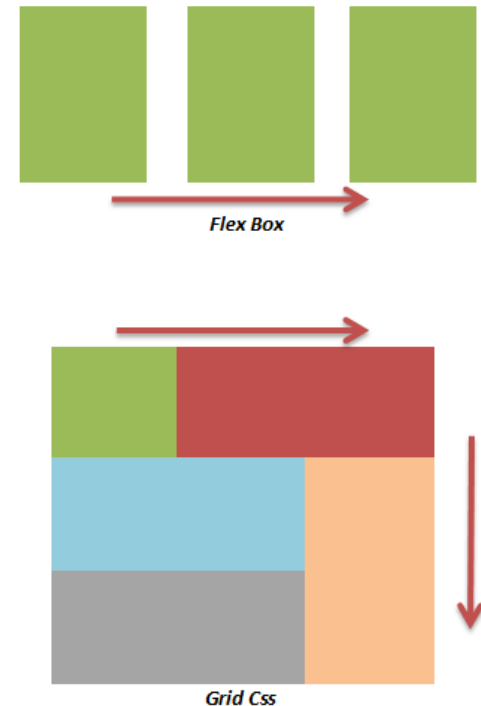


# TEMA 3

CSS GRID

# 1 CSS GRID

- ▶ Permite definir un contenedor como una estructura con capacidad de gestionar estructuras complejas de maquetación. El contenido se puede disponer usando 2D, como si fuera una rejilla, de ahí su nombre.
- ▶ Propiedades:
  - **display:grid;**
  - **display:inline-grid;**
- ▶ La principal desventaja de GRID respecto a FLEX es la de la “*alineación*” dentro de las celdas. Sin embargo, nada impide utilizar las dos. Por lo que un elemento dentro de GRID, puede ser a su vez un contenedor FLEX.



# 2 PROPIEDADES DEL CONTENEDOR

## 2.1 DEFINIR COLUMNAS

- **grid-template-columns**: Permite definir tanto el n° de columnas como el tamaño de las mismas. Tendrá tantas columnas como valores pongamos. Los valores se pueden expresar de diferentes maneras. Las principales son:

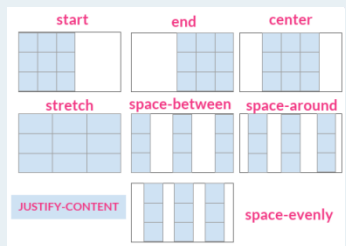
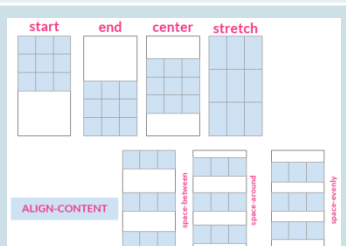
Unidad	Descripción	Ejemplo
%		<b>grid-template-columns: 20% 50% 30%;</b>
Píxeles		<b>grid-template-columns: 100px auto 100px 100px;</b>
Auto	El sistema calculará automáticamente su ancho en función del contenido. Además, las columnas con valor <i>auto</i> son estiradas si así lo especificamos en las propiedades <i>align-content</i> y <i>justify-content</i> . Como por defecto, esas propiedades tienen el valor <i>stretch</i> , las columnas <i>auto</i> ocuparán el espacio disponible en el <i>contenedor grid</i> .	<b>grid-template-columns: auto auto auto auto;</b>
fr	Permite distribuir el espacio sobrante de forma proporcional a los valores que establezcamos	<b>grid-template-columns: 2fr 100px 1fr 2fr;</b> En este caso, el espacio que queda libre lo dividimos en cinco partes y le daremos 2 a la primera columna, 1 a la tercera y 2 a la última. La segunda tendrá 100px
repeat	Permite repetir una misma configuración el número de veces que le indiquemos	<b>grid-template-columns: repeat(3, 20%) auto;</b>
[identificador]	Asignar un identificador entre corchetes al índice numérico que identifica la línea donde comienza o termina la columna. Cuando declaramos un grid, a cada línea se le asigna por defecto un índice numérico.	<b>grid-template-columns: [id] 100px [nombre] 300px [apellidos] auto [fin];</b>

1	2	3	1
id	nombre	apellidos	fin

## 2.2 FILAS Y SEPARACIÓN

Propiedad	Descripción	Ejemplo
<i>grid-template-rows</i>	Controla la altura de las distintas filas. Funciona igual que la propiedad grid-template-columns.	
<i>grid-column-gap</i>	Controla la separación entre las columnas.	
<i>grid-row-gap</i>	Controla la separación entre las filas.	

