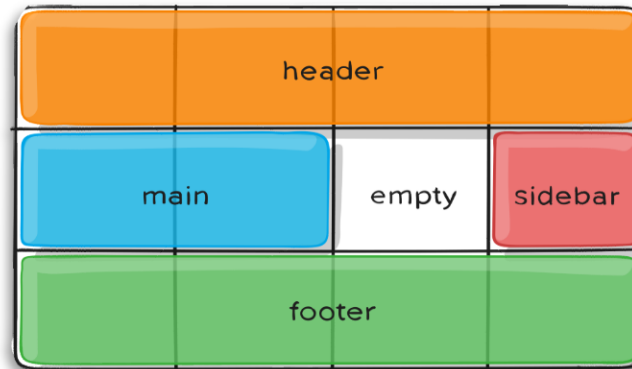


Diseño de Interfaces Web

CSS Grid



CSS Grid

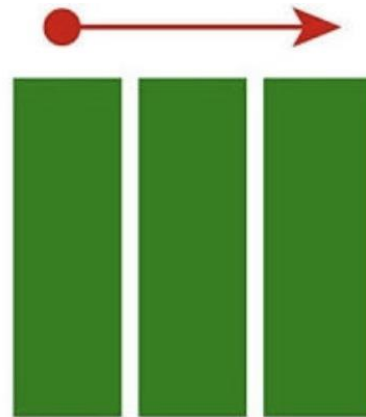


1. [Introducción](#)
2. [Terminología grid](#)
3. [Cuadrícula grid](#)
4. [Alineación](#)
5. [Grid Auto-fill / Grid Auto-fit](#)
6. [Orden de los elementos](#)
7. [Colocación implícita](#)
8. [Áreas grid](#)
9. [Anidación](#)
- * [Referencias](#)

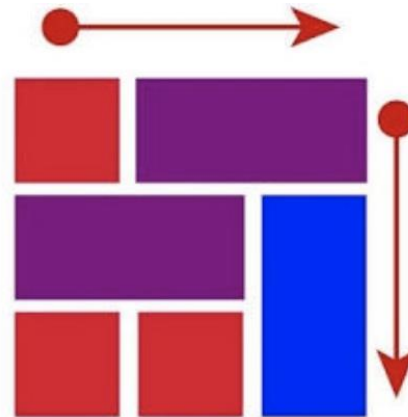


1. Introducción

- ¿Qué es CSS Grid Layout?
 - Módulo de CSS3 que nos va a permitir maquetar en **dos dimensiones**, teniendo en cuenta las filas y las columnas
 - Conseguiremos desarrollar estructuras complejas con una sintaxis sencilla



