http://www.robertsallent.com
@robertsallent

CSS14: GRID Layout

El modelo de tabla flexible de CSS3

Robert Sallent
www.robertsallent.com

SOC Servei d'Ocupació de Catalunya
CIFO Sabadell- Mancomunitat (Vallès)
Generalitat de Catalunya
Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Índice

- *Grid Layout.*
- Contenedores y elementos.
- Filas, columnas y huecos.
- `grid-template-columns`.
- Alineando el contenido.
- Líneas grid.
- Colocando elementos.
- Grid dentro de grid.
- `grid-column`, `grid-row`, `grid-area`
- Definiendo áreas.

- Ejercicios

Grid layout

- El módulo **Grid Layout** ofrece un sistema de posicionamiento de los elementos basado en una estructura de tabla, con filas y columnas.

- De esta forma es más sencillo para los diseñadores ubicar los elementos en la posición deseada.

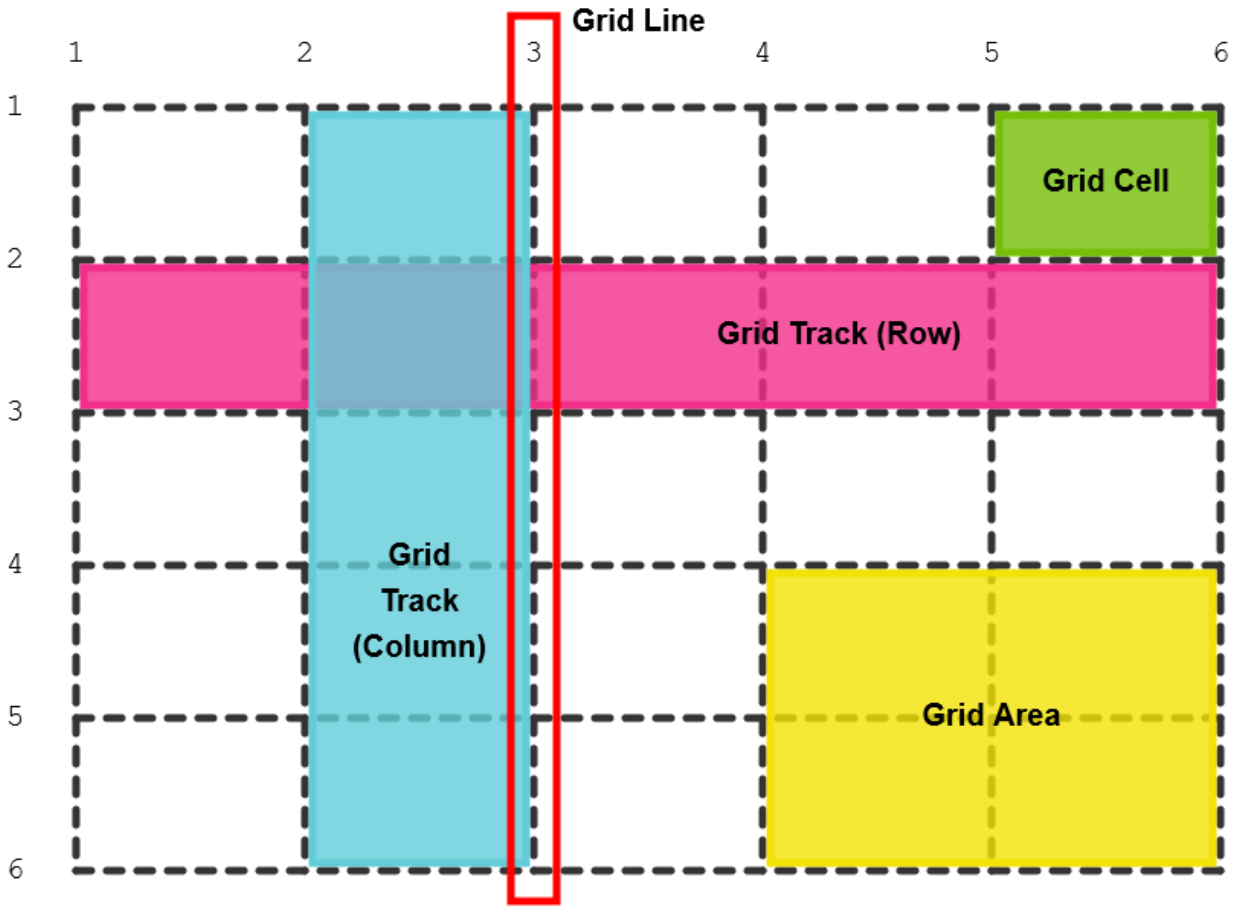
- No sustituye a `Flex`, se complementan.

Grid layout

- Una tabla o parrilla (*grid*) consiste en un elemento padre, con un conjunto de elementos hijos.

- Existen, por tanto:
  - **Contenedores *grid***, que tienen:
    - Filas
    - Columnas
    - Celdas
    - Áreas
  - **Elementos *grid***, colocados en las posiciones de la parrilla.

Grid layout

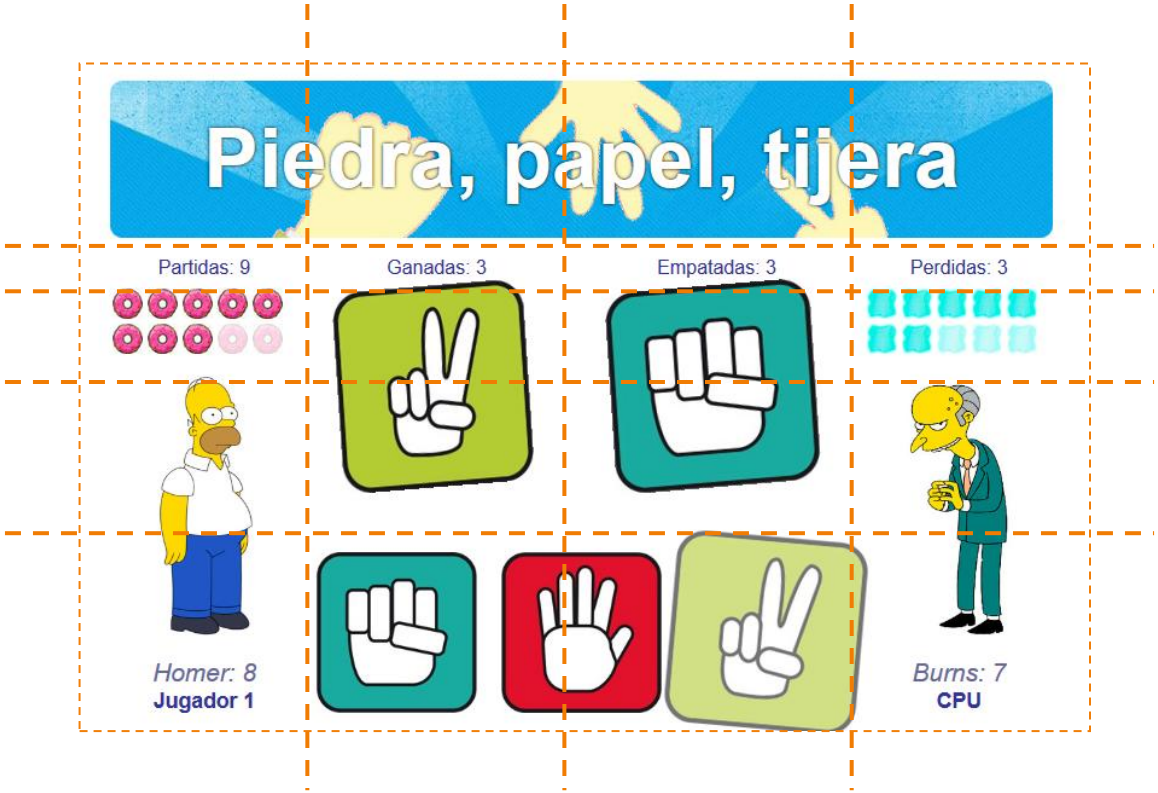

Grid Line
1
2
3
4
5
6
Grid Cell
Grid Track (Row)
Grid
Track
(Column)
Grid Area

Ejemplo (1 de 2)

Juego hecho con Grid

https://www.juegayestudia.com/juego/ppt

Ejemplo (2 de 2)



https://www.juegayestudia.com/juego/ppt

Contenedores y elementos

- Para hacer que un elemento se comporte como un **contenedor *grid***, debemos ponerle la propiedad `display` a `grid` o `inline-grid`.

- Una vez hecho esto ya podemos colocarle dentro elementos.

- Los elementos contenidos en un `grid-container`, serán elementos *grid* (`grid-items`).