## 2.3 ALINEACIÓN

Propiedad	Descripción	Ejemplo
<i>justify-items</i>	Permite ajustar el contenido horizontalmente dentro de la celda. <b>start end center stretch</b>	<b>justify-items: center;</b>
<i>align-items</i>	Permite ajustar el contenido verticalmente dentro de la celda. <b>start end center stretch</b>	<b>align-items: start;</b>
<i>place-items</i>	Propiedad resumen de las dos anteriores. Primero la vertical y luego la horizontal.	<b>place-items: center start;</b>
<i>justify-content</i>	Tiene efecto si el grid no ocupa el espacio total del contenedor. Lo que hace es alinear horizontalmente el grid. <b>start end center stretch* space-between space-around space-evenly</b>  * No	
<i>align-content</i>	Configura la alineación vertical del grid <b>start end center stretch* space-between space-around space-evenly</b>	

# 3 PROPIEDADES DE LOS GRID ITEMS

Propiedad	Descripción	Ejemplo
<i>grid-column-start</i> <i>grid-column-end</i> <i>grid-row-start</i> <i>grid-row-end</i>	Permiten definir qué celdas del grid va a ocupar el elemento <b>auto</b>   <b>span n°</b>   <b>n°</b>	<b>grid-column-start: 1;</b> <b>grid-column-end: 3;</b> <b>grid-row-start: 1;</b> <b>grid-row-end: 2;</b>
<i>grid-column: start end</i> <i>grid-row: start end</i>	Propiedades shorthand	<b>grid-column: 1 / 2 span</b> <b>grid-row: 1 / 2</b>
<i>justify-self</i>	Sobrescribe el valor justify-items para el elemento.	
<i>align-self</i>	Sobrescribe el valor align-items para el elemento.	

# 4 GRID AREAS

- ▶ Las **grid-areas** permiten hacer la maquetación de una forma más fácil.
- ▶ La idea es utilizar un nombre en lugar de tener que especificar las posiciones de cada elemento.
- ▶ Propiedades:
  - **grid-template-areas**: Se define a nivel de contenedor y permite darle un nombre a las diferentes zonas del grid. Para ello pondremos entre comillas los nombres que le damos a las celdas de cada fila.
  - **grid-area**: Se establece a nivel de elemento. Permite asignarle a ese elemento una de las zonas que hemos definido en el contenedor.

```
<body>
  <div class="contenedor">
    <div class="p1">Cabecera</div>
    <div class="p2">Menú Lateral</div>
    <div class="p3">Sección principal</div>
    <div class="p4">Pie</div>
  </div>
</body>
</html>
```

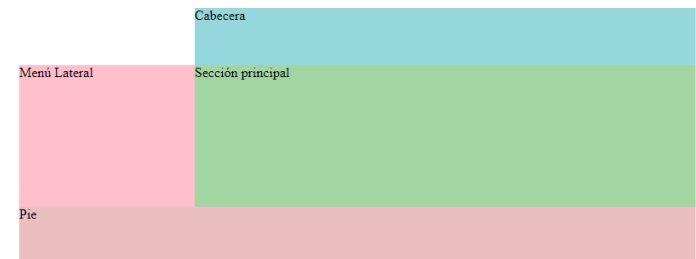
```
.contenedor{
  display:grid;
  grid-template-columns:repeat(5,auto);
  grid-template-rows:repeat(5,auto);
  height:300px;
  grid-template-areas:". cab cab cab cab"
  "menu main main main main"
  "menu main main main main"
  "menu main main main main"
  "pie pie pie pie pie"
}

.p1{
  grid-area: cab;
  background-color:#95d7dd;
}

.p2{
  grid-area:menu;
  background-color:pink;
}

.p3{
  grid-area:main;
  background-color:#a2d5a2;
}

.p4{
  grid-area:pie;
  background-color:#ebbfbf;
}
```