Flexbox
One Dimensions

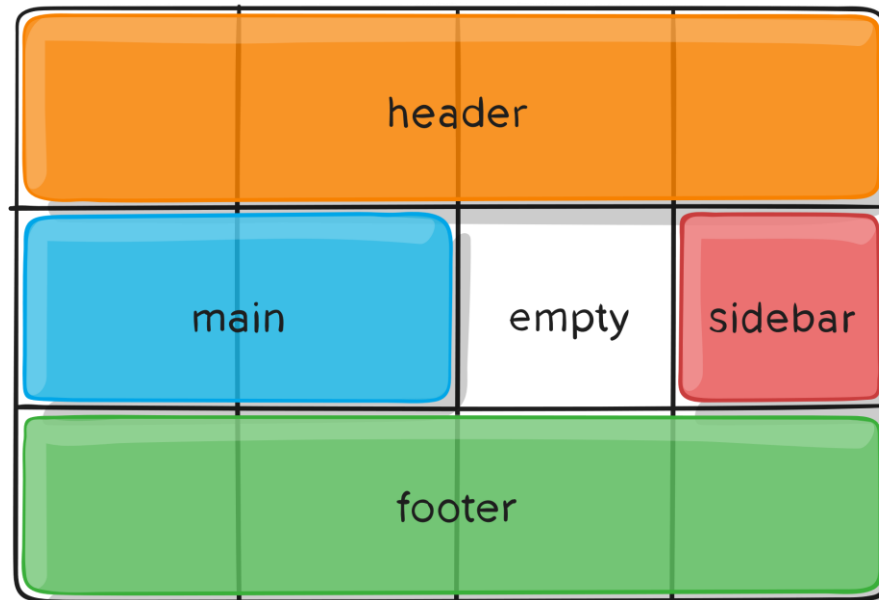


CSS Grids
Two Dimensions

1. Introducción



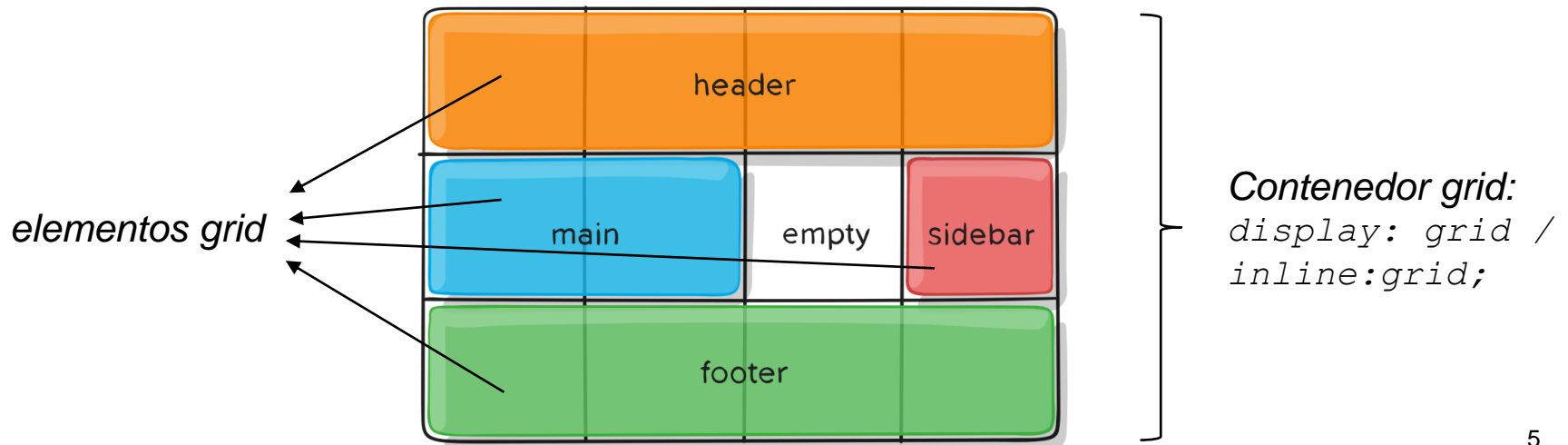
- ¿Qué es CSS Grid Layout?
 - Nuestro trabajo consistirá en definir una cuadrícula base y después indicar qué celdas de esa cuadrícula ocuparán los elementos que queremos distribuir





1. Introducción

- Grid container y grid items
 - Al igual que en Flexbox, distinguiremos entre las propiedades que se aplican al contenedor padre (**grid container**) y las que se aplican a sus hijos directos (**grid items**)
 - Al elemento contenedor aplicamos **display: grid;** o **display: inline-grid;**

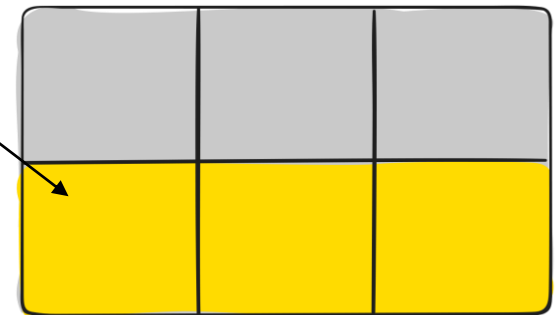
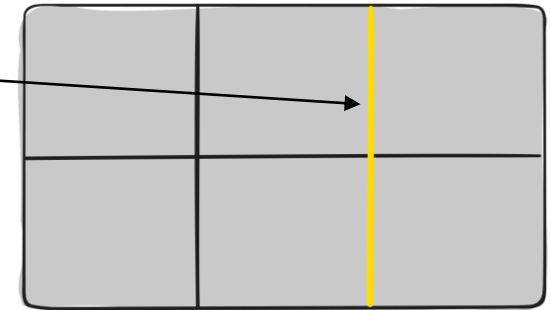
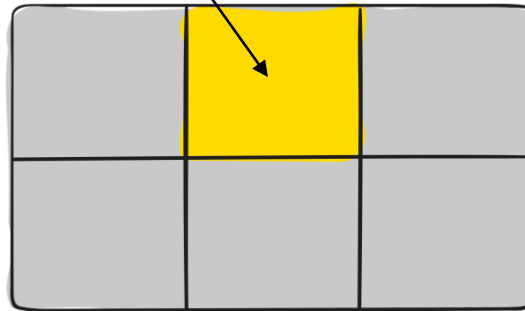
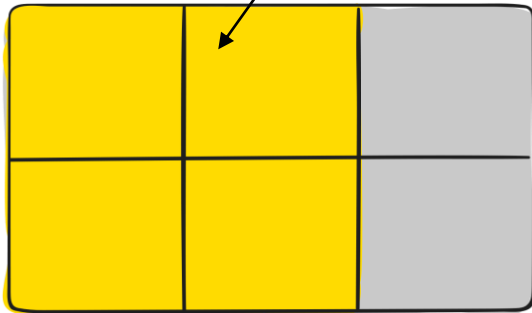