Para los ejemplos

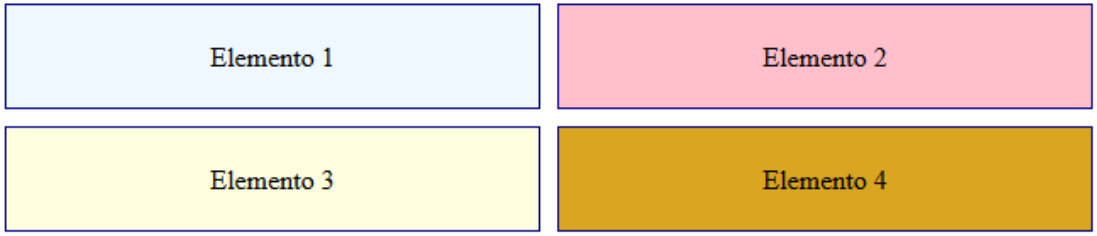
- El *CSS* que se muestra a la derecha se usa en los próximos ejemplos.
- Lo enseño aquí una única vez para no tener que repetirlo en todas las diapositivas.

```css
*{
    margin: 0px;
    padding: 0px;
}

body{padding: 10px;}

div{
    text-align: center;
    border: solid 1px darkblue;
    padding: 20px;
}

div:nth-of-type(1){background-color:aliceblue;}
div:nth-of-type(2){background-color:pink;}
div:nth-of-type(3){background-color:lightyellow;}
div:nth-of-type(4){background-color:goldenrod;}
```

Ejemplo 1 grid layout

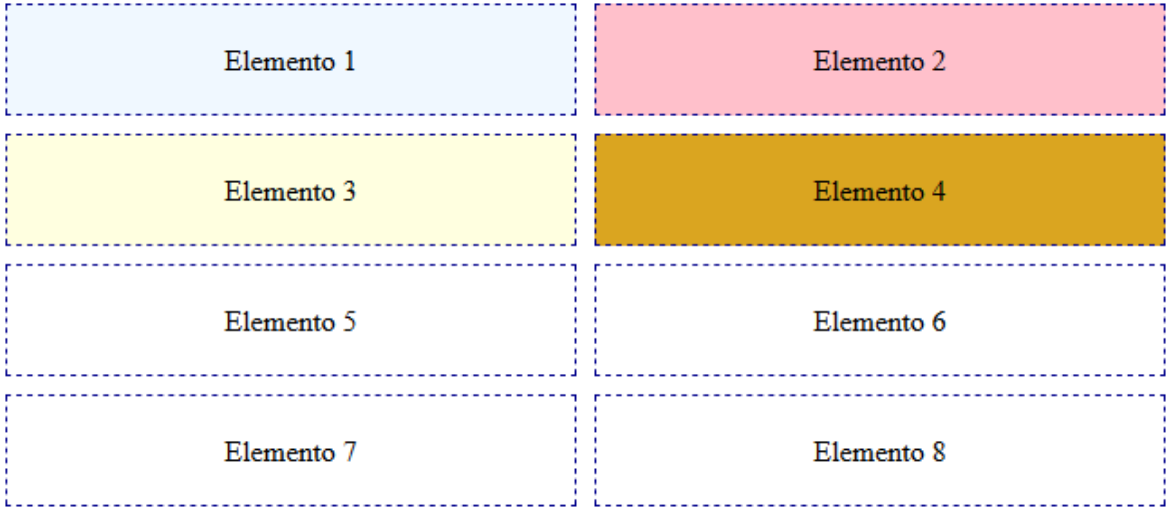
```
<body class="grid-container" />
    <div>Elemento 1</div>
    <div>Elemento 2</div>
    <div>Elemento 3</div>
    <div>Elemento 4</div>
</body>
```

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto;
   grid-gap: 10px;
}
```



Contenedores y elementos

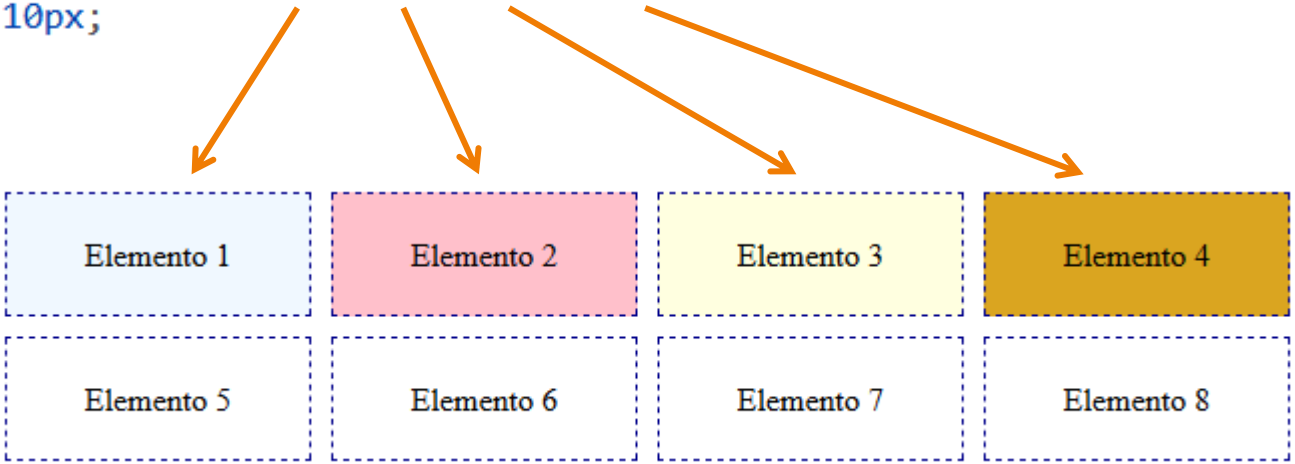
- En el ejemplo anterior, si colocáis más `divs` en el `body`, veréis cómo se van ubicando automáticamente en nuevas filas `grid`.



Contenedores y elementos

- Si modificáis la propiedad `grid-template-columns` podéis hacer más columnas.

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto auto auto;
   grid-gap: 10px;
}
```



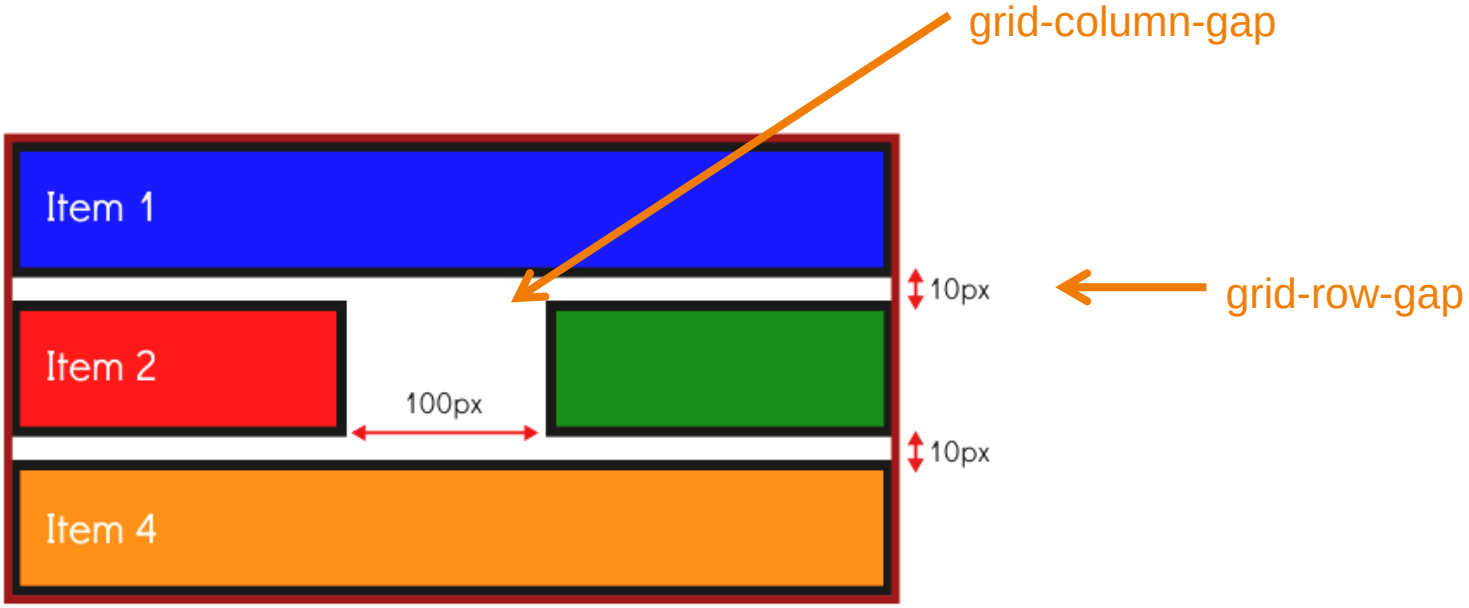
Filas, columnas y huecos

- Como ya podemos imaginar, la línea vertical de elementos es la **columna** y la línea horizontal de elementos es la **fila**.

- El espacio entre columnas se llama "*column gap*" y el espacio entre filas "*row gap*".

- Se puede ajustar el espacio entre filas y columnas usando las propiedades: `grid-column-gap`, `grid-row-gap` y `grid-gap`.

