÓN DE CONTENIDO MULTIMEDIA

FLEXBOX Y CSS GRID

DIW- FRANCISCO CORBERA NAVAS

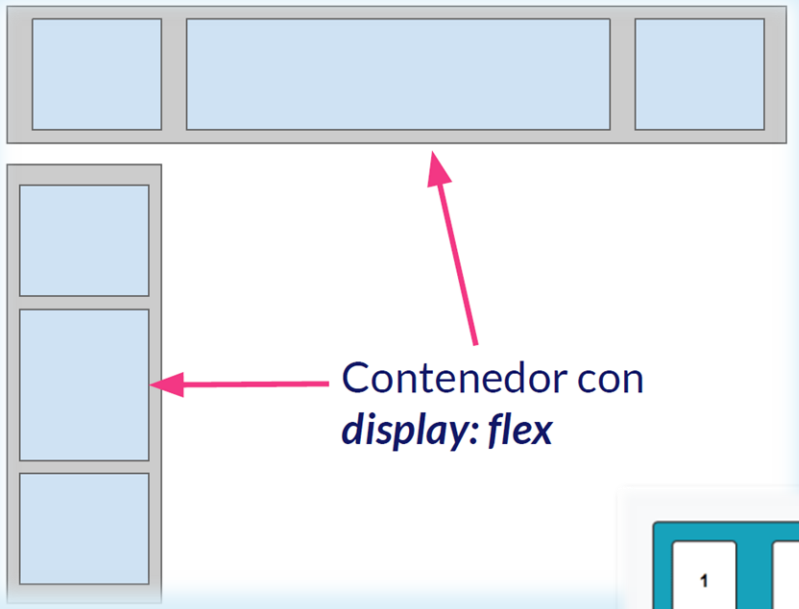
¿QUÉ SON?

FLEX y GRID son dos nuevos valores (HTML 5) que vamos a poder dar a la propiedad CSS **display** y que nos van a permitir, junto con otras propiedades, maquetar nuestras páginas web de una manera más fácil a la que se usaba tradicionalmente.

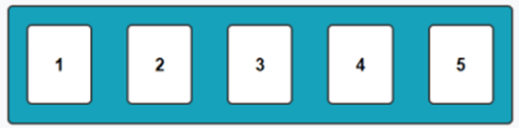
INTRODUCCIÓN

- **FLEX**
  - Elemento contenedor con **display: flex**
  - Modelo unidimensional, diseño en una sola dimensión, ya sea como fila o como columna.
  - Podremos:
    - Alinear los elementos que contiene.
    - Dar tamaño a los elementos que contiene.
    - Distribuir el espacio sobrante entre los elementos.

INTRODUCCIÓN



display: flex;

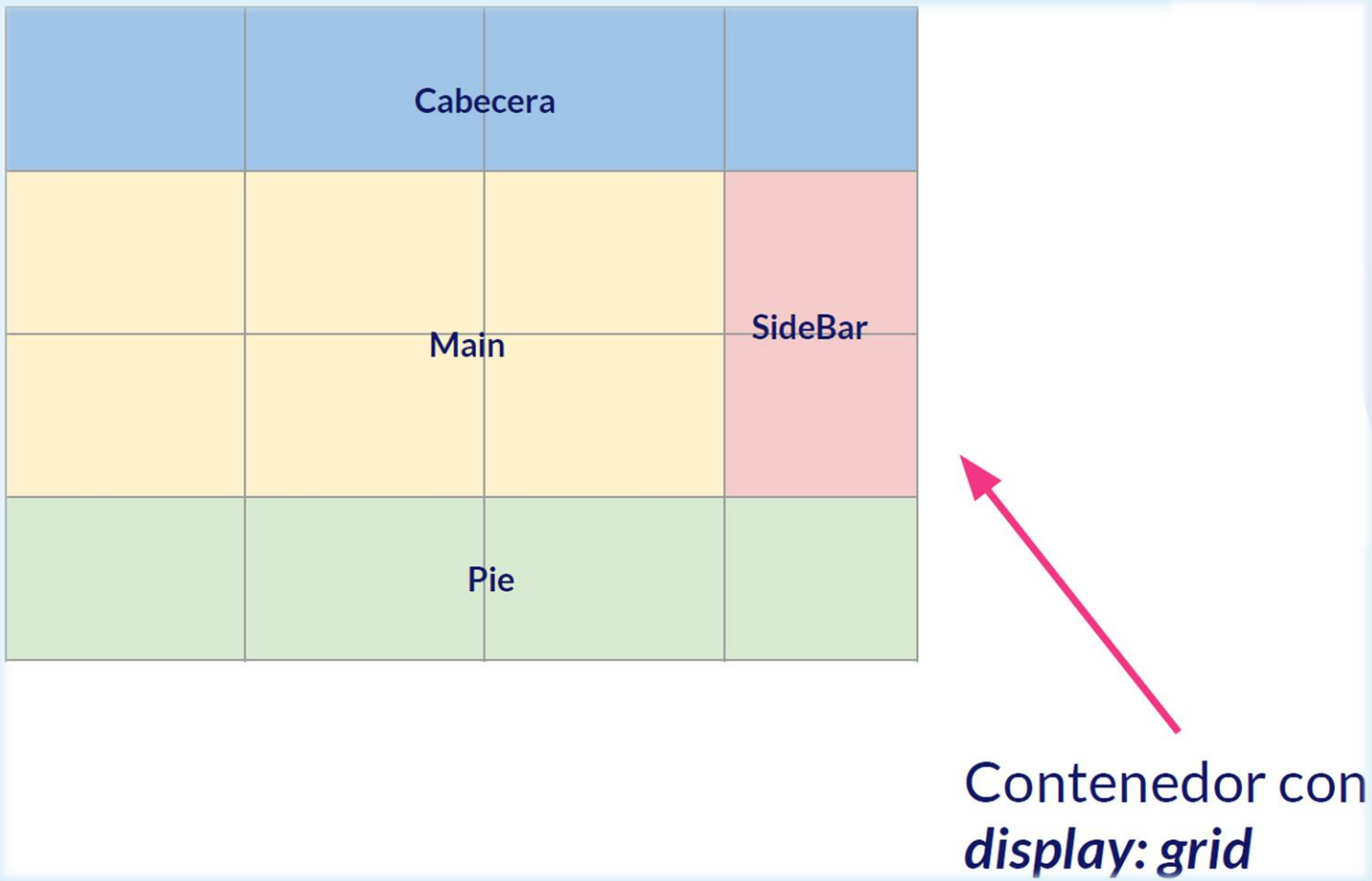


display: inline-flex;

INTRODUCCIÓN

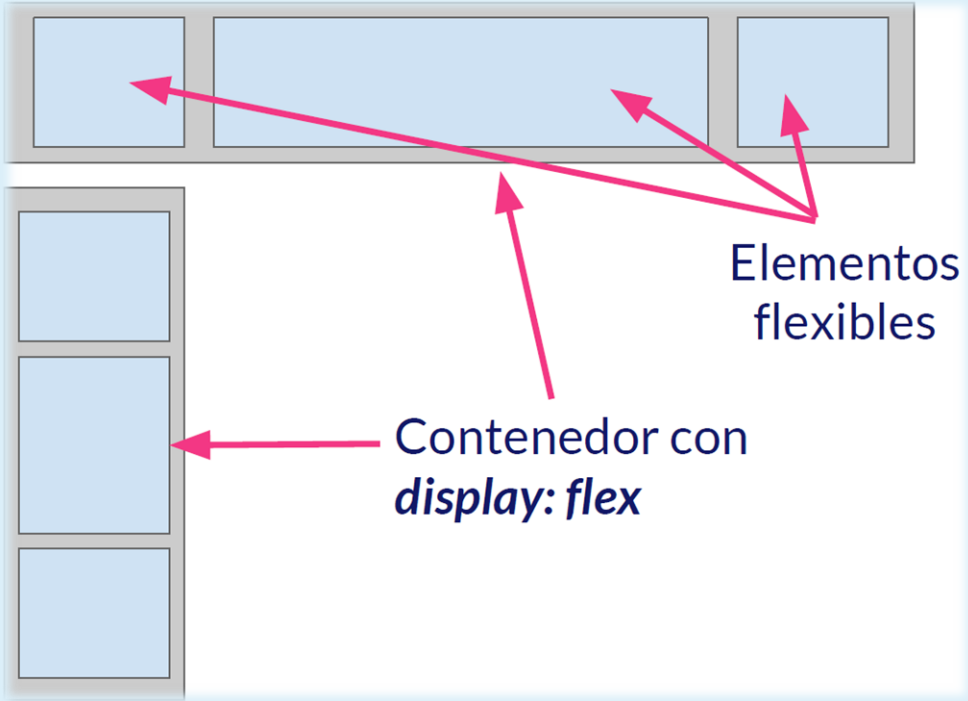
- **GRID**
  - Elemento contenedor con **display: grid**
  - Modelo bidimensional, permite trabajar con filas y columnas al mismo tiempo.
  - El sistema de maquetación más potente.
  - Muchos frameworks y librerías utilizan un sistema grid, donde definen una cuadrícula determinada. Con solo ir modificando los nombres de las clases de los elementos HTML, podemos darle tamaño, posicionarlo o colocarlo.

INTRODUCCIÓN



FLEX

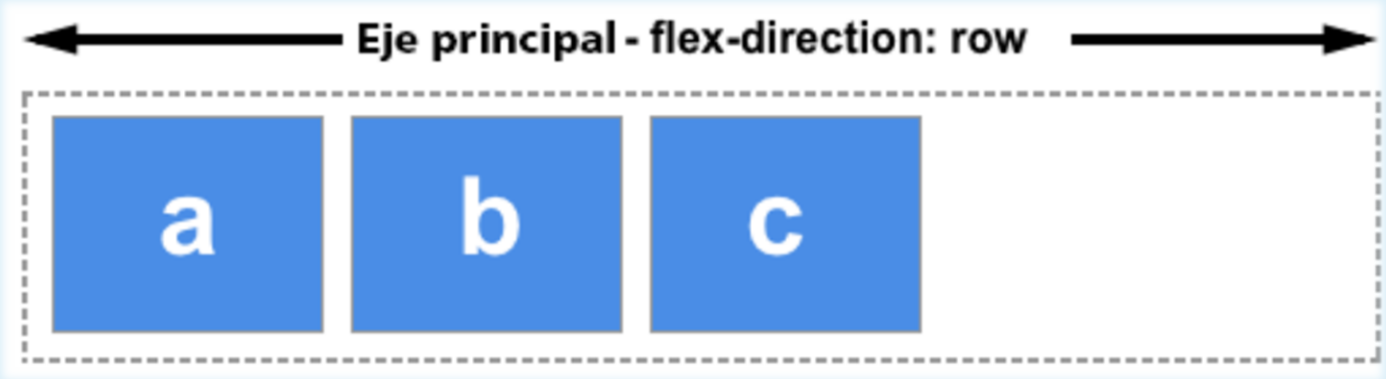
- La idea principal de la maquetación con elementos FLEX es que vamos a tener un elemento, **contenedor** (display: flex) , que va poder controlar propiedades de los elementos que contiene, **ítems**.