2. Terminología grid



- Términos Grid

- Grid container / Grid items
- Grid lines (líneas)
- Grid tracks (vías [filas o columnas])
- Grid cells (celdas)
- Grid areas (áreas)



3. Cuadrícula grid



- Definición de la cuadrícula
 - Propiedades para grid container
 - **grid-template-columns**: define el número y tamaño de las columnas del grid;
 - **grid-template-rows**: define el número y tamaño de las filas;
 - El tamaño se especifica en *porcentajes*, *píxeles*, *fracciones (fr)*, ...
 - Otros valores: *auto*, *min-content*, *max-content*
 - Las líneas se pueden etiquetar mediante [] para referenciarlas
 - Shorthand:
 - **grid**: filas / columnas;

3. Cuadrícula grid



- Definición de la cuadrícula
 - Propiedades para grid container
 - Ejemplos

✓ `grid-template-columns: 25% 25% 25% 25%;`

✓ `grid-template-rows: 50% 50%;`

✓ `grid-template-columns: 100px 100px auto 200px;`

✓ `grid-template-rows: [uno] 33.33% [dos] 33.33%
[tres] 33.33% [cuatro];`

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

✓ `grid-template-rows: 3fr 1fr;`

✓ `grid: 3fr 1fr / 1fr 1fr 1fr;`

3. Cuadrícula grid



- Definición de la cuadrícula

- Separación entre vías

- **row-gap** y **column-gap**: separación entre filas y columnas
 - **gap**: valor_row_gap / valor_columna_gap

- Ejemplos

✓ row-gap: 1rem;	}	gap: 1rem 3rem;
✓ column-gap: 1rem;		



3. Cuadrícula grid

- Definición de la cuadrícula
 - Propiedades para grid container
 - Repetición de valores: **repeat (X, valor);**
 - Ejemplos
 - ✓ `grid-template-columns: repeat(4, 25%);`
 - ✓ `grid-template-rows: repeat(2, 50%);`
 - ✓ `grid-template-columns: 50% repeat(2, 25%);`
 - ✓ `grid-template-columns: repeat(3, 1fr);`
 - ✓ `grid: repeat (2, 50%) / repeat(3, 33,3%);`
 - ✓ `grid: repeat (2, 50%) / 50% repeat(2, 25%);`