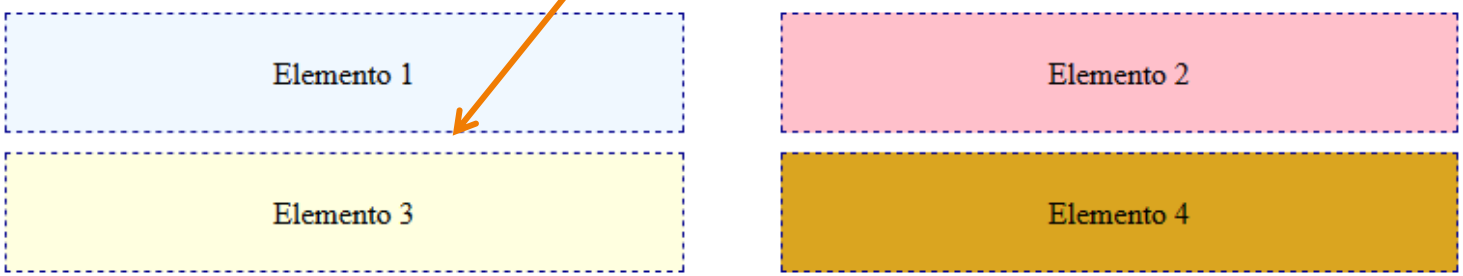
Filas, columnas y huecos



Ejemplo 2 grid-gap

```
<body class="grid-container" />
    <div>Elemento 1</div>
    <div>Elemento 2</div>
    <div>Elemento 3</div>
    <div>Elemento 4</div>
</body>
```

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto;
   grid-row-gap: 10px;
   grid-column-gap: 50px;
}
```



grid-template-columns

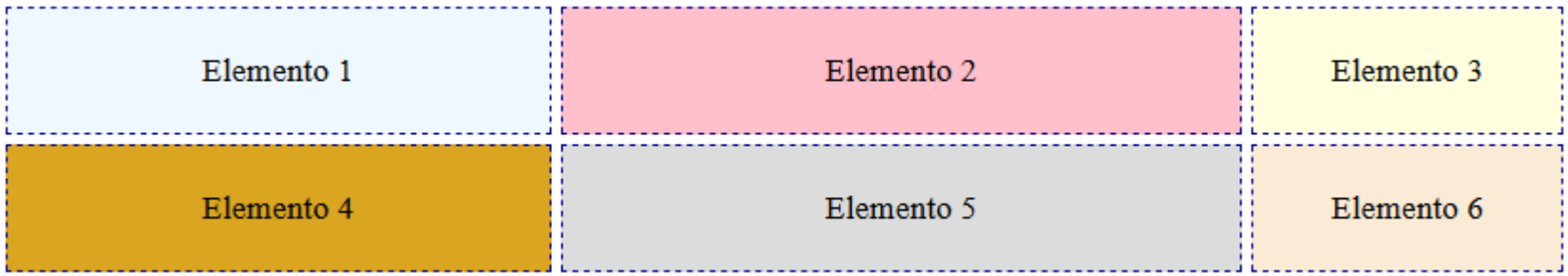
- La propiedad `grid-template-columns` define el **número de columnas** en nuestro *layout*, pudiendo definir además el **ancho** para cada columna (incluso con el valor `auto`).

- Dicha propiedad es una lista de valores separados por espacios, donde cada valor es el ancho de la columna respectiva.

Ejemplo 3 grid-template-columns

```
<body class="grid-container" />
    <div>Elemento 1</div>
    <div>Elemento 2</div>
    <div>Elemento 3</div>
    <div>Elemento 4</div>
    <div>Elemento 5</div>
    <div>Elemento 6</div>
</body>
```

```
.grid-container {
   display: grid;
   grid-template-columns: 35% auto 20%;
   grid-gap: 5px;
}
```



Recapitulando…

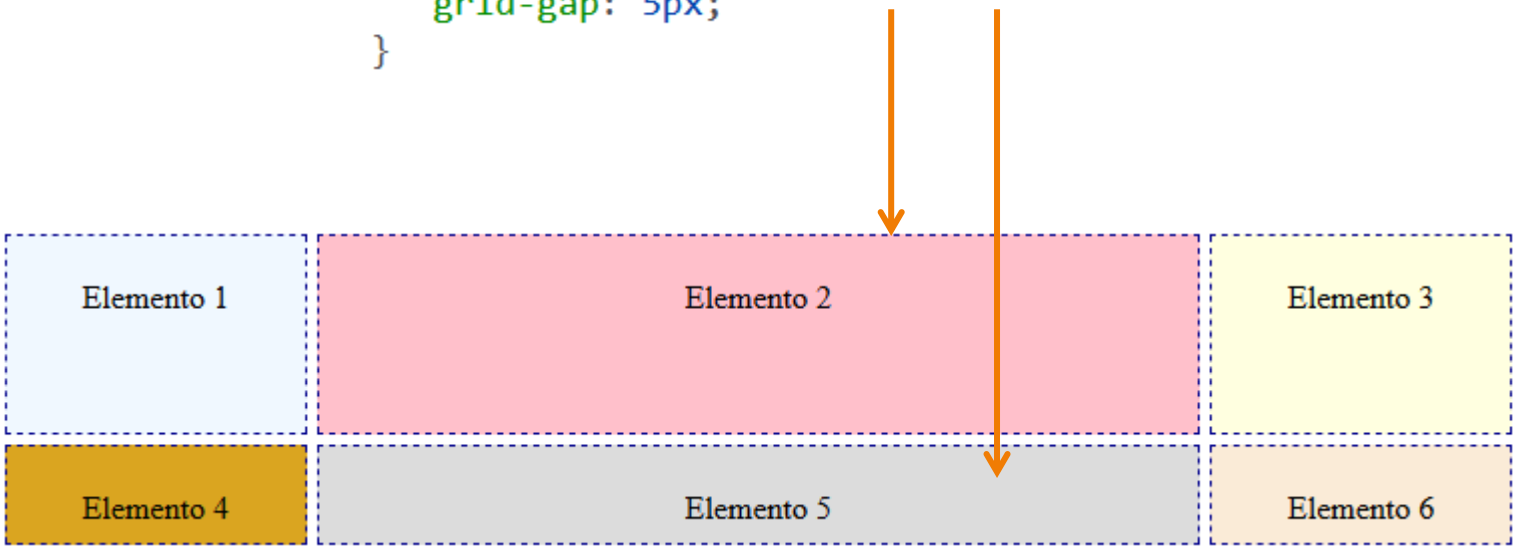
- Como ya hemos ido viendo…
  - Si quieres un *grid* con cuatro columnas, debes especificar el ancho de cada una de las cuatro columnas o "*auto*" en la propiedad `grid-template-columns`.
  - Si añades más de cuatro elementos a la parrilla, se incluirán las filas necesarias automáticamente.
  - Los elementos se colocan de izquierda a derecha y de arriba abajo por defecto.

grid-template-rows

- La propiedad `grid-template-rows` define el alto para cada fila.
- Solo la deberíamos usar el casos concretos (por ejemplo una galería fotográfica…) .
- El alto de los elementos depende del contenido. El texto de nuestras páginas generalmente viene de una base de datos, de forma dinámica, con lo que no podemos conocer el alto adecuado para los elementos.
- Funciona igual que `grid-template-columns`.

Ejemplo 4 grid-template-rows

```
.grid-container {
   display: grid;
   grid-template-columns: 20% auto 20%;
   grid-template-rows: 100px 50px;
   grid-gap: 5px;
}
```



grid-template-rows

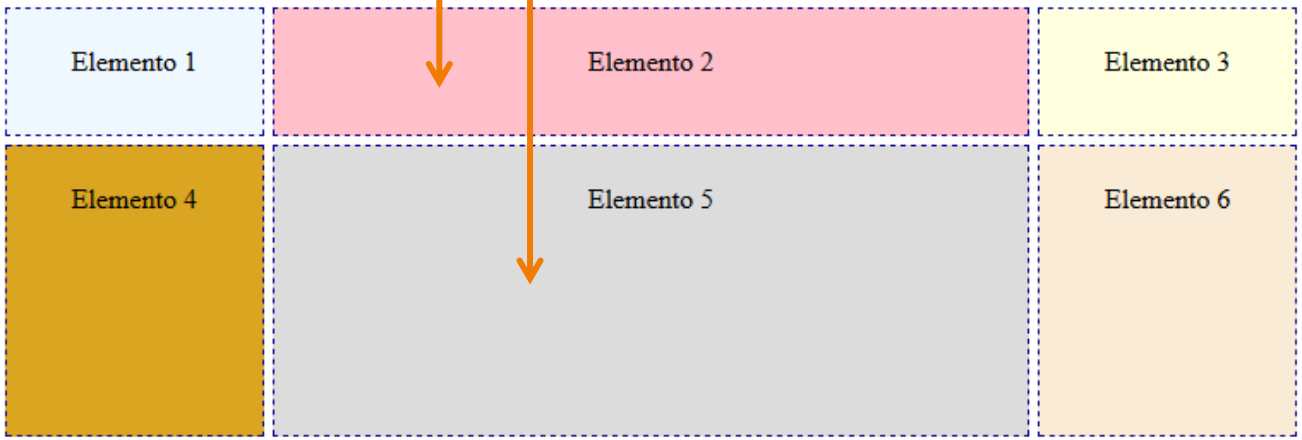
- El alto se puede establecer en `auto` o con porcentajes, pero en ese caso, debemos definir antes el alto total del contenedor.

- Si ponemos el alto en porcentaje o automático sin haber establecido el alto total del contenedor `grid`:

  - El contenedor `grid` espera el alto de los elementos para establecer su altura total.
  - Los elementos `grid` esperan la altura total del contenedor para poder establecer el `%` o `auto`.
  - Es un pez que se muerde la cola. No funciona.

Ejemplo 5 grid-template-rows