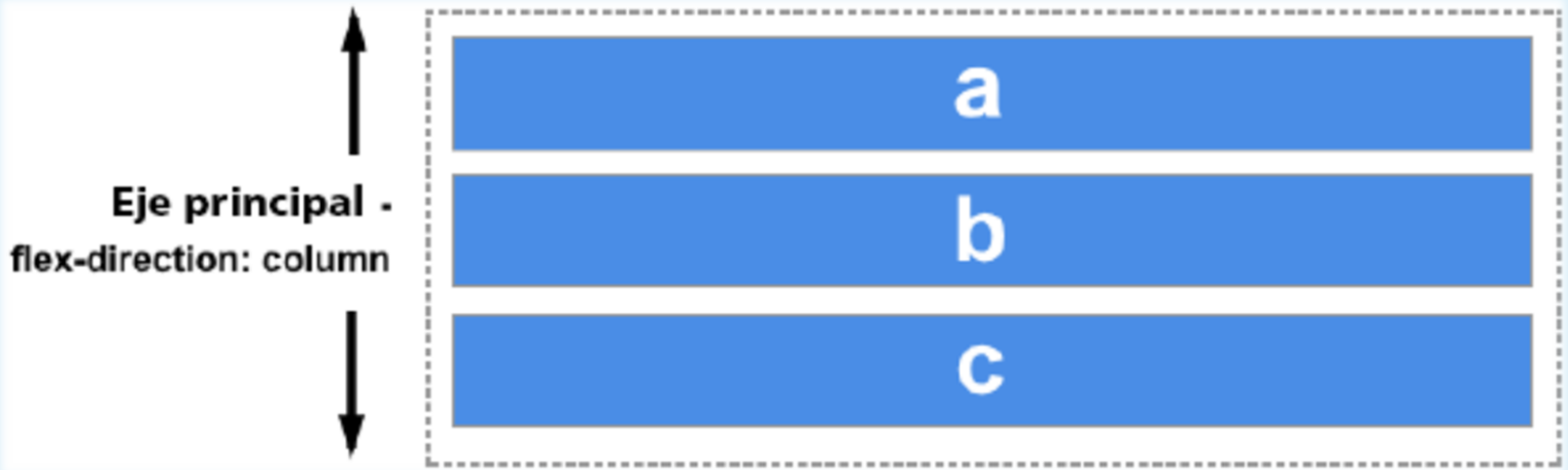


FLEX: DIRECCIÓN

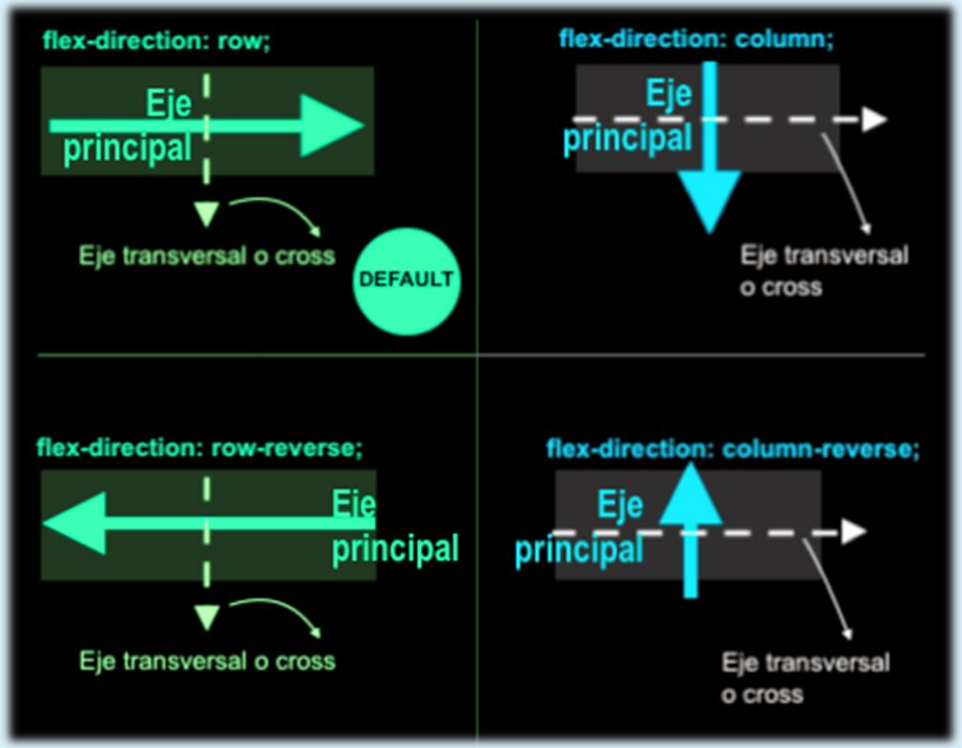
- Los contenedores tienen una orientación principal específica, que por defecto es la horizontal (fila).
- Esta dirección está definida en la propiedad **flex-direction**, que posee cuatro posibles valores:
  - row (por defecto)
  - row-reverse
  - column
  - column-reverse

FLEX: DIRECCIÓN





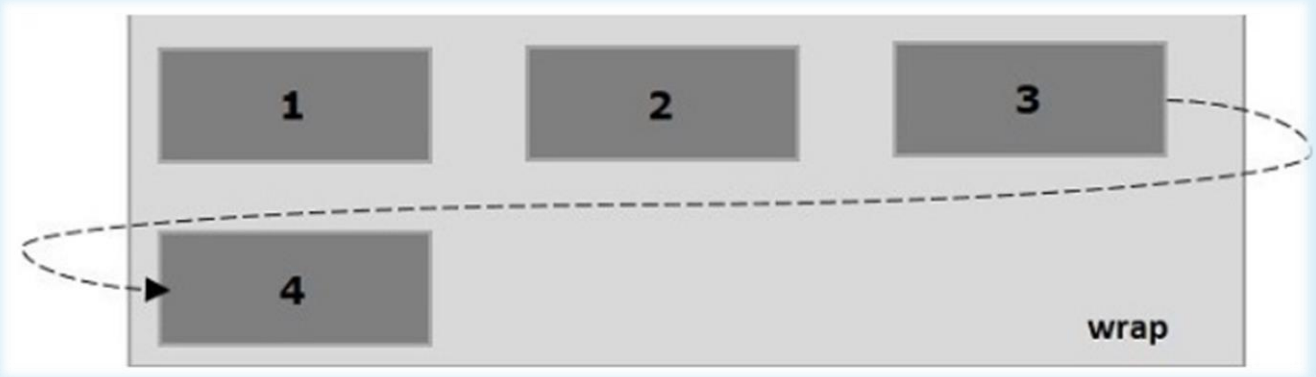
FLEX: DIRECCIÓN

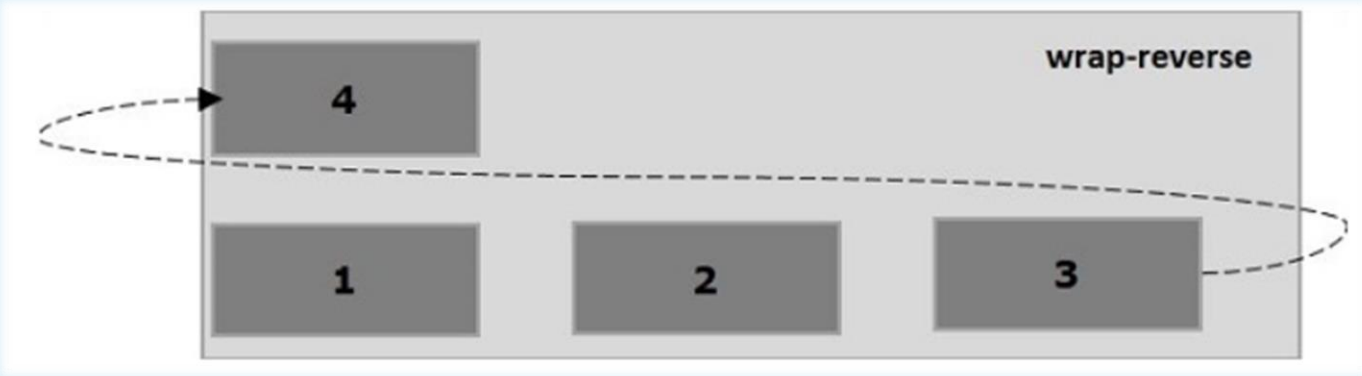


FLEX: MULTILÍNEA (DESBORDE)

- Aunque flexbox es un modelo unidimensional, es posible lograr que los ítems flex sean repartidos en varías líneas.
- Cuando los ítems no caben en una sola línea pueden repartirse en varias añadiendo la propiedad **flex-wrap**, con los siguientes valores
  - nowrap: ítems en una sola línea (por defecto)
  - wrap: los elementos se disponen en múltiples líneas si no caben. De arriba abajo
  - wrap-reverse: igual que con wrap pero en dirección inversa.

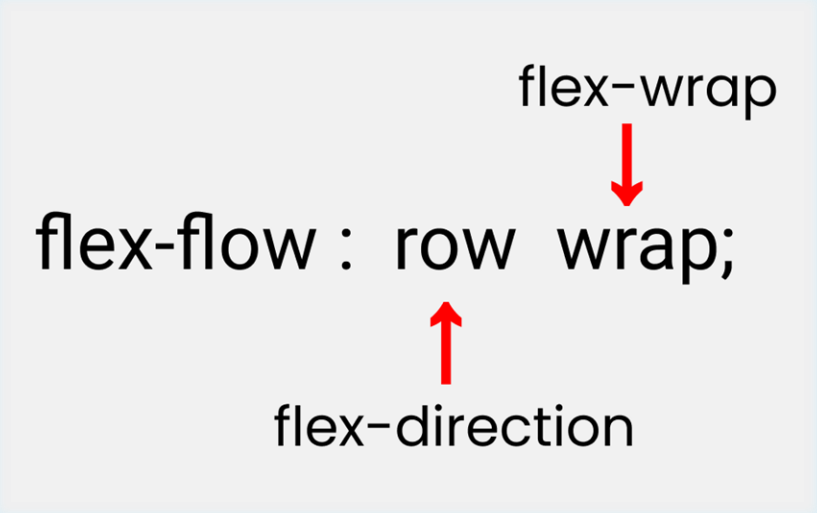
FLEX: MULTILÍNEA (DESBORDE)



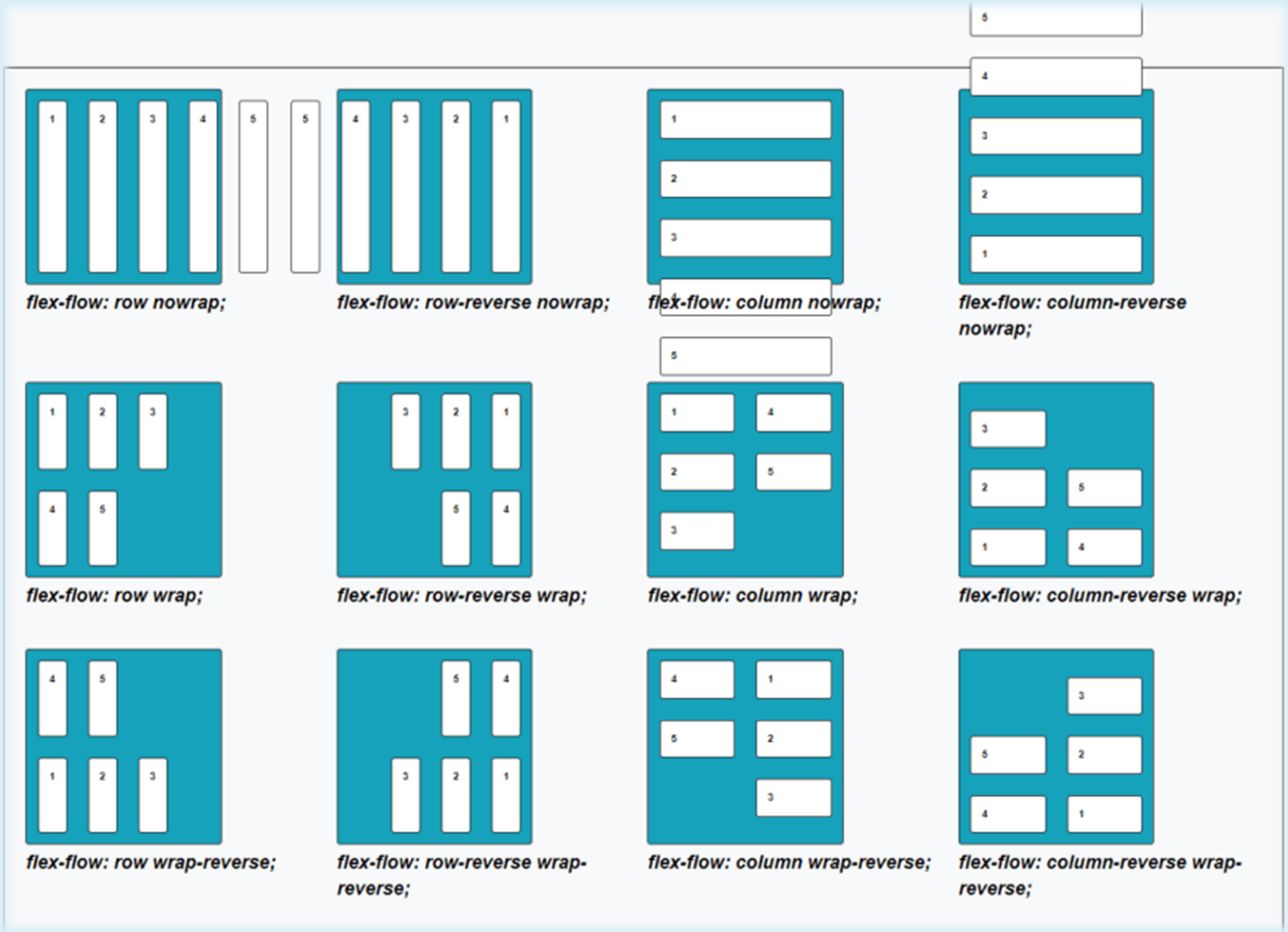


FLEX: SHORTHAND FLEX-FLOW

- Se pueden combinar las propiedades flex-direction y flex-wrap en la propiedad abreviada **flex-flow** . El primer valor especificado es flex-direction y el segundo valor es flex-wrap.