# 5 CAMBIAR GRID AREA Y RESOLUCIÓN

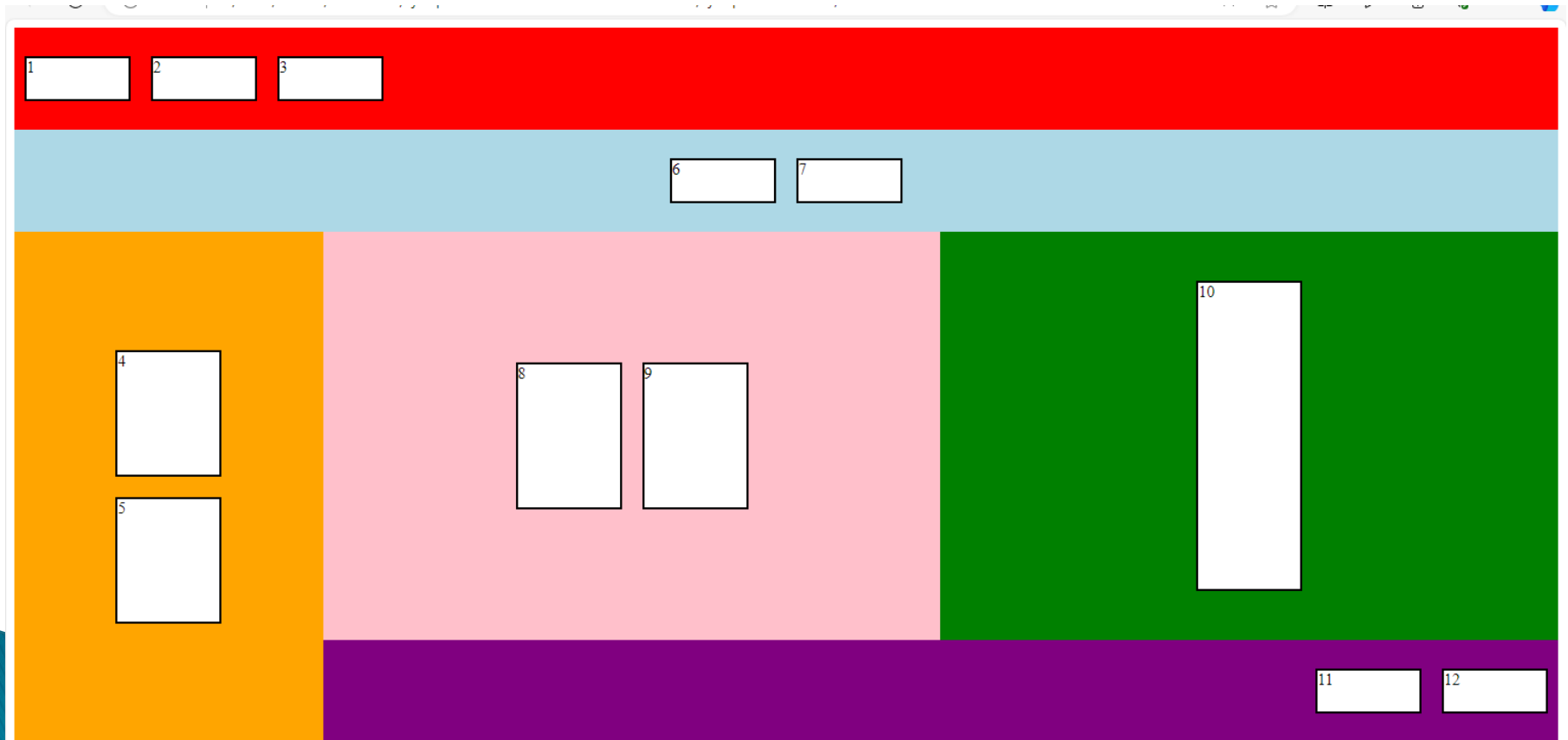
- ▶ Es posible configurar los grid área en función de la resolución de la pantalla.
- ▶ Se define una media query con **@media (max-width: 768px)**. En este ejemplo se aplican estilos específicos cuando el ancho de la pantalla sea de hasta 768px.

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr;  
  grid-template-areas:  
    "header header"  
    "sidebar content"  
    "footer footer";  
}  
  