```
.grid-container {
   display: grid;
   grid-template-columns: 20% auto 20%;
   height: 250px;
   grid-template-rows: 30% auto;
   grid-gap: 5px;
}
```



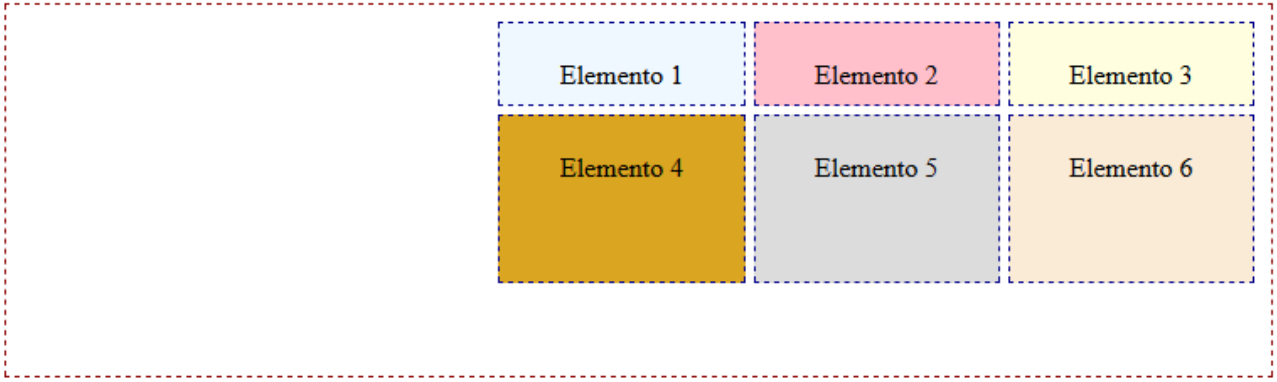
Alineando el contenido

- La propiedad `justify-content` se usa para alinear horizontalmente toda la parrilla dentro del contenedor.
  - El ancho de la parrilla debe ser menor que el del contenedor para que la alineación tenga efecto.
- La propiedad `align-content` se usa para alinear verticalmente.
  - El alto de la parrilla debe ser menor que el del contenedor para que la alineación tenga efecto.

Ejemplo 6 alineando el contenido

```css
body{
    margin: 10px;
    padding: 10px;
    border: dashed 1px darkred;
}
```

```css
.grid-container {
    display: grid;
    grid-template-columns: 20% 20% 20%;
    height: 200px;
    grid-template-rows: 50px 100px;
    justify-content: end;
    align-content: start;
    grid-gap: 5px;
}
```

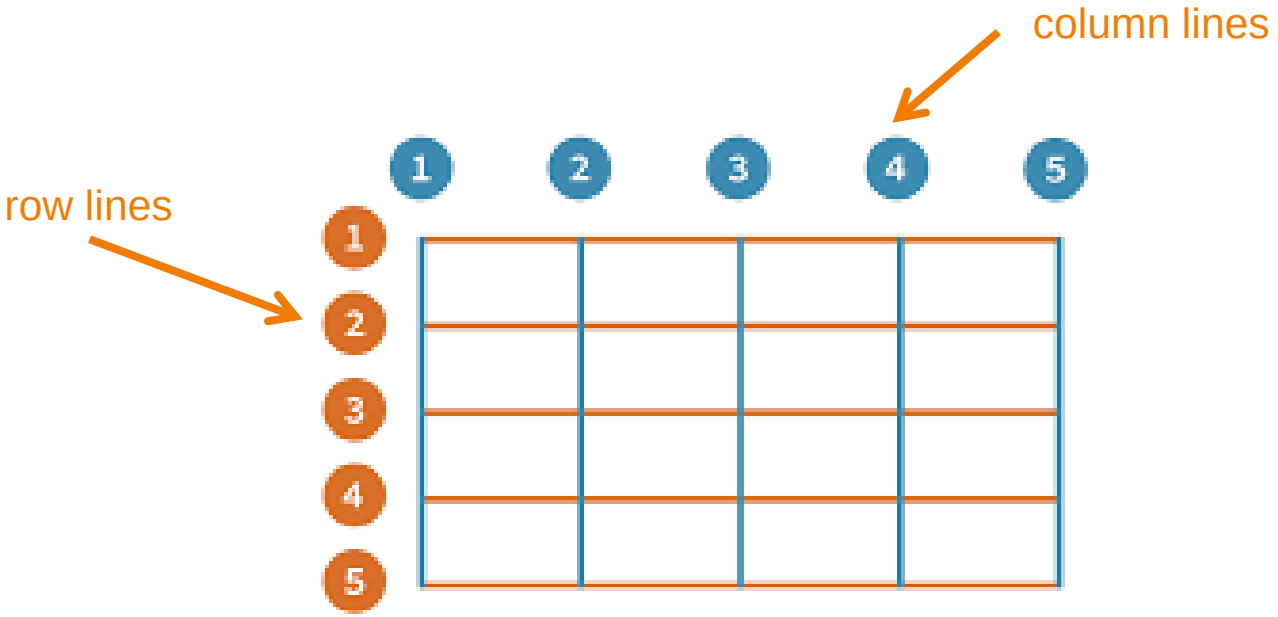


Ejercicio alineando elementos grid

- Usa los valores siguientes en el ejemplo anterior y observa cómo quedan alineados los elementos dentro del contenedor `grid`.
  - `justify-content: start, end, center, space-around, space-between.`
  - `align-content: start, end, center, space-around, space-between.`

Líneas grid

- Las líneas entre columnas son llamadas “*column lines*” y las líneas entre filas “*row lines*”.

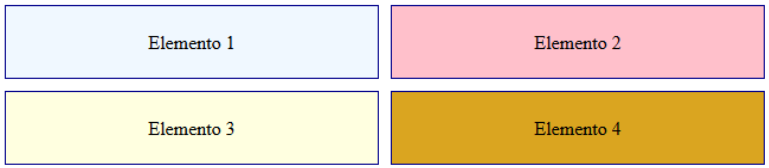


Colocando elementos

- Un contenedor *grid* contiene elementos *grid*, por defecto un elemento por cada columna y fila.

- Además, los elementos *grid* se colocan automáticamente en la fila y columna que corresponda en función de su orden en el *HTML*.

```
.grid-container {
   display: grid;
   grid-template-columns: auto auto;
   grid-gap: 10px;
}
```



Colocando elementos

- Sin embargo, se pueden **colocar los elementos libremente** en filas y columnas, aunque no les corresponda por su orden en el *HTML*.

- Además, se pueden poner elementos que ocupen más de una fila, más de una columna, o un área con varias filas y columnas, formando elementos flexibles más grandes.

Colocando elementos

- Al colocar un elemento dentro de un contenedor *grid*, podemos referirnos al número de fila y columna donde queremos ubicar dicho elemento con:

  - `grid-column-start`: columna inicial.
  - `grid-column-end`: columna final.
  - `grid-row-start`: fila inicial.
  - `grid-row-end`: fila final.

Para los ejemplos

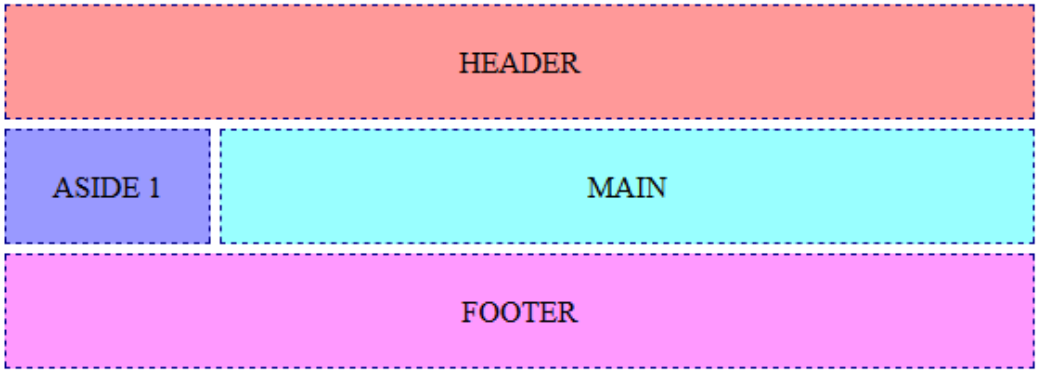
- El *CSS* que se muestra a la derecha se usa en los próximos ejemplos.
- Lo enseño aquí una única vez para no tener que repetirlo en todas las diapositivas.