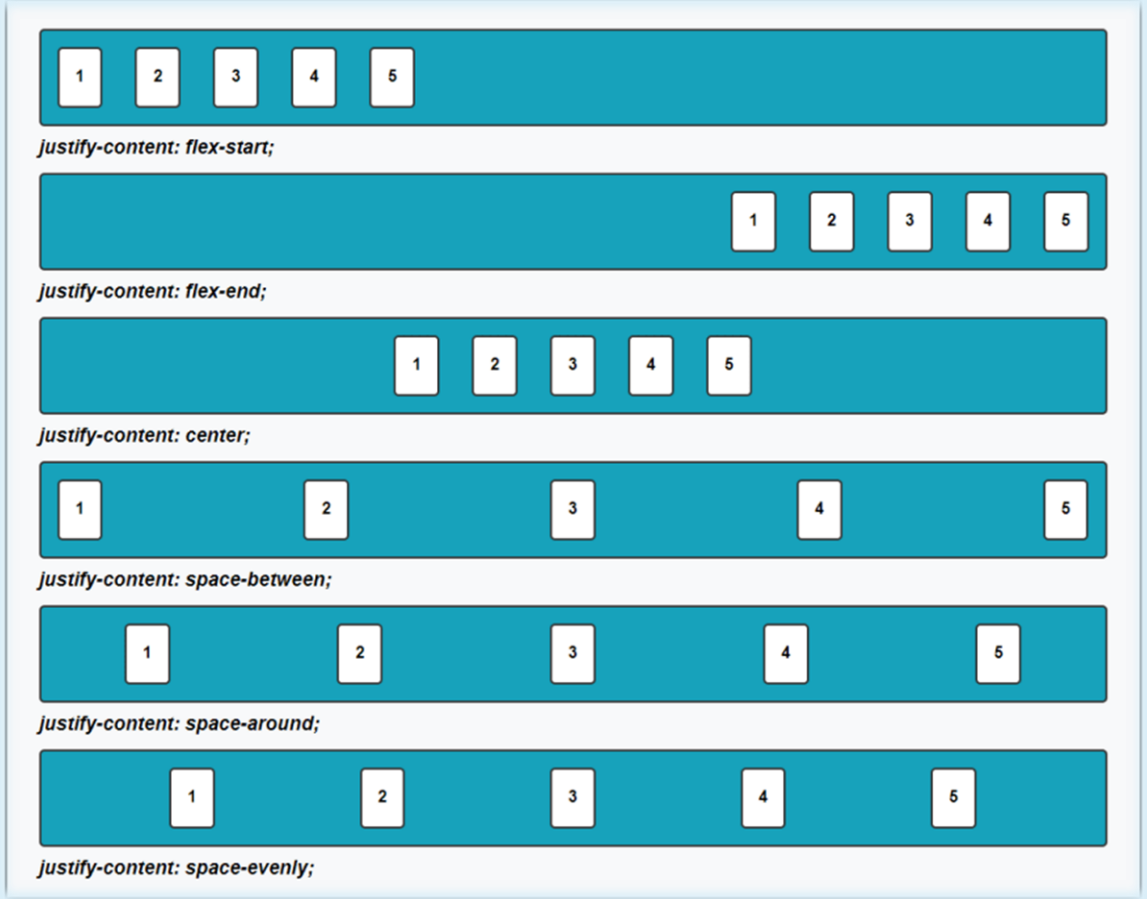
FLEX: SHORTHAND FLEX-FLOW



FLEX: ALINEACIÓN HORIZONTAL

– Para colocar los ítems mediante una disposición concreta a lo largo del eje principal (por defecto en horizontal) modificamos la propiedad **justify-content**.

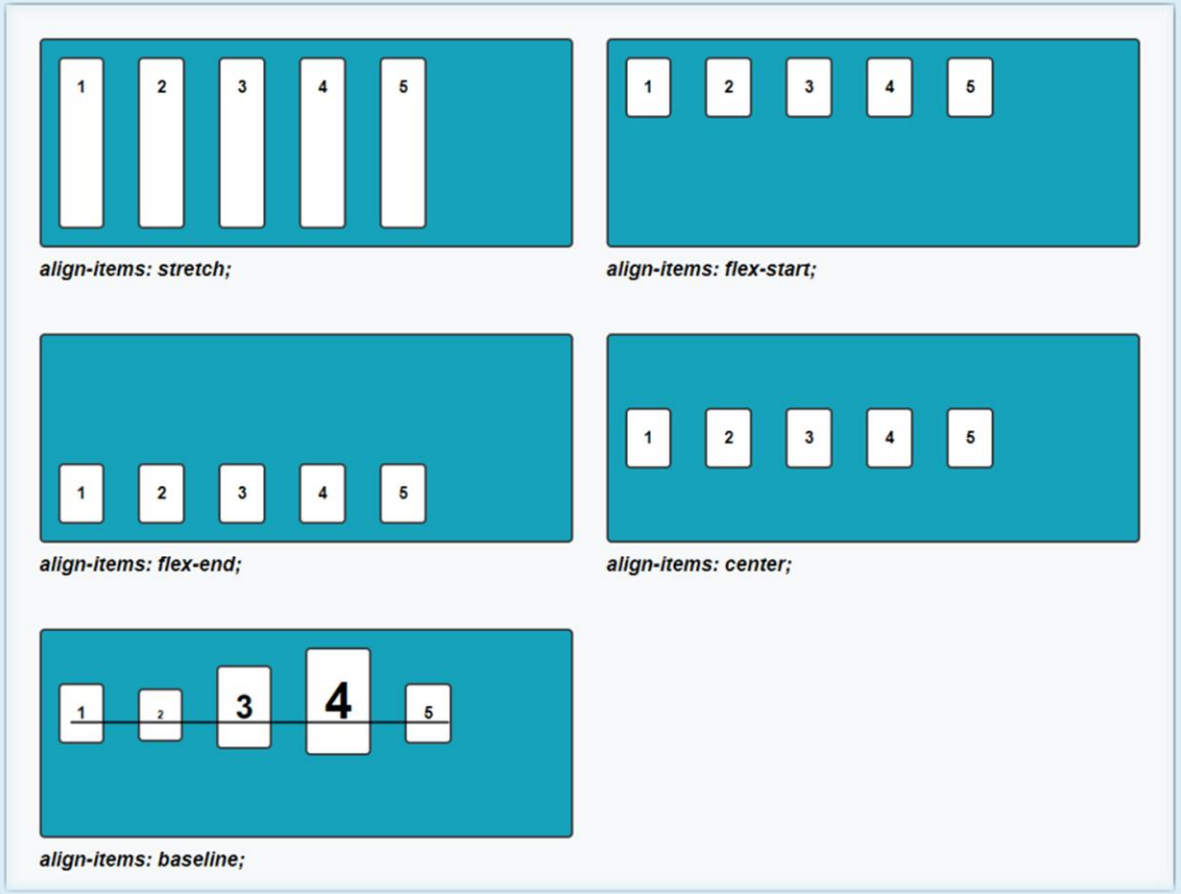
- flex-start (defecto)
- flex-end
- center
- space-between
- space-around
- space-evenly



FLEX: ALINEACIÓN VERTICAL

– Para colocar los ítems mediante una disposición concreta a lo largo del eje secundario (por defecto en vertical) modificamos la propiedad **align-items**.

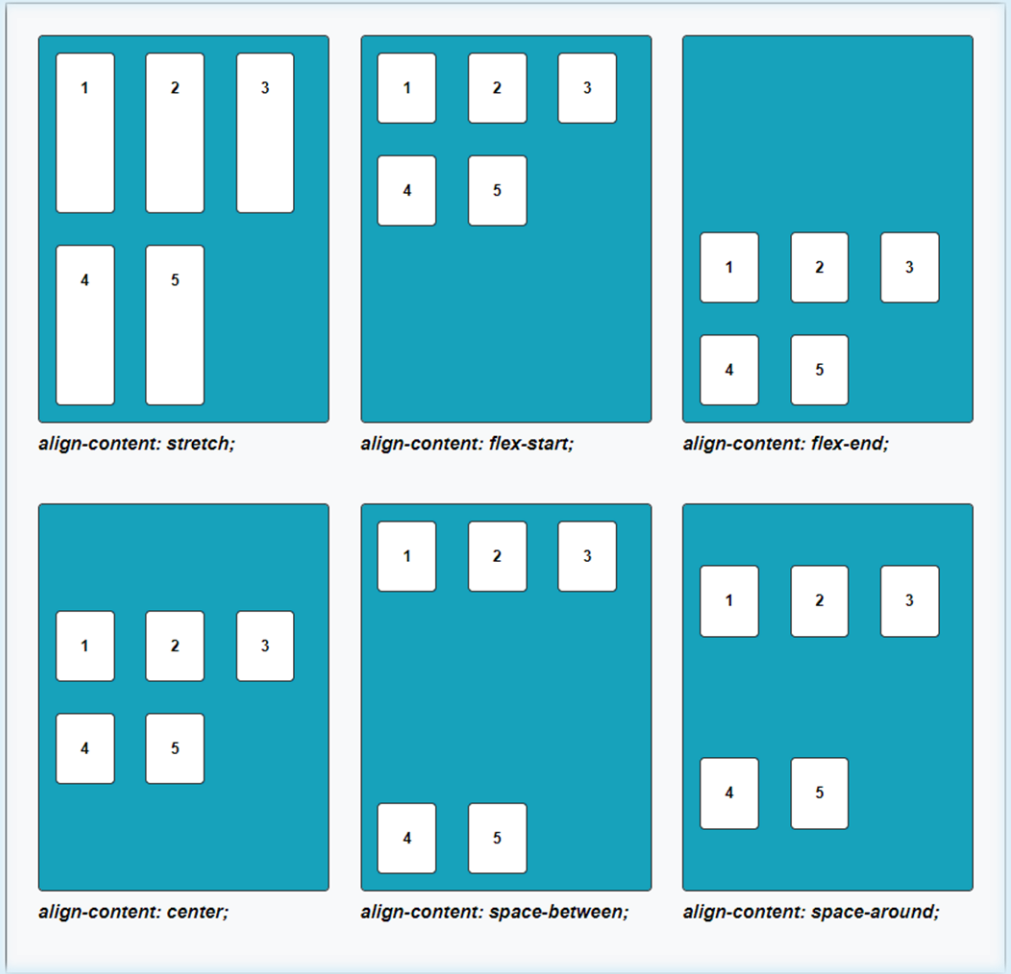
- stretch (defecto)
- flex-start
- flex-end
- center
- baseline



FLEX: ALINEACIÓN VERTICAL CON VARIAS LÍNEAS (WRAP)

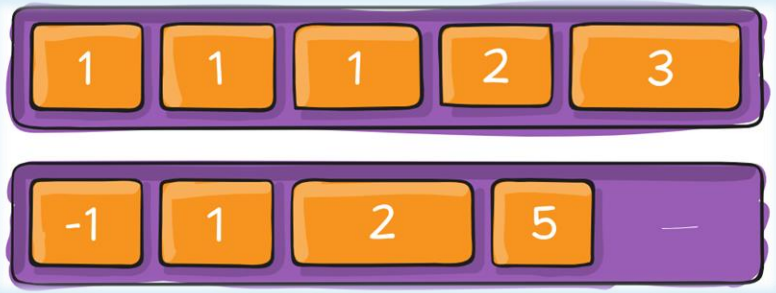
– Cuando tengo varias líneas para alinear verticalmente usamos la propiedad **align-content**.