3. Cuadrícula grid

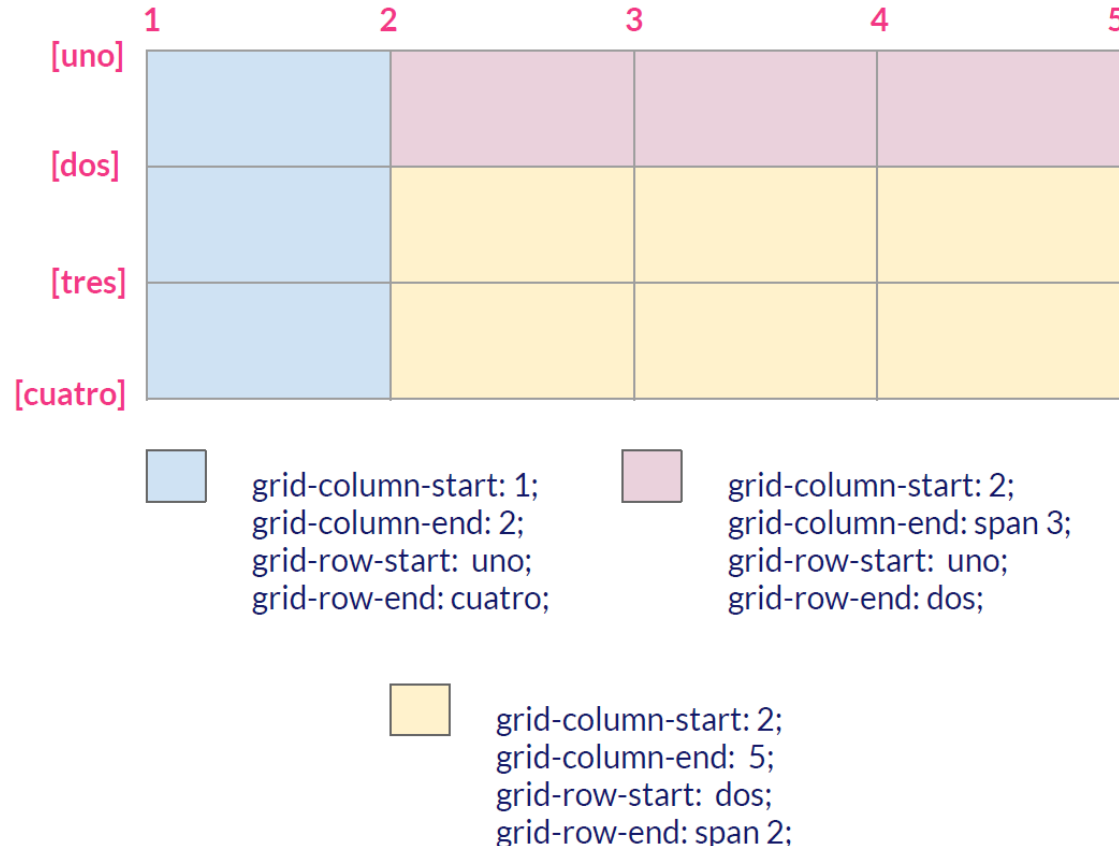


- Definición de las áreas (celdas que se ocuparán)
 - Propiedades para grid items
 - Indicamos los **números de líneas** de inicio y de fin o **etiquetas**
 - **grid-column-start:** X
 - **grid-column-end:** X
 - **grid-row-start:** X
 - **grid-row-end:** X
 - **grid-column:** X / X
 - **grid-row:** X / X
 - **grid-area:** row_start / column_start / row-end / column_end

3. Cuadrícula grid



- Definición de las áreas (celdas que se ocuparán)
 - Ejemplo



4. Alineación



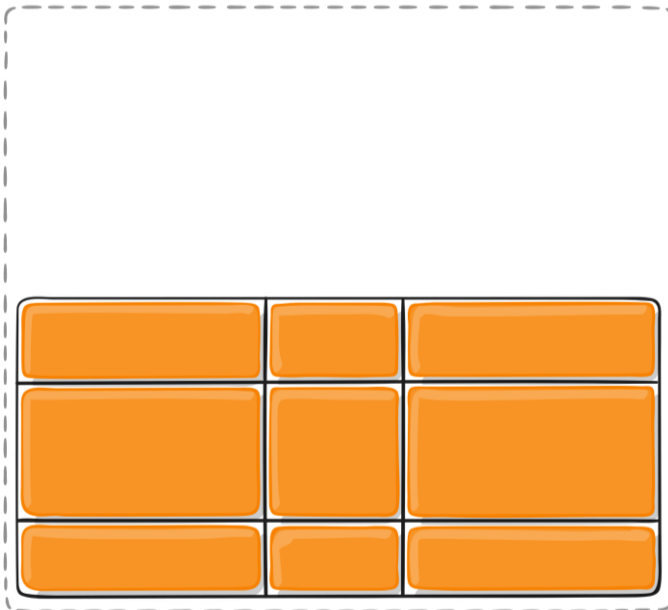
- Alineación elementos grid
 - En el caso de que el contenedor grid no ocupe todo el ancho de la página o su contenedor principal, podemos indicarle dónde se tiene que colocar dentro del espacio disponible
 - Propiedades para **grid container**
 - Alineación **horizontal**
 - **justify-content**: *start / center / end / space-between / space-around / space-evenly*
 - Alineación **vertical**
 - **align-content**: *start / center / end / stretch / initial*
 - Ambas alineaciones
 - **place-content**: *valor_vertical valor_horizontal*