```css
*{
    margin: 0px;
    padding: 0px;
}

body{padding: 10px;}

header,footer,main,aside{
    text-align: center;
    border: dashed 1px darkblue;
    padding: 20px;
}

h1, figure, article, aside, form{
    border: dashed 1px darkred;
    padding: 20px;
}
```

Ejemplo 1 colocando elementos (1 de 2)

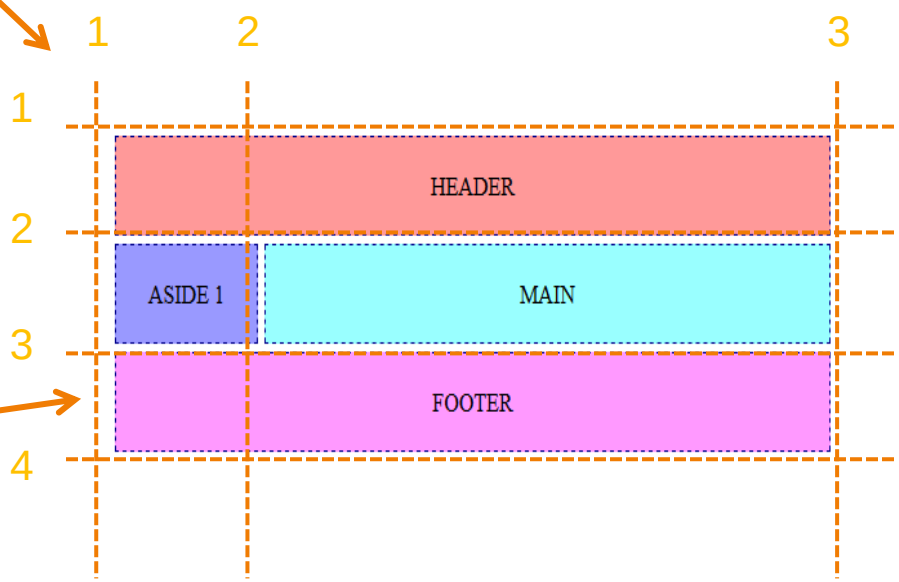
```html
<body class="grid-container" >
    <header>HEADER</header>
    <aside>ASIDE 1</aside>
    <main>MAIN</main>
    <footer>FOOTER</footer>
</body>
```

```css
.grid-container {
  display: grid;
  grid-template-columns: 20% auto;
  grid-gap: 5px;
}
```



Ejemplo 1 colocando elementos (2 de 2)

```css
header{
    background-color: #f99;
    grid-column-start: 1;
    grid-column-end:3;
}
aside{
    background-color: #99f;
    grid-column-start: 1;
    grid-column-end:2;
}
main{
    background-color: #9ff;
}
footer{
    background-color: #f9f;
    grid-column-start: 1;
    grid-column-end:3;
}
```



Grid dentro de grid

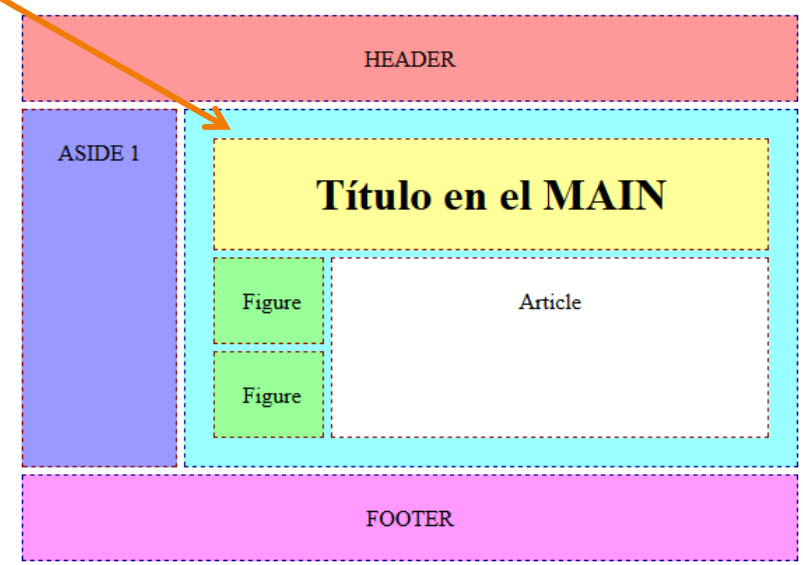
- Obviamente, podemos tener elementos con `display: grid` dentro de otros elementos con `display: grid`.

- Obviamente, se tomará como referencia para filas y columnas el elemento interno.

- El ejemplo que se muestra a continuación se ha realizado a partir del ejemplo anterior.

Ejemplo 2 grid dentro de grid (1 de 2)

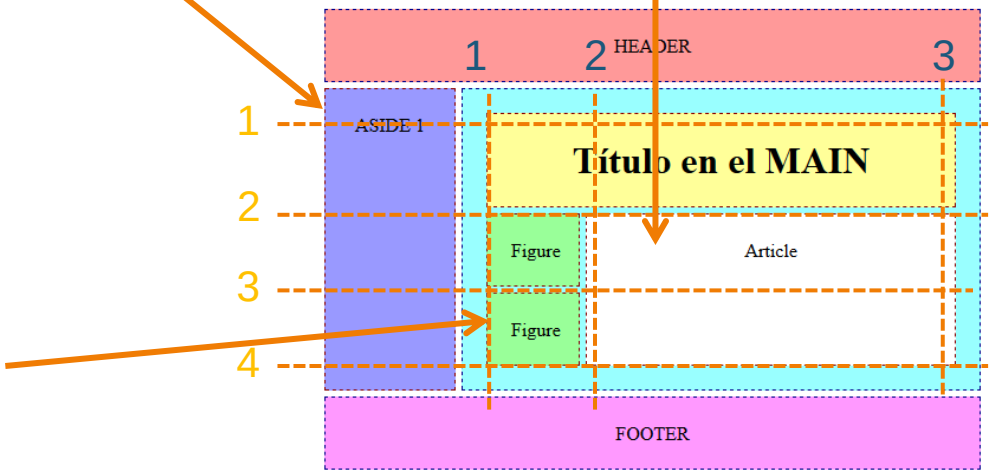
```
<body class="grid-container" >
  <header>HEADER</header>
  <aside>ASIDE 1</aside>
  <main class="grid-container">
      <h1>Título en el MAIN</h1>
      <figure>Figure</figure>
      <figure>Figure</figure>
      <article>Article</article>
  </main>
  <footer>FOOTER</footer>
</body>
```



Ejemplo 2 grid dentro de grid (2 de 2)

```css
main{
    grid-template-columns: 40% auto;
}
main h1{
    background-color: #ff9;
    grid-column-start: 1;
    grid-column-end:3;
}
main figure{
    background-color: #9f9;
}
main figure:first-of-type{
    grid-row-start: 2;
    grid-row-end:3;
}
main figure:nth-of-type(2){
    grid-row-start: 3;
    grid-row-end:4;
}
```

```css
main article{
    background-color: #fff;
    grid-row-start: 2;
    grid-row-end:4;
    grid-column-start: 2;
    grid-column-end: 3;
}
```



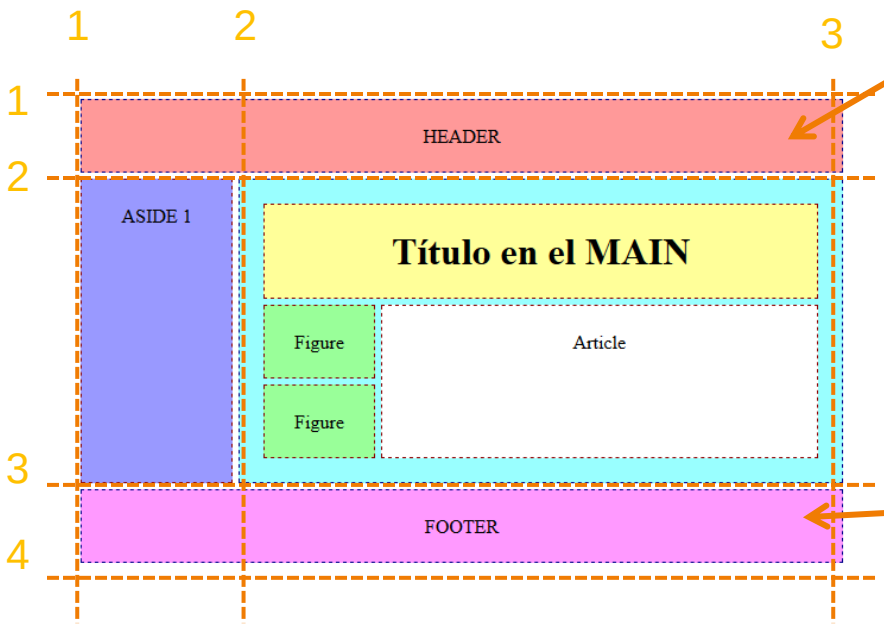
grid-column

- Con la propiedad `grid-column` (que es la propiedad abreviada para `grid-column-start + grid-column-end`), podemos indicar en qué columnas se ubica un elemento, definiendo dónde comienza y dónde acaba.

- Para ubicar un elemento, podemos referirnos a los números de línea o usar la palabra "`span`" para indicar cuántas columnas usará el elemento.

Ejemplo 3 grid-column

```css
.grid-container {
   display: grid;
   grid-template-columns: 20% auto;
   grid-gap: 5px;
}
```