- stretch (defecto)
- flex-start
- flex-end
- center
- space-between
- space-around
- space-evenly



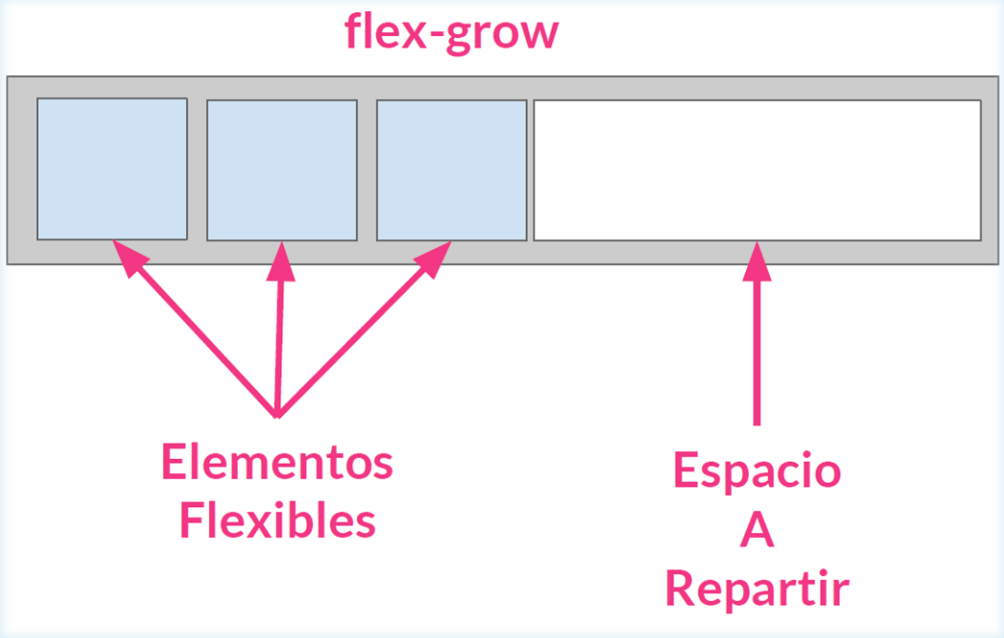
FLEX: ORDEN DE LOS ELEMENTOS FLEXIBLES (ÍTEMS)

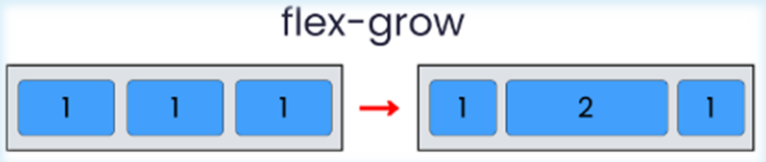
- Podemos modificar el orden en el que se van a mostrar los elementos flexibles añadiéndoles la propiedad **order** (valor por defecto 0)
- Si indicamos un order con un valor numérico, irá recolocando los ítems según su número, posicionando antes los ítems con número más pequeño (incluso valores negativos) y después los ítems con números más altos.



FLEX: TAMAÑO DE LOS ELEMENTOS FLEXIBLES (ÍTEMS)

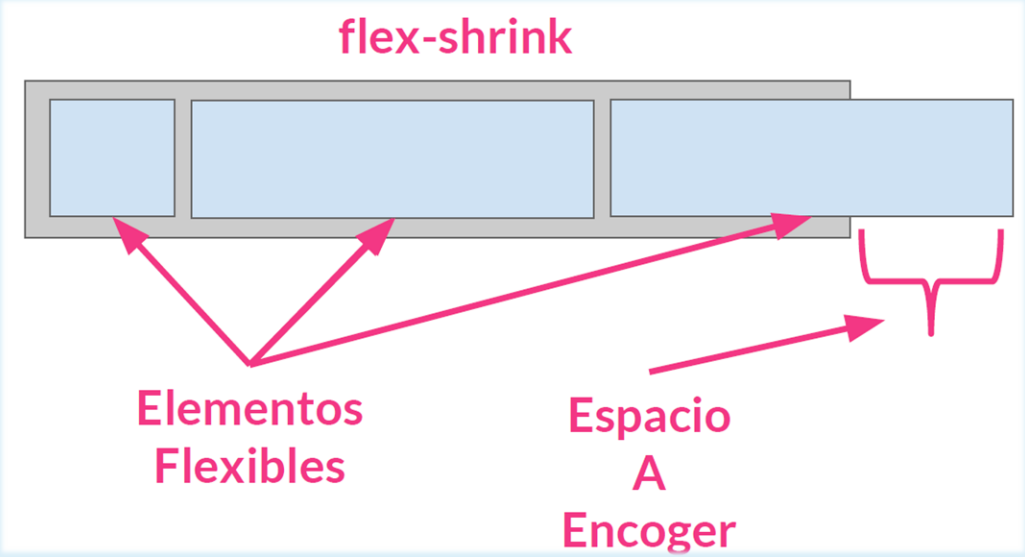
- **flex-grow**: factor de crecimiento de un elemento flexible cuando se distribuye el espacio restante. Por defecto es 1, pero debo indicárselo a todos.

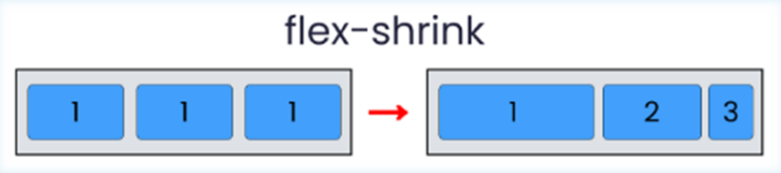




FLEX: TAMAÑO DE LOS ELEMENTOS FLEXIBLES (ÍTEMS)

- **flex-shrink**: factor de contracción de un elemento flexible cuando sobrepasan el tamaño del contenedor. Hace justo lo contrario que flex-grow



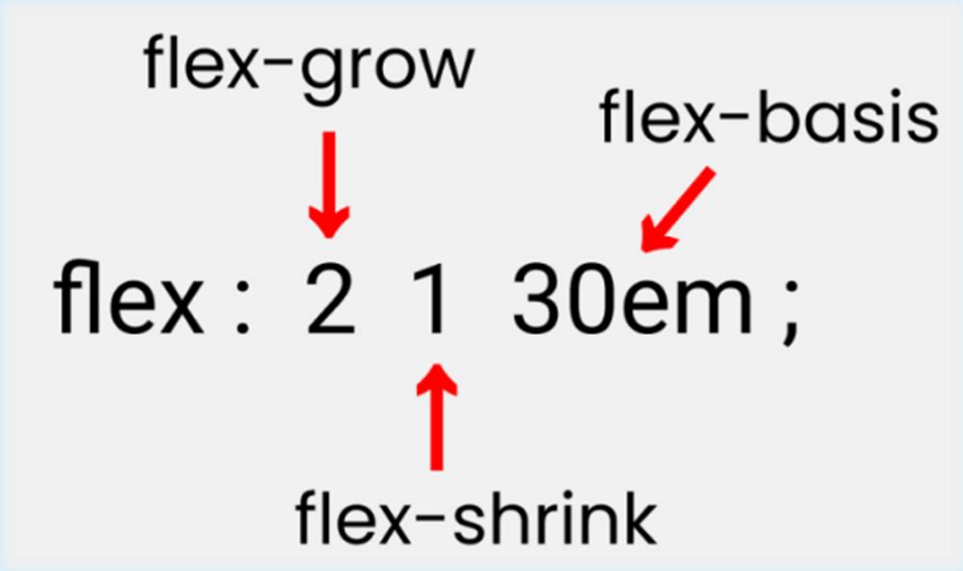


FLEX: TAMAÑO DE LOS ELEMENTOS FLEXIBLES (ÍTEMS)

- **flex-basis**: define el tamaño por defecto (de base) que tendrán los ítems antes de aplicarle la distribución de espacio.
- Generalmente, se aplica un tamaño (unidades, porcentajes, etc...), pero también se puede aplicar la palabra clave content que ajusta automáticamente el tamaño al contenido del ítem. Por defecto viene en auto, donde el navegador debe utilizar los valores de tamaño width (row) o height (si es column).

FLEX: TAMAÑO DE LOS ELEMENTOS FLEXIBLES (ÍTEMS)

- La propiedad **flex** nos sirve para escribir las 3 propiedades propias del tamaño los elementos flex en una sola línea:

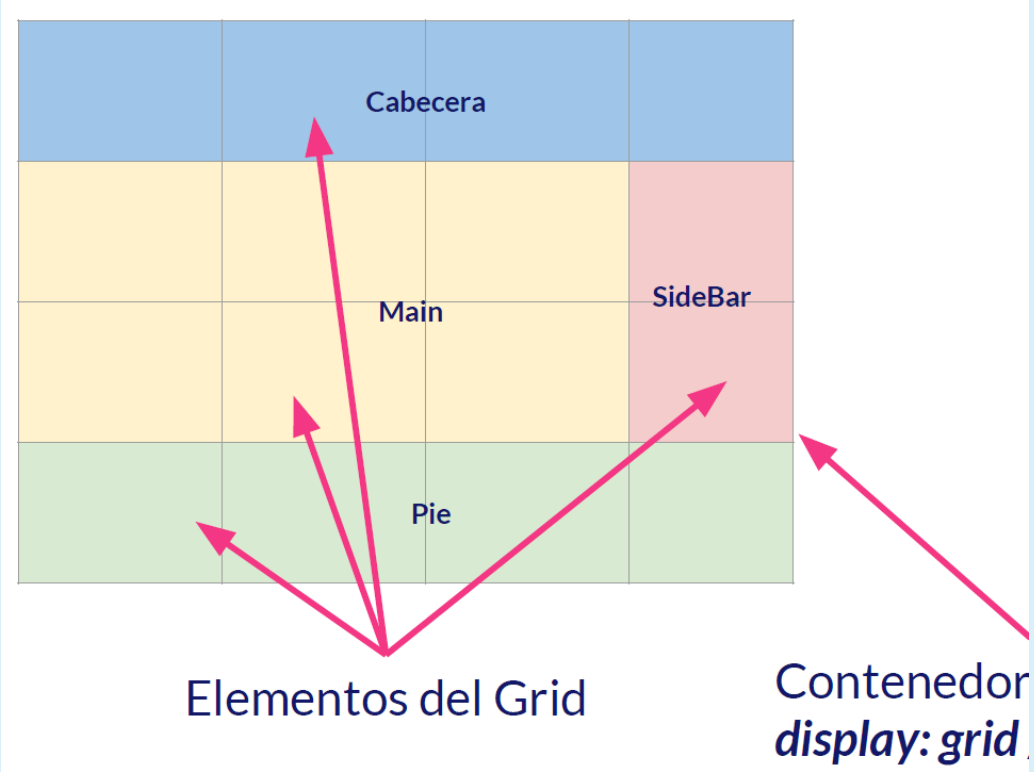


FLEX: ALINEACIÓN VERTICAL INDIVIDUAL

- Desde el contenedor flex podíamos controlar la alineación vertical de todos los elementos flexibles a la vez con la propiedad align-items.
- Si queremos que un elemento tenga una alineación propia debemos usar la propiedad **align-self**.
  - stretch (defecto)
  - flex-start
  - flex-end
  - center
  - baseline
  - auto (asigna el valor de la propiedad align-items)

GRID

– La idea principal de la maquetación con elementos GRID es que vamos a tener un elemento, **contenedor** (display: grid o display: inline-grid), que va poder controlar propiedades de los elementos grid que están dentro, modificándolas para poder distribuirlos atendiendo a una estructura compleja.