4. Alineación

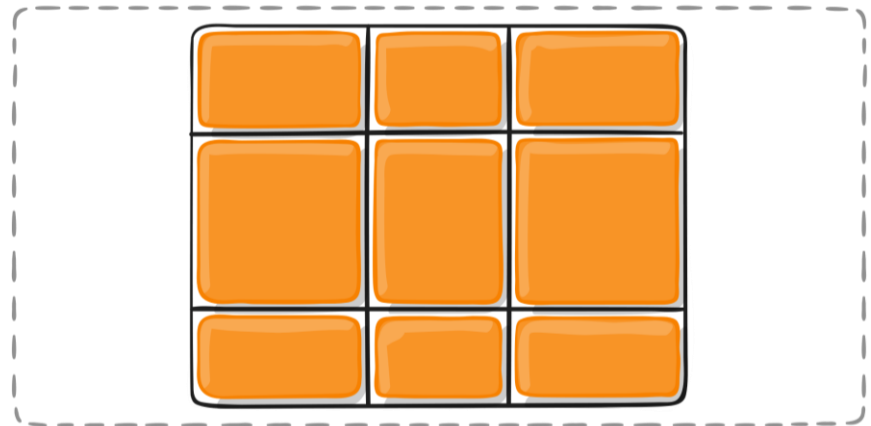


- Alineación elementos grid
 - Ejemplos

`align-content: end;`



`justify-content: center;`





4. Alineación









- Alineación del contenido de las celdas
 - Propiedades grid container
 - Alineación **horizontal**
 - **justify-items**: *start / center / end / stretch* (por defecto)
 - Alineación **vertical**
 - **align-items**: *start / center / end / stretch* (por defecto)
 - Ambas alineaciones
 - **place-item**: *valor_vertical valor_horizontal*
 - Alineación de elementos concretos
 - **justify-self**
 - **align-self**

4. Alineación



- Alineación del contenido de las celdas
 - Ejemplos

`align-items: end;`

`justify-items: center;`

5. Grid Auto-fill / Grid Auto-fit



- Grid Auto-fill / Grid Auto-fit
 - Hacen que los elementos grid se coloquen de forma automática en la cuadrícula según un tamaño establecido
 - **Grid Auto-fill**
 - ✓ `grid-template-columns: repeat(auto-fill, 200px);`
 - Se mejora con **minmax** (indica lo que los elementos deben medir como mínimo y como máximo)
 - ✓ `grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));`
 - **Grid auto-fit** (además rellena el espacio en blanco sobrante)
 - ✓ `grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));`

6. Orden de los elementos



- Order
 - La propiedad **order** permite cambiar el orden en el que aparecen los elementos en el espacio grid
 - Propiedad equivalente al *order* de Flexbox
 - Por defecto todos los elementos tienen **order: 0**;
 - Un valor negativo hará que el elemento aparezca antes que el resto y los valores positivos lo harán después



7. Colocación implícita

- Colocación implícita
 - ¿Qué sucede cuando colocamos elementos grid fuera de la estructura definida en el contenedor grid (estructura explícita)?
 - En este caso, el espacio grid hace una prolongación de las columnas o filas (crea una retícula implícita)
 - Propiedades:
 - ✓ **grid-auto-columns:** X
Especificamos el tamaño que deberían tener las columnas adicionales que se creen en la retícula implícita
 - ✓ **grid-auto-rows:** X
Especificamos el tamaño que deberían tener las filas adicionales que se creen en la retícula implícita