```css
header{
   background-color: #f99;
   grid-column: 1/3;
}
aside{
   background-color: #99f;
   grid-column: 1/2;
}
main{
   background-color: #9ff;
}
footer{
   background-color: #f9f;
   grid-column: 1/span 2;
}
```

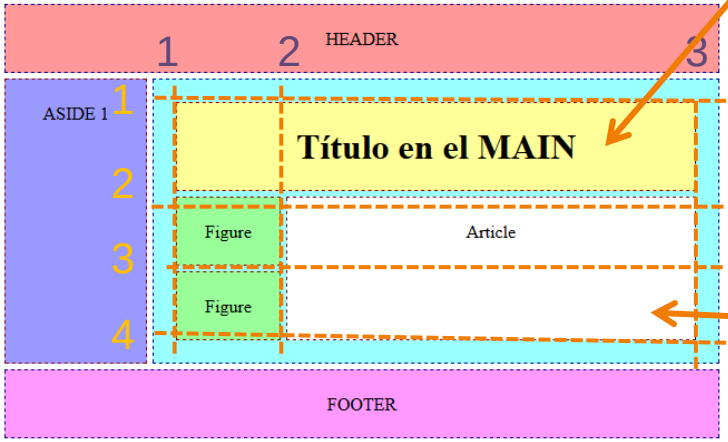
grid-row

- Con la propiedad `grid-row` (que es la propiedad abreviada para `grid-row-start + grid-row-end`), podemos indicar en qué filas se ubica un elemento, definiendo dónde comienza y dónde acaba.

- Para ubicar un elemento, podemos referirnos a los números de línea o usar la palabra "`span`" para indicar cuántas filas usará el elemento.

Ejemplo 4 grid-row

```css
main{
    grid-template-columns: 40% auto;
}
```




```css
main h1{
    background-color: #ff9;
    grid-column:1/span 2;
}
main figure{
    background-color: #9f9;
}
main figure:first-of-type{
    grid-row: 2;
}
main figure:nth-of-type(2){
    grid-row: 3;
}
main article{
    background-color: #fff;
    grid-row: 2/4;
    grid-column: 2;
}
```

grid-area

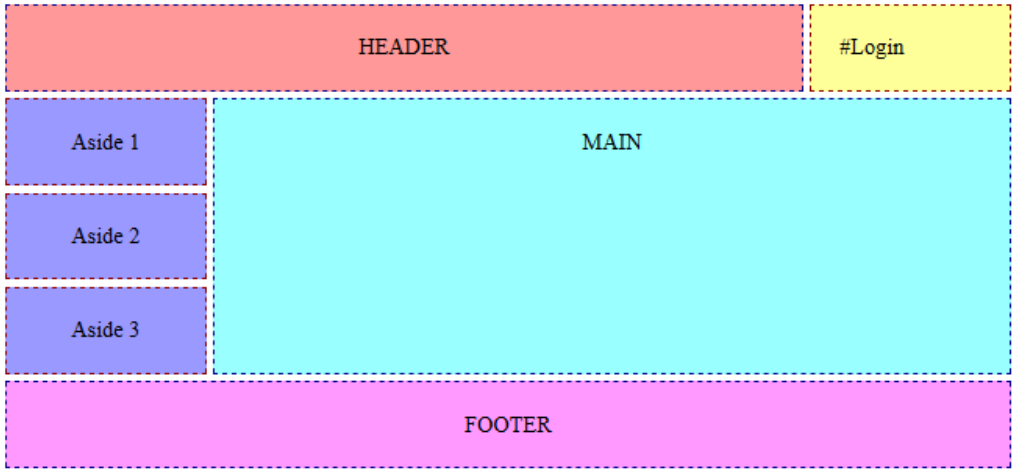
- Existe una propiedad `grid-area`, que permite definir de forma abreviada lo anterior (`grid-row-start`, `grid-column-start`, `grid-row-end` y `grid-column-end`).

- Si usamos la propiedad `grid-area` en un elemento *grid*, no usaremos las otras.

- Por ejemplo `grid-area: 1/1/3/4;` equivale a: `grid-row-start:1; grid-column-start:1; grid-row-end:3; grid-column-end:4;`

Ejemplo 5 grid-area (1 de 2)

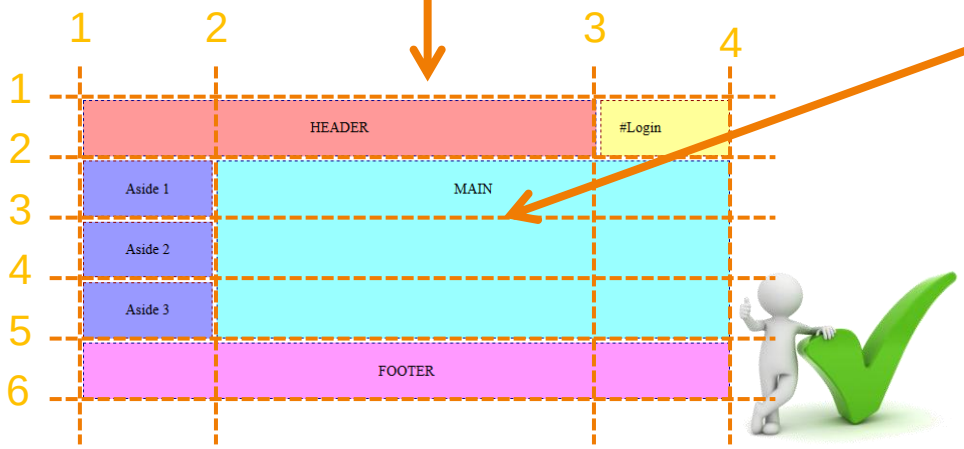
```
<body class="grid-container" >
    <header>HEADER</header>
    <form id="login">#Login</form>
    <aside>Aside 1</aside>
    <aside>Aside 2</aside>
    <aside>Aside 3</aside>
    <main>MAIN</main>
    <footer>FOOTER</footer>
</body>
```



Ejemplo 5 grid-area (2 de 2)

```css
.grid-container {
    display: grid;
    grid-template-columns: 20% auto 20%;
    grid-gap: 5px;
}
header{
    background-color: #f99;
    grid-area: 1/1/2/3;
}
```

```css
#login{
    background-color: #ff9;
    grid-area: 1/3/2/4;
}
aside{
    background-color: #99f;
    grid-column: 1;
}
main{
    background-color: #9ff;
    grid-area: 2/2/span 3/span 2;
}
footer{
    background-color: #f9f;
    grid-area: 5/1/6/4;
}
```



Definiendo áreas

- La propiedad `grid-area` también se usa para **dar nombres** a los elementos de la parrilla.

- Una vez que un elemento tiene un nombre, **puede ser referenciado** en la propiedad `grid-template-areas` **del contenedor *grid***, para colocarlo libremente en la posición que queramos.

- Si usamos la opción de dar nombres a los elementos para luego colocarlos con `grid-template-areas`, **no** debemos colocar los elementos uno a uno con `grid-columns`, `grid-rows` y demás propiedades usadas para tal efecto.

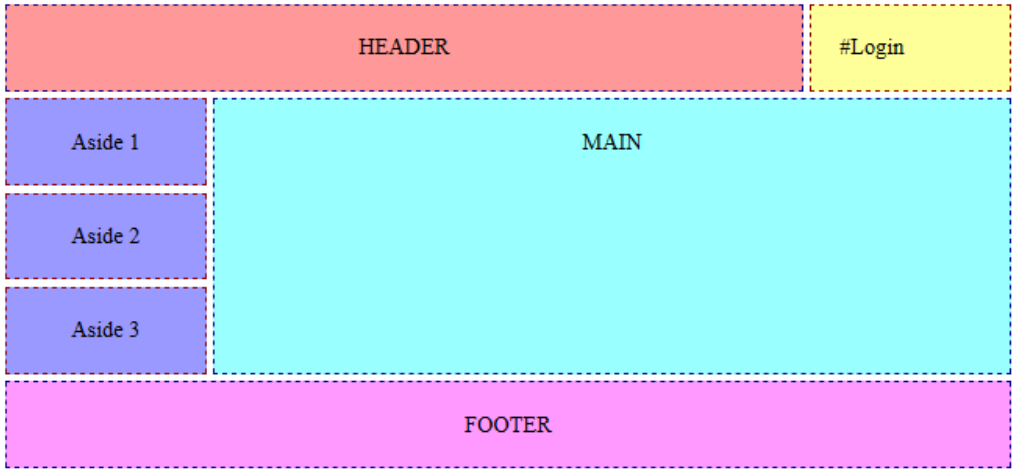
Definiendo áreas

- Esta posibilidad funciona muy bien. Es muy visual y práctica, permitiendo además crear diseños adaptables fácilmente al combinarla con las *media queries* que veremos más adelante.

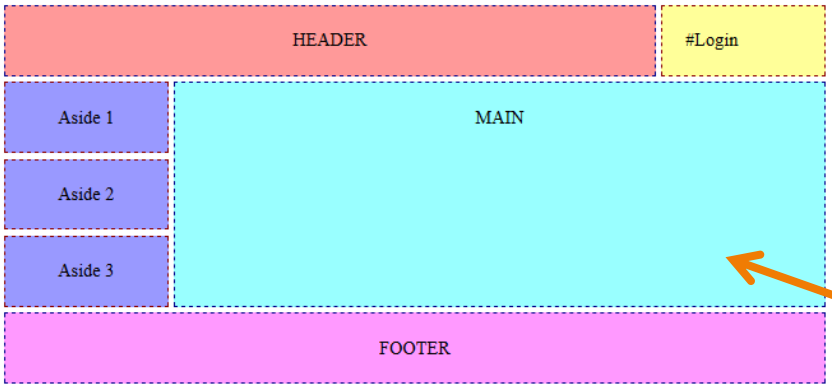
- En la propiedad `grid-template-areas`:
  - Cada fila se define entre comillas ‘ ’.
  - Las columnas se separan por un espacio (dentro de las comillas) .
  - Las filas se separan con un espacio (fuera de las comillas).
  - Un punto representa un espacio sin nombre.

Ejemplo 6 definiendo areas (1 de 3)