GRID: ESTRUCTURA

- Para definir la estructura del elemento contenedor, tendremos las siguientes propiedades:
  - **grid-template-columns** : número de columnas del grid y tamaño de las mismas
  - **grid-template-rows**: número de filas del grid y tamaño de las mismas.
  - **grid-row-gap:** separación entre las filas.
  - **grid-column-gap:** separación entre las columnas.

GRID: ESTRUCTURA, COLUMNAS

- **grid-template-columns**: debemos de poner tantos valores de anchura como columnas quiero que tenga el GRID.

  - grid-template-columns: 20% 50% 30%
  - grid-template-columns: 100px auto 100px 100px (la segunda columna tiene el resto del ancho del elemento contenedor grid)
  - grid-template-columns: auto auto auto auto auto
  - grid-template-columns : [id] 100px [nombre] 300px [apellidos] auto

GRID: ESTRUCTURA, FILAS

- **grid-template-rows**: debemos de poner tantos valores de altura como filas quiero que tenga el GRID.

  - grid-template-rows: 20% 50% 30%
  - grid-template-rows: 100px auto 100px 100px (la segunda fila tiene el resto de altura del elemento contenedor grid)
  - grid-template-rows: auto auto auto auto auto
  - grid-template-rows : [uno] 100px [dos] 300px [tres] auto

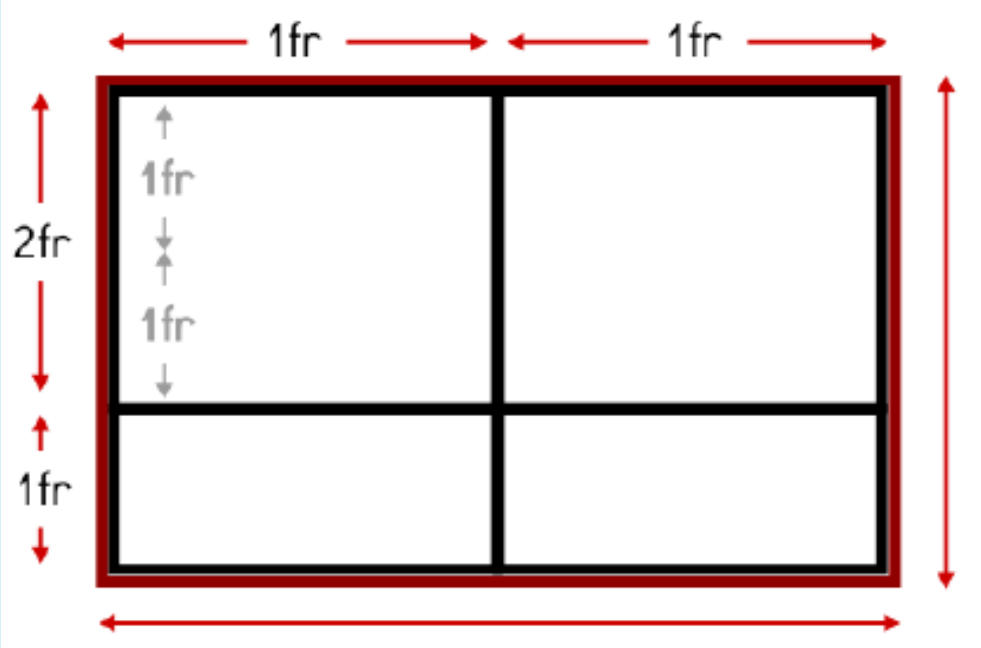
GRID: ESTRUCTURA, REPETICIÓN DE VALORES

- En las propiedades grid-template-columns y grid-template-rows podemos indicar expresiones de repetición, indicando celdas que repiten un mismo patrón de celdas varias veces.
- La expresión a utilizar sería la siguiente: repeat([núm de veces], [valor o valores]).
  - grid-template-columns: 100px repeat(2, 50px) 200px
  - grid-template-columns: repeat(3, 20% [col-start]) auto

GRID: ESTRUCTURA, UNIDAD FR

– Para definir el tamaño de las columnas y las filas podemos usar una unidad especial **fr** (fraction), que simboliza una fracción del espacio restante en el grid.

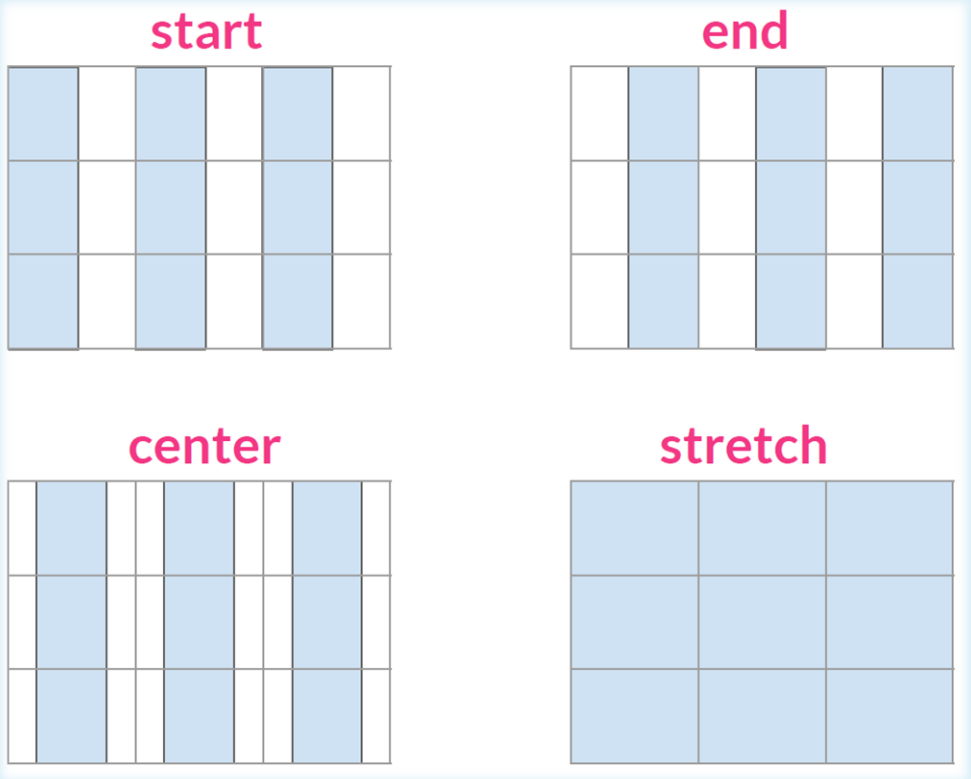
```
.grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 2fr 1fr;
}
```



GRID: ALINEACIÓN HORIZONTAL DEL CONTENIDO

– Para alinear horizontalmente el contenido que hay dentro de cada celda usaré la propiedad CSS **justify-ítems**.

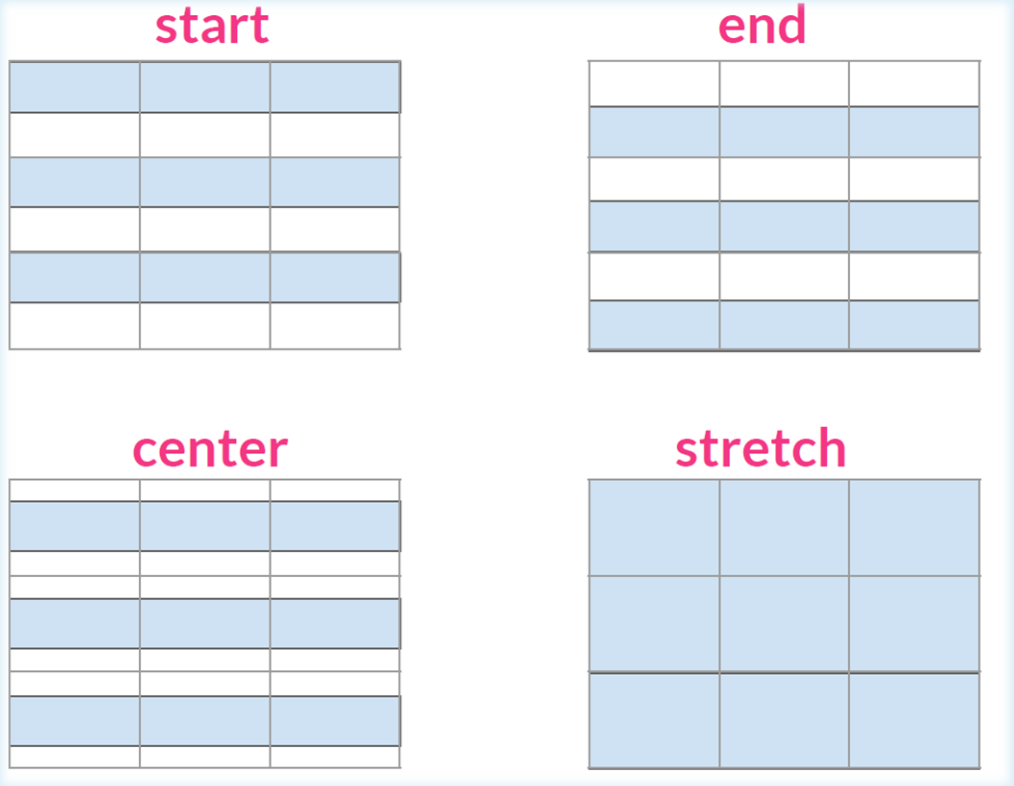
- stretch (defecto)
- start
- end
- center



GRID: ALINEACIÓN VERTICAL DEL CONTENIDO

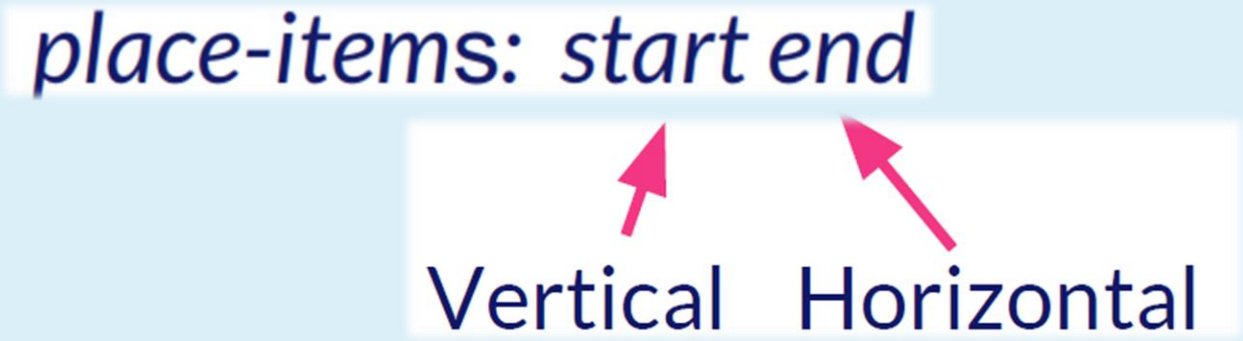
– Para alinear verticalmente el contenido que hay dentro de cada celda usaré la propiedad CSS **align-ítems**.