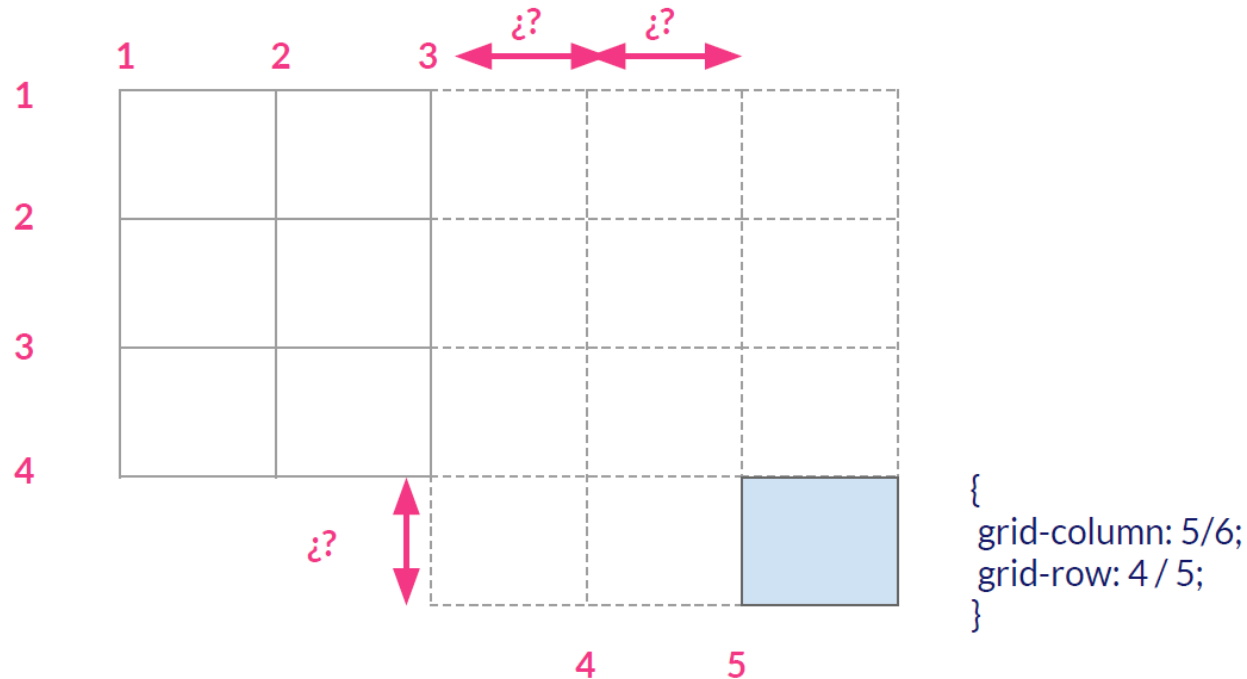
7. Colocación implícita

- Colocación implícita
 - ¿Qué sucede cuando a los elementos no se les indica una posición específica en la retícula grid?
 - Se puede usar la propiedad **grid-auto-flow**
 - ✓ **grid-auto-flow:** *row / column / dense*
 - *row*: rellena primero las filas (por defecto) y añade nuevas filas si es necesario
 - *column*: rellena primero las columnas y añade nuevas si es necesario
 - *dense*: mediante el algoritmo propio de CSS Grid intenta rellenar todos los huecos disponibles para que quede una retícula compacta



7. Colocación implícita

- Colocación implícita



```
{  
  grid-column: 5/6;  
  grid-row: 4 / 5;  
}
```

```
.container{  
  grid-auto-columns: 50px;  
  grid-auto-rows: 50px;  
}
```

} (por defecto 0)

8. Áreas grid

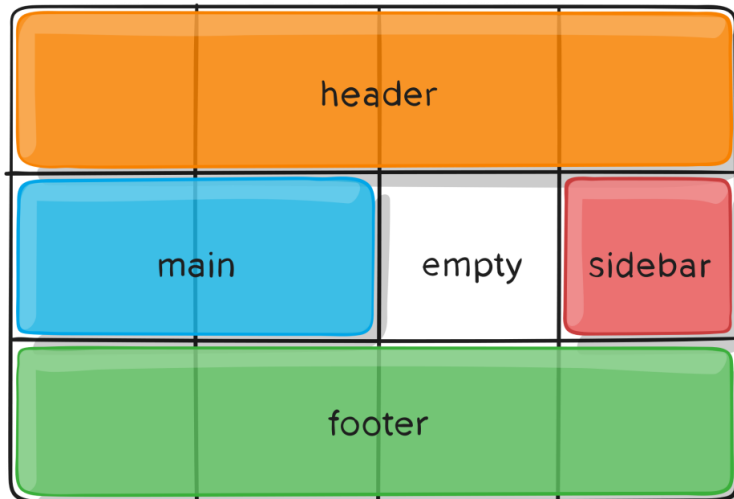


- Grid Template Areas
 - Otra forma de definir la cuadrícula y colocar los elementos es creando un **área grid**
 - En el contenedor le damos un nombre común a todas las celdas que forman un área a modo de plantilla en **grid-template-area**
 - A los elementos grid le indicamos qué área ocupan con **grid-área: nombre_área**