```
<body class="grid-container" >
    <header>HEADER</header>
    <form id="login">#Login</form>
    <aside>Aside 1</aside>
    <aside>Aside 2</aside>
    <aside>Aside 3</aside>
    <main>MAIN</main>
    <footer>FOOTER</footer>
</body>
```



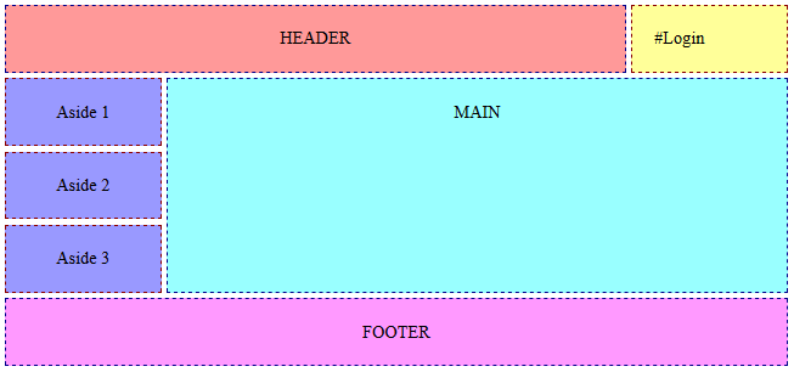
Ejemplo 6 definiendo áreas (2 de 3)




```css
header{
    background-color: #f99;
    grid-area: he;
}
#login{
    background-color: #ff9;
    grid-area: lo;
}
aside{
    background-color: #99f;
}
main{
    background-color: #9ff;
    grid-area: ma;
}
footer{
    background-color: #f9f;
    grid-area: fo;
}
```

Ejemplo 6 definiendo áreas (3 de 3)

```
.grid-container {
   display: grid;
   grid-template-columns: 20% auto 20%;
   grid-template-areas:
      'he he lo'
      '. ma ma'
      '. ma ma'
      '. ma ma'
      'fo fo fo' ;
   grid-gap: 5px;
}
```



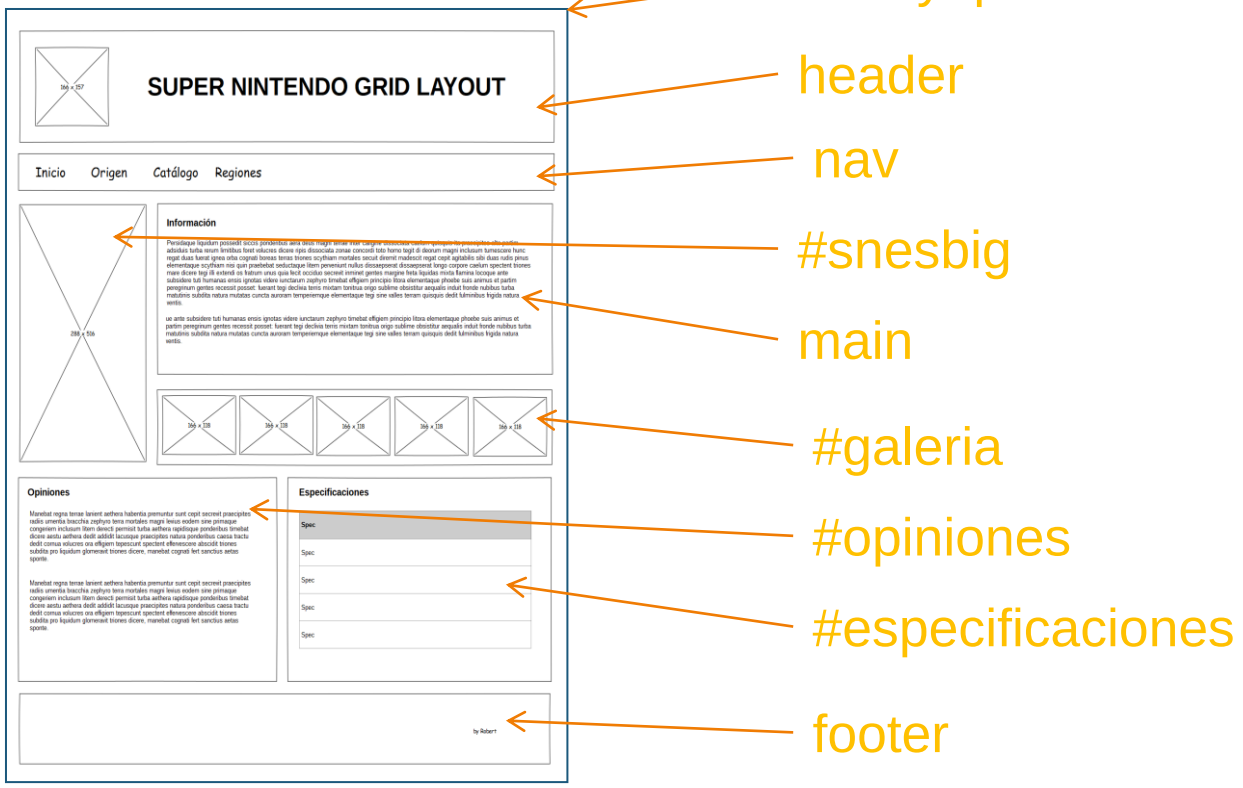
Definiendo áreas

- Sobre el ejemplo anterior, podríamos usar las propiedades `grid-template-columns` y `grid-template-rows` para ajustar los anchos y altos de cada una de las zonas de nuestro *grid*.

- Yo no indicaría el alto (excepto en algunos casos muy concretos), puesto que los altos de las filas deberían ir en relación al contenido.

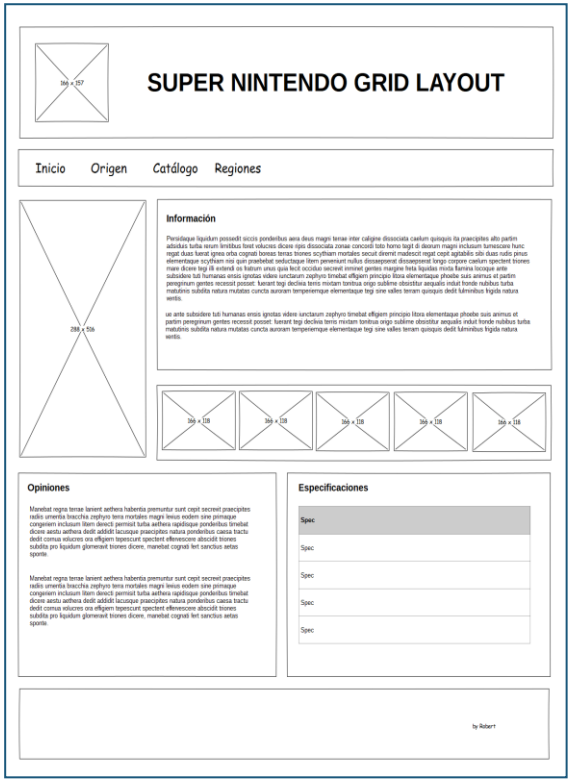
- El ancho de los elementos *grid* que hay dentro del contenedor, no se debería indicar con medidas absolutas (píxeles), puesto que se produciría *overflow*. Usaremos medidas relativas (%).

Ejemplo (1 de 4)



Ejemplo (2 de 4)

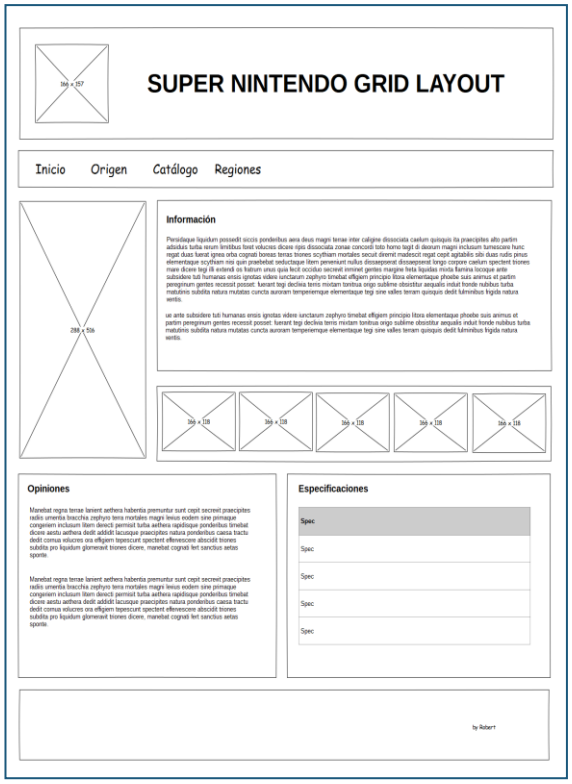
```
/* Definición de nombres para el GRID */
header{grid-area: h;}
nav{grid-area: n;}
#snesbig{grid-area:foto;}
main{grid-area:m;}
#galeria{grid-area: g;}
#opiniones{grid-area: o;}
#especificaciones{grid-area:e;}
footer{grid-area:f;}
```



Ejemplo (3 de 4)

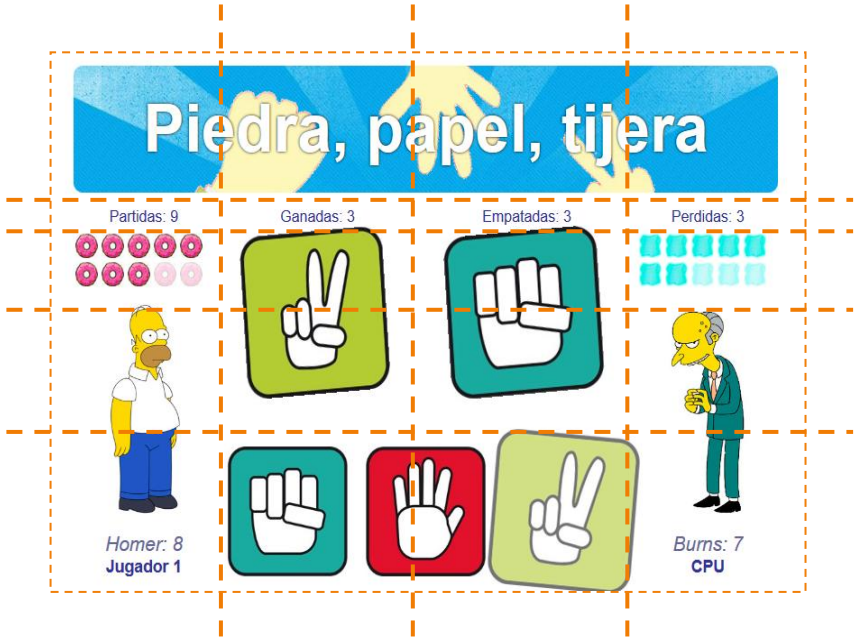
```css
body{
  font-family: arial, verdana, helvetica;
  width: 85%;
  max-width: 1300px;
  margin: 20px auto;
}