- stretch (defecto)
- start
- end
- center



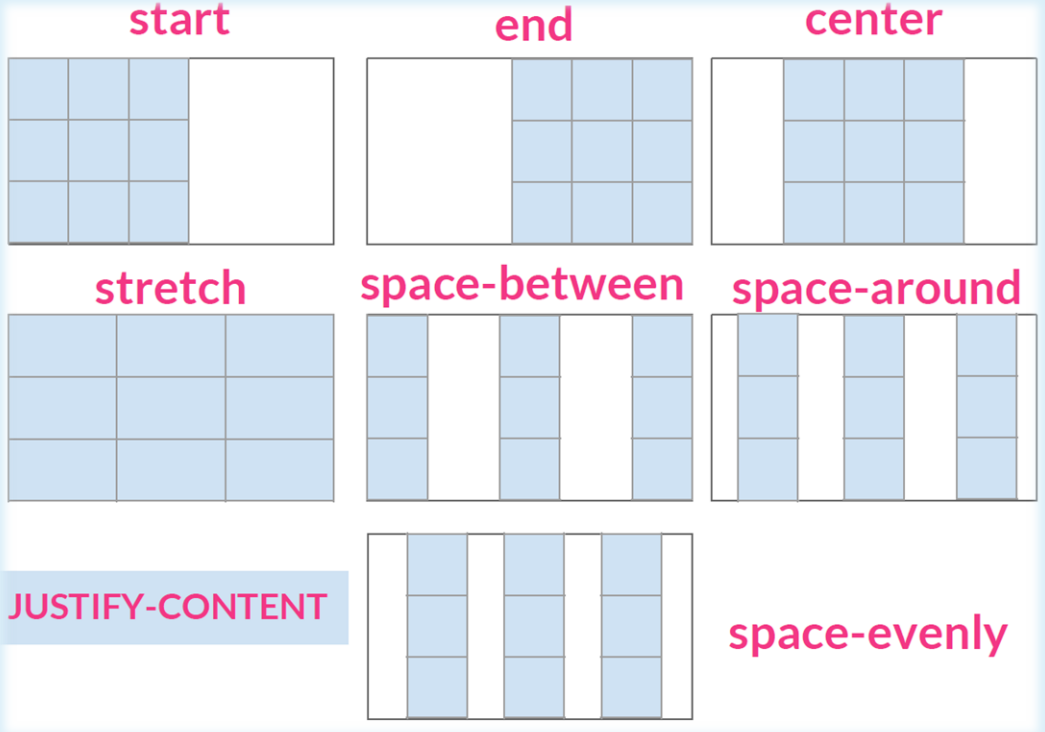
GRID: ALINEACIÓN VERTICAL Y HORIZONTAL

- Se pueden unir las dos alineaciones anteriores en la propiedad **place-items**.



GRID: DISTRIBUCIÓN HORIZONTAL DENTRO DEL CONTENEDOR

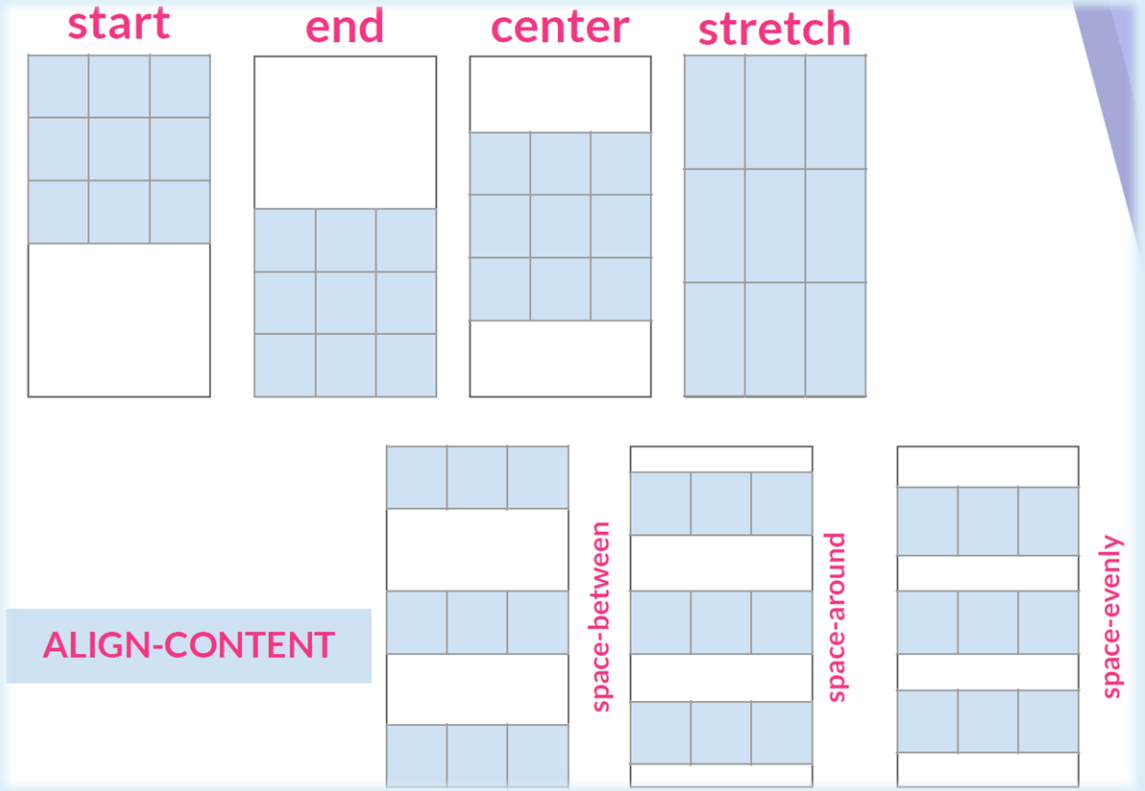
- En el caso de que los elementos GRID (celdas) no ocupen todo el ancho del contenedor GRID podemos distribuirlos con la propiedad **justify-content**. (igual que en flex)
  - start (defecto)
  - end
  - center
  - stretch
  - space-between
  - space-around
  - space-evenly



GRID: DISTRIBUCIÓN VERTICAL DENTRO DEL CONTENEDOR

– En el caso de que los elementos GRID (celdas) no ocupen todo el alto del contenedor GRID podemos distribuirlos con la propiedad **align-content**.

- start (defecto)
- end
- center
- stretch
- space-between
- space-around
- space-evenly



GRID: ALINEACIÓN VERTICAL Y HORIZONTAL DENTRO DEL CONTENEDOR

- Se pueden unir las dos alineaciones anteriores, igual que ocurría con la alineación del contenido, (place-ítems), en la propiedad **place-content**.

GRID: ELEMENTOS GRID

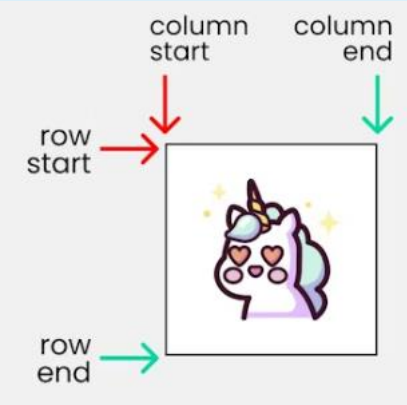
- Los elementos GRID son los hijos directos (dentro del árbol DOM) del elemento con la propiedad CSS display: grid (o inline-grid).
- Podremos definir:
  - Área del GRID, región o conjunto de celdas que va ocupar.
  - Alineación horizontal de manera individual.
  - Alineación vertical de manera individual.
  - Colocación implícita si no especificamos nada.

** No les afectan las propiedades float, display: inline-block, display: table-cell, vertical-align, etc..

GRID: ÁREA A OCUPAR POR LOS ELEMENTOS

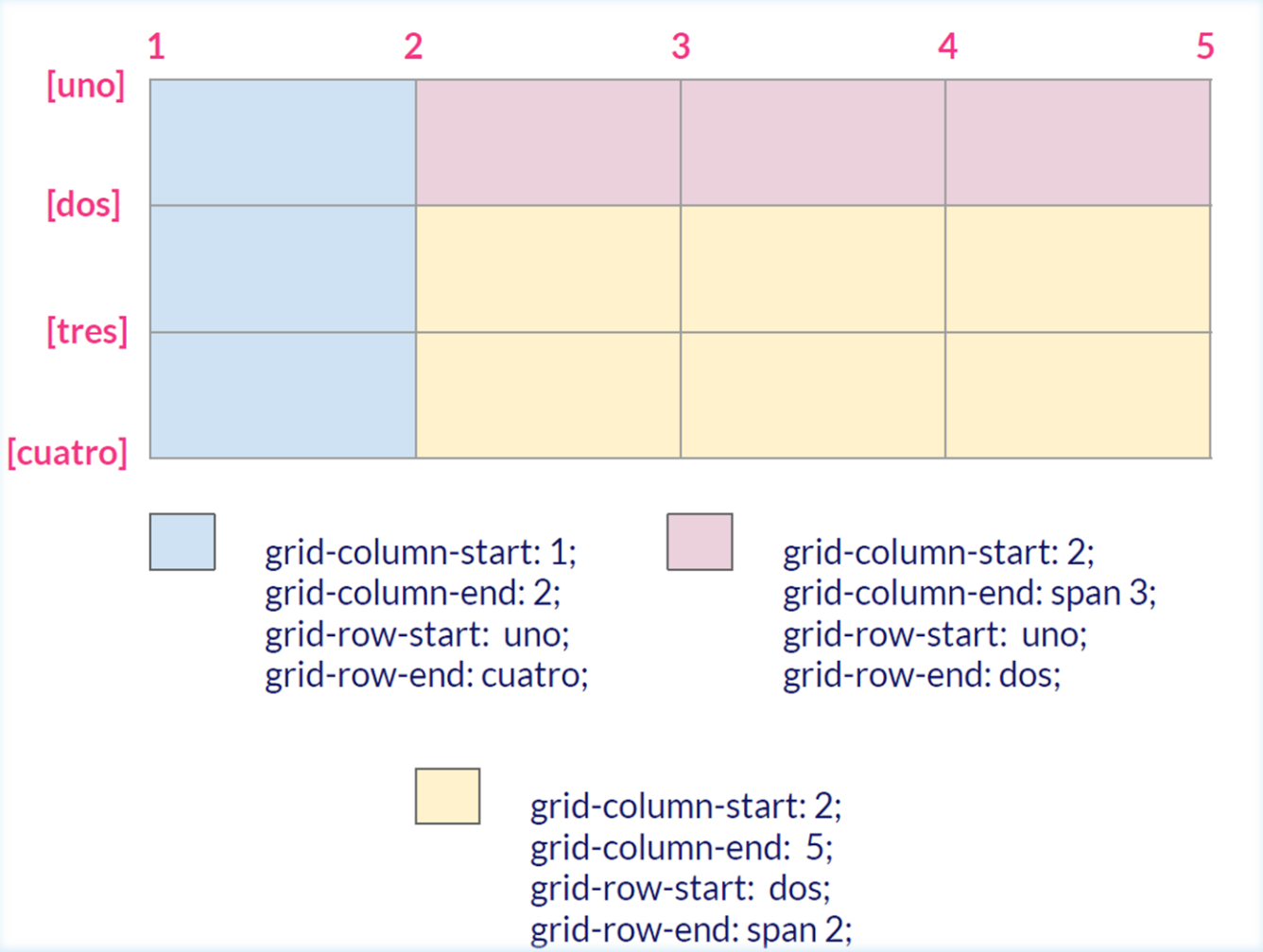
– Podemos indicar a cada uno de los elementos GRID las celdas del contenedor que va ocupar modificando las siguientes propiedades:

- **grid-column-start:** columna donde empieza el elemento o ítem.
- **grid-column-end:** columna donde termina el elemento o ítem.
- **grid-row-start:** fila donde empieza el elemento o ítem.
- **grid-row-end:** fila donde termina el elemento o ítem.