8. Áreas grid



- Grid Template Areas
 - Ejemplo

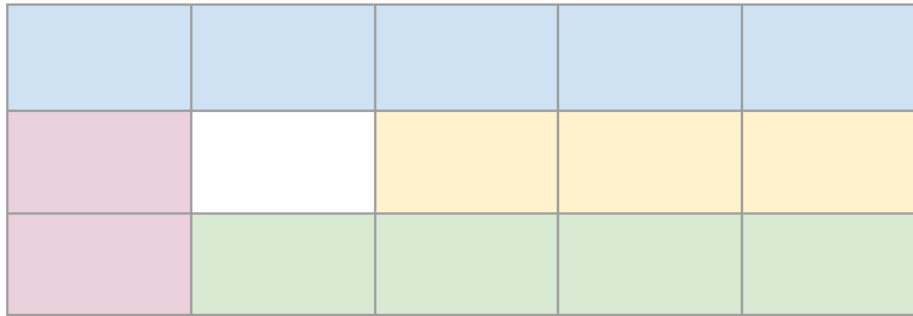


```
.item-a {  
  grid-area: header;  
}  
.item-b {  
  grid-area: main;  
}  
.item-c {  
  grid-area: sidebar;  
}  
.item-d {  
  grid-area: footer;  
}  
  
.container {  
  display: grid;  
  grid-template-columns: 50px 50px 50px 50px;  
  grid-template-rows: auto;  
  grid-template-areas:  
    "header header header header"  
    "main main . sidebar"  
    "footer footer footer footer";  
}
```

8. Áreas grid



- Grid Template Areas
 - 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";  
}
```

Hueco



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



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



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

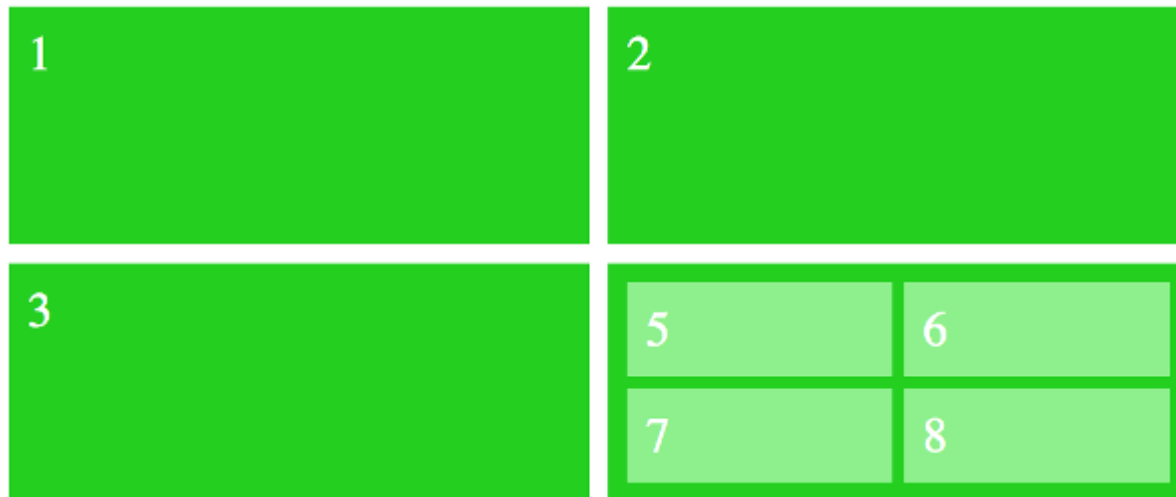


```
{  
  grid-area: menu;  
}
```




9. Anidación

- Anidación en grid (nested grid)
 - Para diseños más complejos, podemos anidar nuevas estructuras grid dentro de un elemento grid
 - Aplicamos **display: grid;** a un elemento grid
 - Valor **subgrid** (por implementar)





*. Referencias

- Bibliografía y referencias
 - Curso Domestika “Layout web con CSS Grid y Flexbox y otras técnicas modernas”, de Javier Usobiaga Ferrer
 - Curso Openwebinars “Maquetación con Flexbox y Grid”, de Juan Diego Pérez
 - Curso Udemy “CSS Grid y Flexbox, la guía definitiva”, de Juan Pablo de la Torre
 - A complete guide to CSS Grid: <https://css-tricks.com/snippets/css/complete-guide-grid/>
 - MDN web docs: <https://developer.mozilla.org/es/>