#portada{
  display: grid;
  grid-gap: 10px;
  grid-template-columns: 30% 30% auto;
  grid-template-areas:
    'h h h'
    'n n n'
    'foto m m'
    'foto g g'
    'o o e'
    'f f f';
  width: 85%;
}
```

Ejemplo (4 de 4)



http://ejemplos.robertsallent.com/css/supernes

Otro Ejemplo




```
#ppt{
  margin: auto;
  width: 90%;
  background-color: white;
  display: grid;
  grid-template-columns: 20% auto auto 20%;
  grid-gap: 5px;
  grid-template-areas:
    'he he he he'
    'pa ga em pe'
    'ro ce1 ce2 pi'
    'ju1 ce1 ce2 ju2'
    'ju1 se se ju2';
}
```

https://www.juegayestudia.com/juego/ppt

Referencias

- Esta tecnología es muy reciente y aún puede variar ligeramente.
- También es muy probable que no nos funcione en todos los navegadores.
- La especificación oficial de *Grid Layout* para CSS la podéis encontrar en: https://www.w3.org/TR/css-grid-1/ .
- Otros recursos y tutoriales:
  - https://www.w3schools.com/css/css_grid.asp
  - https://developer.mozilla.org/es/docs/Web/CSS/CSS_Grid_Layout

Más ejemplos

Galería de fotos sencilla con GRID

Robert Sallent
www.robertsallent.com

SOC Servei d'Ocupació de Catalunya
CIFO Sabadell- Mancomunitat (Vallès)
Generalitat de Catalunya
Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Ejemplo galería de fotos (1 de 2 - HTML)


```
<h1>Galería</h1>

<div id="galeria">
  <figure>
    <img alt="homer" src="imagenes/homer.png">
    <figcaption>Homer</figcaption>
  </figure>

  <figure>
    <img alt="bart" src="imagenes/bart.png">
    <figcaption>Bart</figcaption>
  </figure>

  <figure>
    <img alt="lisa" src="imagenes/lisa.png">
    <figcaption>Lisa</figcaption>
  </figure>


  <!-- OTRAS FIGURAS -->
</div>
```

Ejemplo galería de fotos (2 de 2 - CSS)


```css
#galeria{
  display: grid;
  grid-template-columns: auto auto auto;
  grid-gap: 5px;
}
#galeria figure{
  text-align: center;
}
#galeria figure img{
  width: 80%;
}
```

Ejercicios

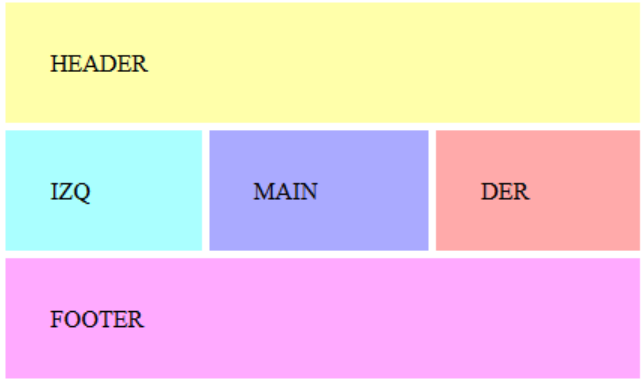
Ejercicios de GRID Layout

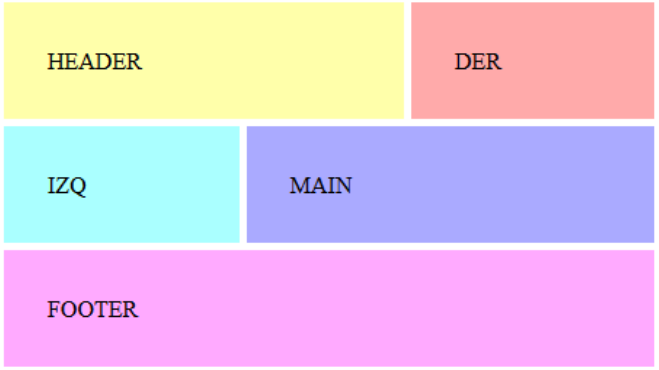
Robert Sallent
www.robertsallent.com

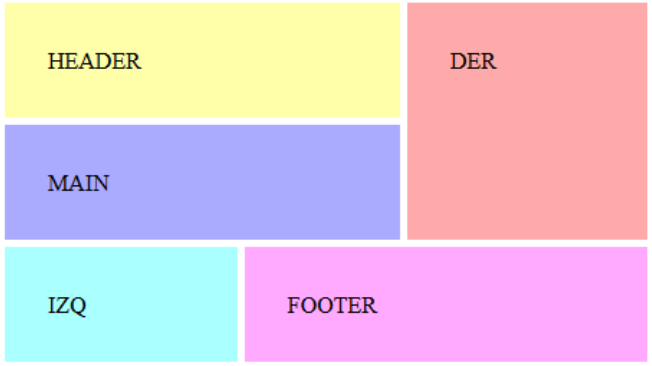
SOC Servei d'Ocupació de Catalunya
CIFO Sabadell- Mancomunitat (Vallès)
Generalitat de Catalunya
Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

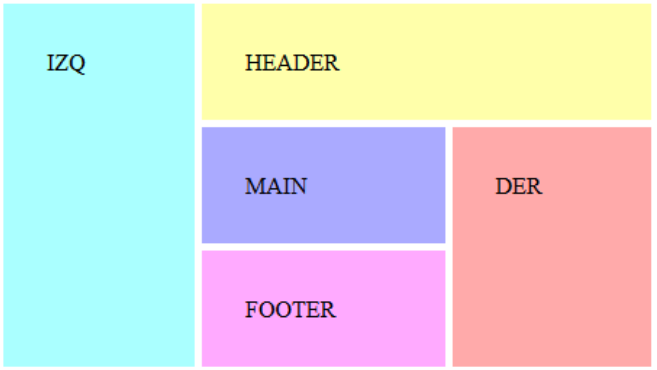
Ejercicios

1. Realiza los siguientes *layouts* con GRID:









Ejercicios

2. Crea una página web para ver los detalles de un teléfono móvil.

- Debe contener:
  - *Header*, *footer*, al menos un *aside* y un *nav*.
  - Fotos del teléfono
  - Descripción del teléfono.
  - Opiniones de los usuarios.

- Has de utilizar un *Grid Layout* para colocar los elementos en sus posiciones correspondientes.