(los números de línea también podemos nombrarlos, indicando las etiquetas cuando definimos la estructura del elemento contenedor a través de las propiedades grid-template-columns y grid-template-rows)

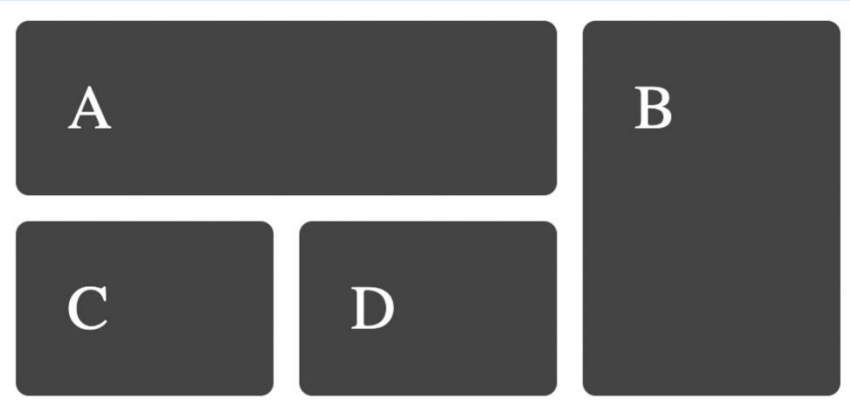
GRID: ÁREA A OCUPAR POR LOS ELEMENTOS



GRID: ÁREA A OCUPAR POR LOS ELEMENTOS

– Podemos unir estas propiedades:

- **grid-column:** start / end.
- **grid-row:** start / end.




```
.a {
    grid-column: 1 / 3;
    grid-row: 1;
}
.b {
    grid-column: 3 ;
    grid-row: 1 / 3;
}
.c {
    grid-column: 1 ;
    grid-row: 2 ;
}
.d {
    grid-column: 2;
    grid-row: 2;
}
```

GRID: GRID1.HTML

```
<div class="container">
  <div id="rojo">ROJO</div>
  <div id="azul">AZUL</div>
  <div id="amarillo">AMARILLO</div>
</div>
```

Contenedor con 4 columnas (25%) y 3 filas (100px)

GRID: ALINEACIÓN HORIZONTAL DE UN ÚNICO ELEMENTO O ÁREA

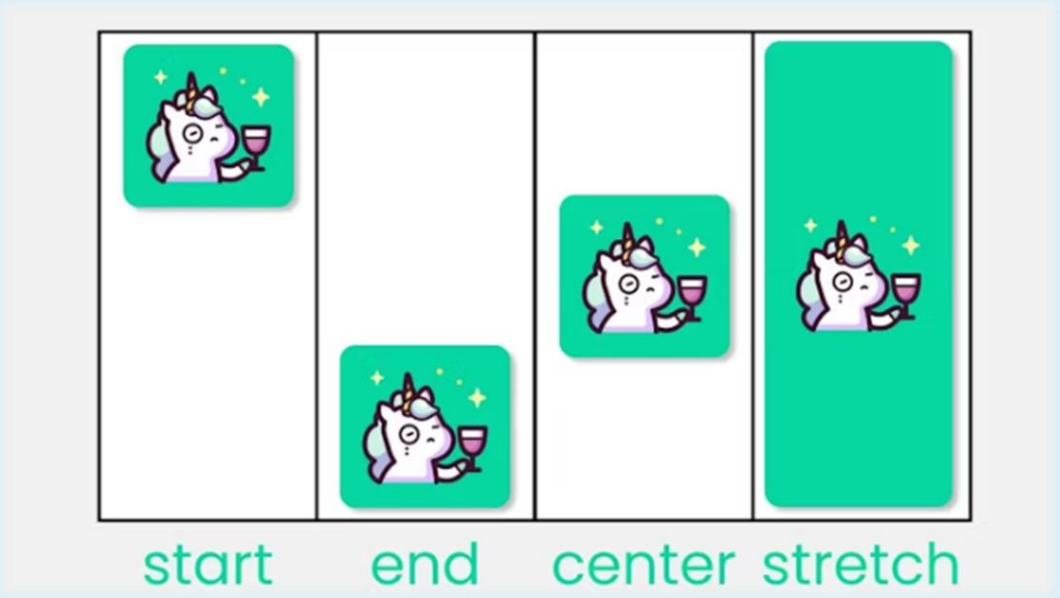
- Para alinear horizontalmente el contenido que hay dentro de una celda o área determinada usaré la propiedad CSS **justify-self** (tiene los mismo valores que tenía la propiedad justify-ítems del contenedor grid).

  - stretch (defecto)
  - start
  - end
  - center

GRID: ALINEACIÓN VERTICAL DE UN ÚNICO ELEMENTO O ÁREA

- Para alinear verticalmente el contenido que hay dentro de una celda o área determinada usaré la propiedad CSS **align-self** (tiene los mismo valores que tenía la propiedad align-ítems del contenedor grid).

  - stretch (defecto)
  - start
  - end
  - center



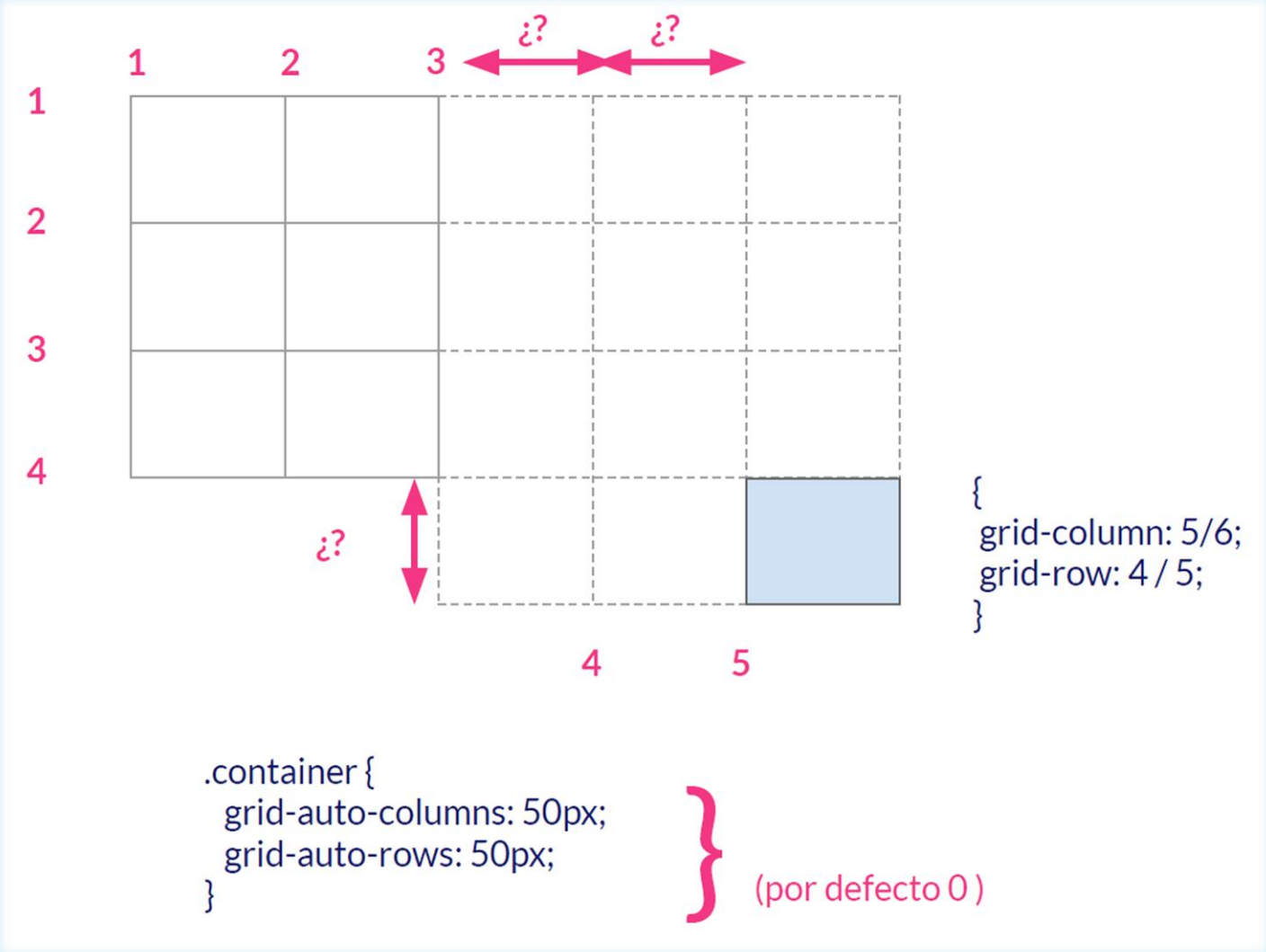
GRID: ALINEACIÓN VERTICAL Y HORIZONTAL

– Se pueden unir las dos alineaciones anteriores, igual que ocurría con place-ítems y place-content, en la propiedad **place-self**.

GRID: COLOCACIÓN IMPLÍCITA

- La colocación implícita de los elementos GRID es lo que sucede cuando colocamos elementos GRID fuera de la estructura que se ha definido en el contenedor GRID o cuando no les damos posición.
- Para que se produzca un ajuste automático habrá que definir las siguientes propiedades en el **contenedor** grid:
  - **grid_auto_columns**: indican el ancho automático de las columnas
  - **grid_auto_rows**: para definir el alto automático de las filas
  - **grid_auto_flow**: indica el flujo de elementos que se irán colocando automáticamente intentando rellenar huecos.

GRID: COLOCACIÓN IMPLÍCITA

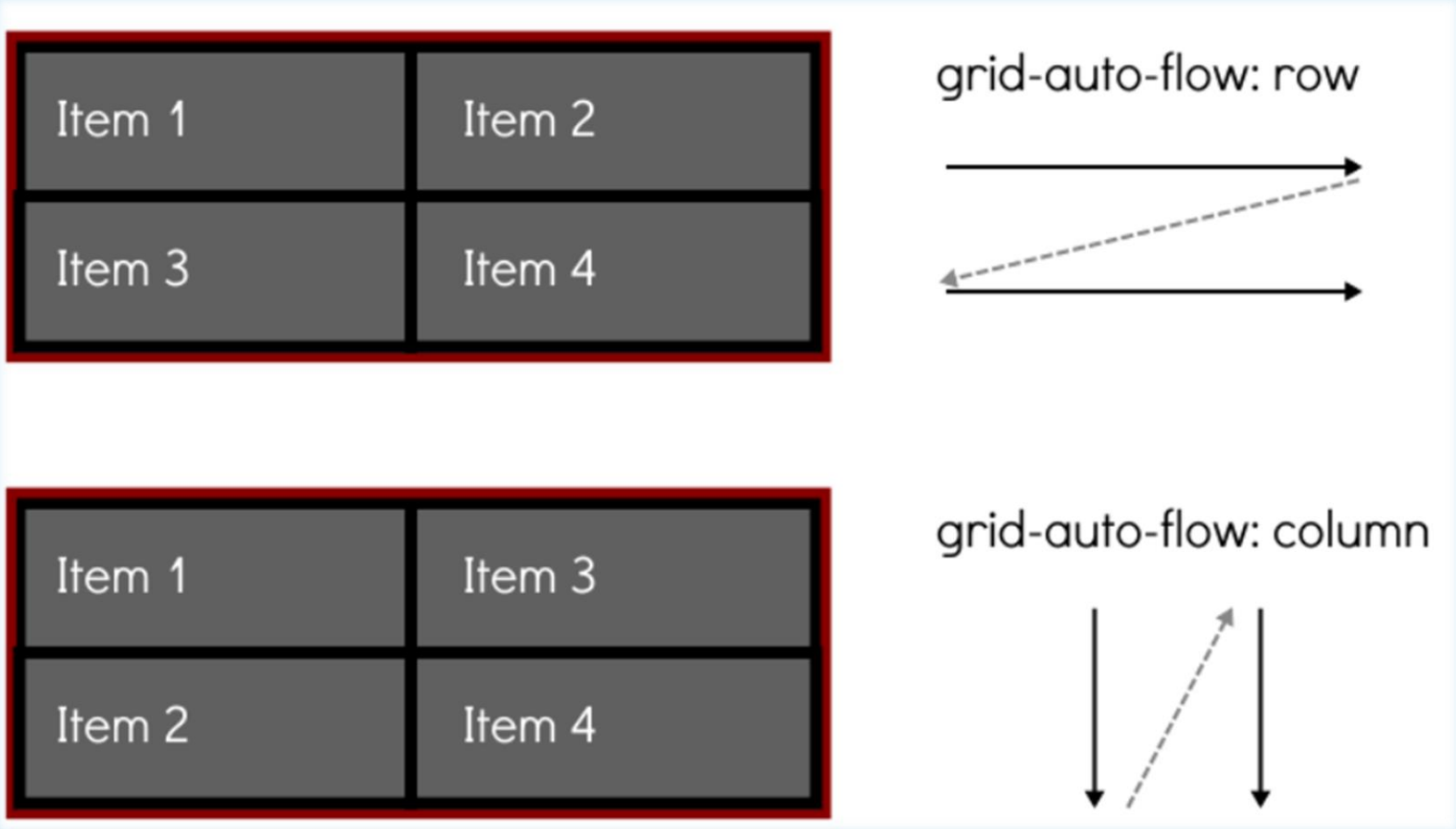


GRID: COLOCACIÓN IMPLÍCITA

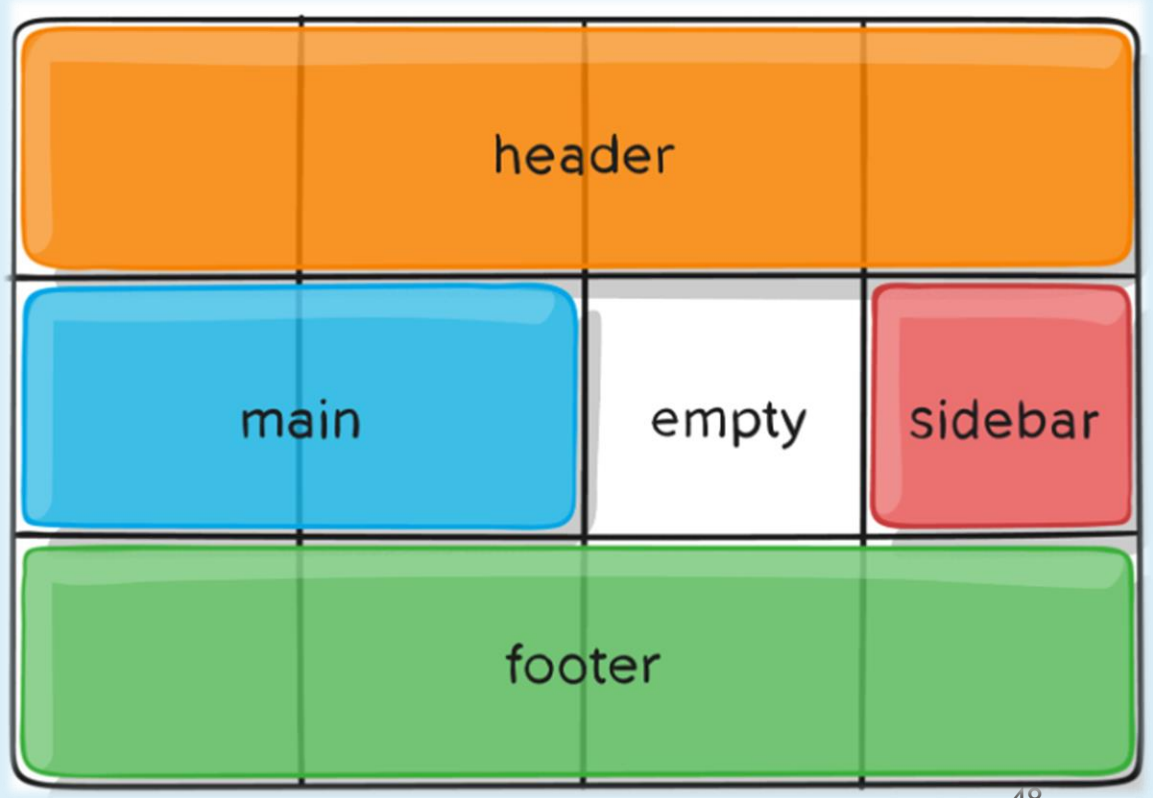
- Si tengo elementos sin definir una colocación en el contenedor GRID la propiedad del contenedor **grid-auto-flow** define cómo se van a colocar esos elementos.

  - row (defecto): rellena las filas primero.
  - column: rellena las columnas primero.
  - dense: intenta rellenar primero los huecos si el elemento es más pequeño y puede colocarse. Puede cambiar el orden de los elementos así que hay que tener cuidado.

GRID: COLOCACIÓN IMPLÍCITA



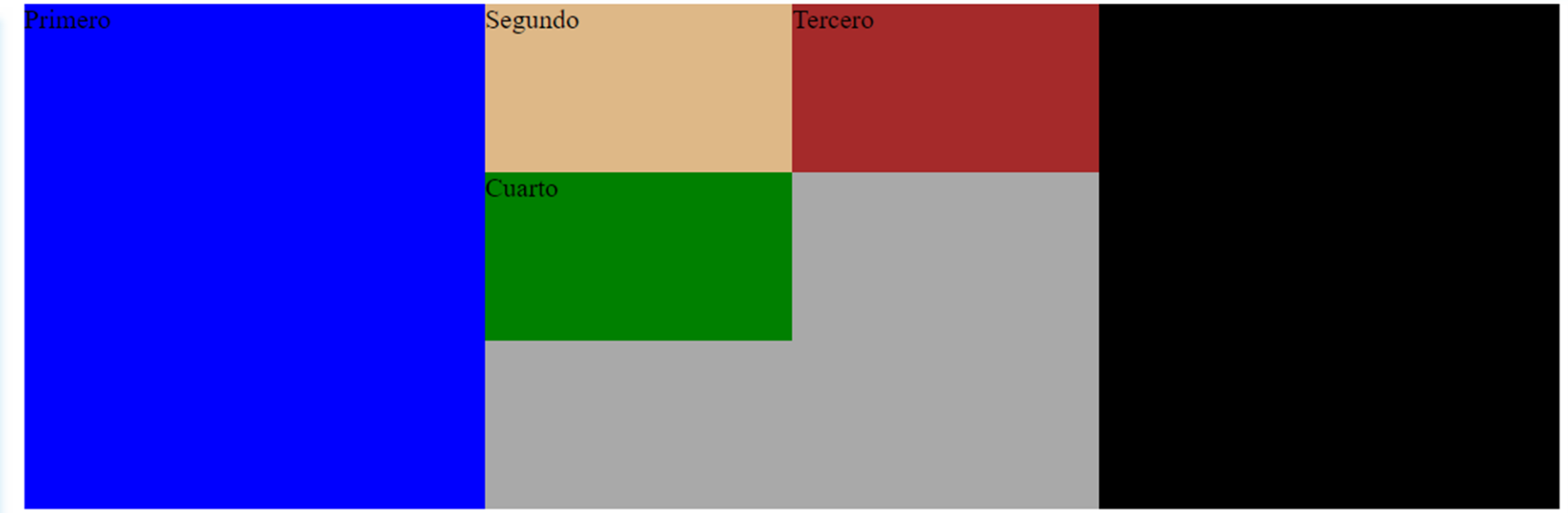
GRID: COLOCACIÓN IMPLÍCITA



GRID: GRID2.HTML

```
<div class="main">
  <div>Primero</div>
  <div>Segundo</div>
  <div>Tercero</div>
  <div>Cuarto</div>
  <div>Quinto</div>
</div>
```

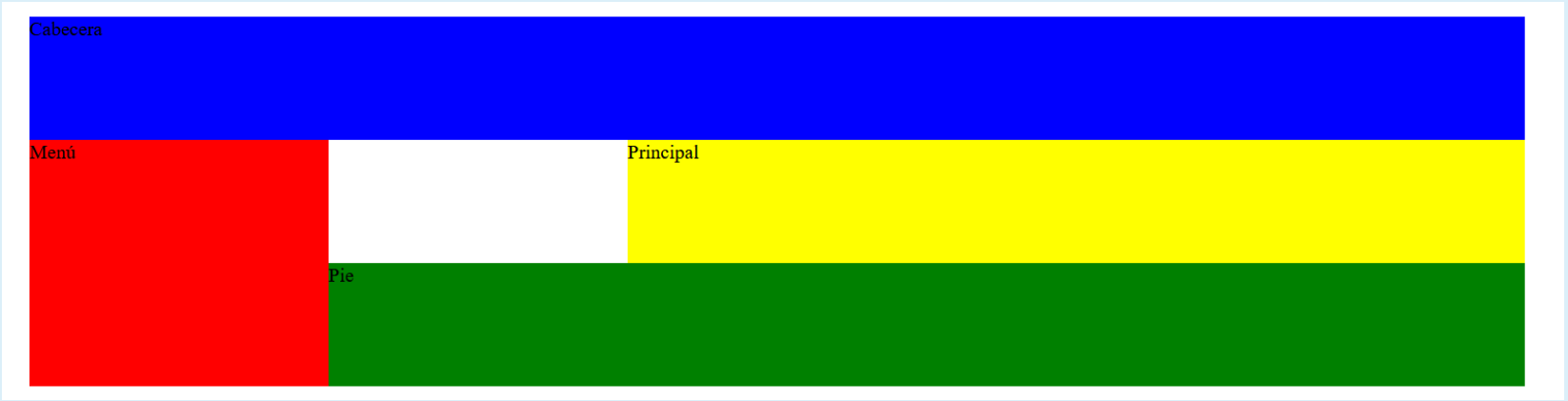
Contenedor con 4 columnas (30% 20% 20% 30%) y 3 filas (100px)



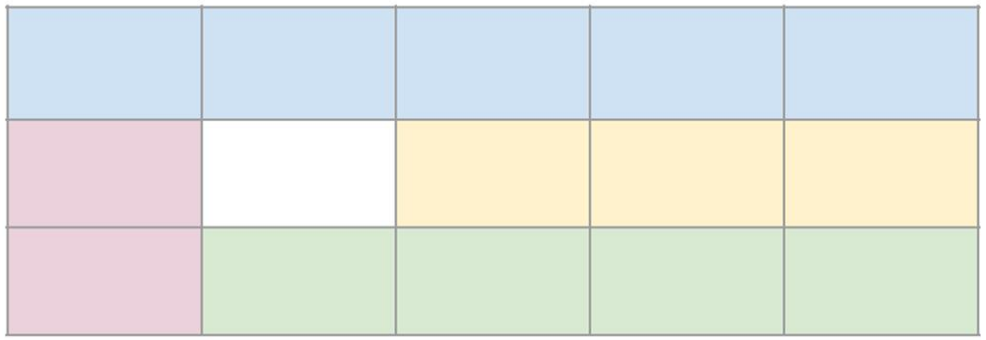
GRID: ÁREAS

– Si no queremos especificar el tamaño para cada elemento del GRID podemos darles nombre a su propiedad **grid-area** y usar esos nombres en la propiedad **grid-template-area** del elemento contenedor.

```
grid-template-columns: repeat(5, 20%);
grid-template-rows: repeat(3, 100px);
```



GRID: ÁREAS



```
.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

GRID: ÁREAS

```
grid: rows / columns
```

```
#layout {
  display: grid;
  grid: 100px 400px 100px / 200px 1fr 200px;
}
```

```
#layout {
  display: grid;
  grid:
    [header-start]  "header  header  header" 100px
    [content-start] "ads     main    links"  400px
    [footer-start]  "footer  footer  footer" 100px
    / [ads] 200px [main] 1fr [links] 200px;
}
```