/* Estilos para pantallas de hasta 768px */  
@media (max-width: 768px) {  
  .container {  
    grid-template-areas:  
      "header header"  
      "content content"  
      "sidebar sidebar"  
      "footer footer";  
  }  
}
```



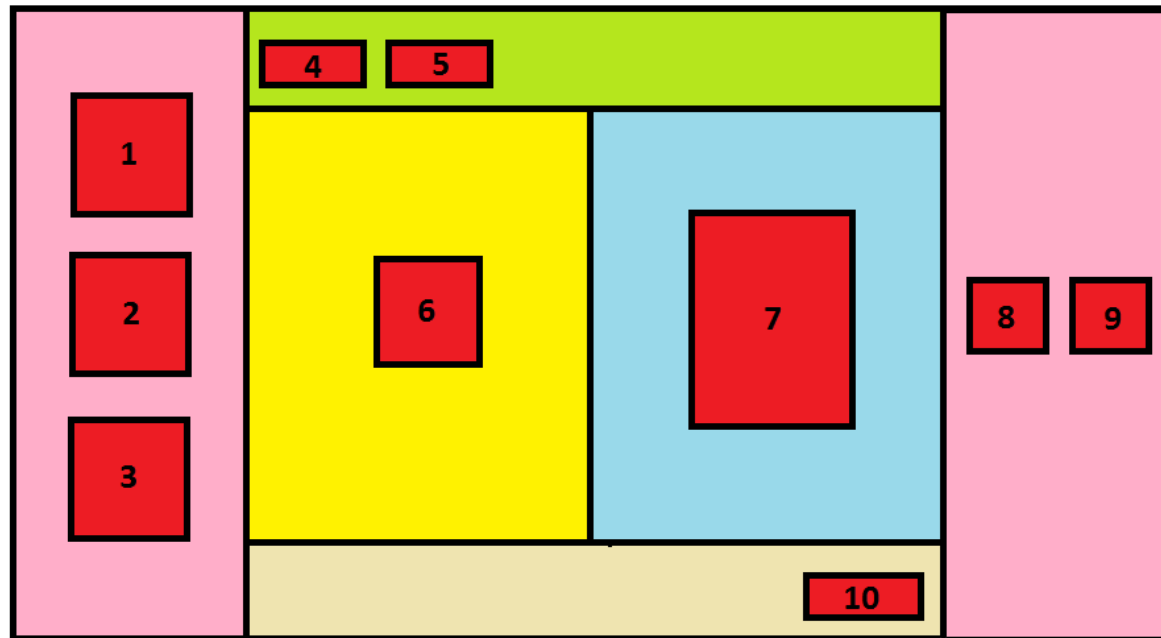
# EJERCICIO 1

- ▶ Usando css grid y flex-box, realiza una hoja de estilos para que ejercicio1.html se visualiza tal y como se muestra en la siguiente imagen. (No puedes modificar el .html)



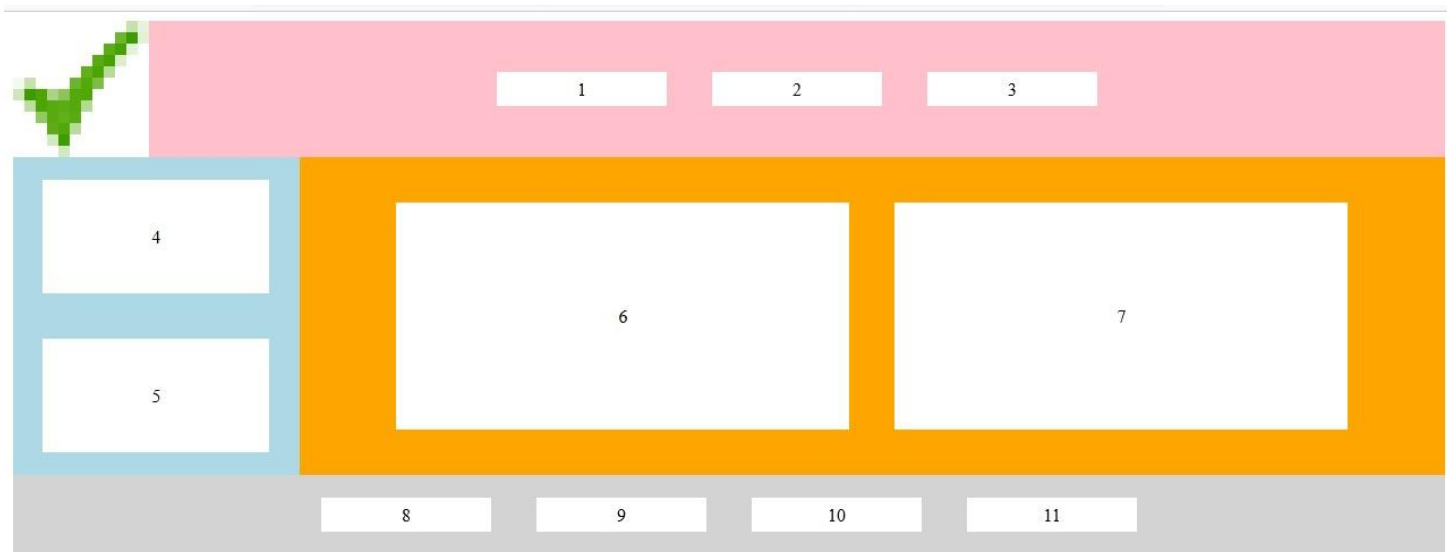
# EJERCICIO 2

- ▶ Usando css grid y flex-box, realiza una hoja de estilos para que ejercicio2.html se visualiza tal y como se muestra en la siguiente imagen. (No puedes modificar el .html)



# EJERCICIO 3

- ▶ Usando css grid y flex-box, realiza una hoja de estilos para que ejercicio3.html se visualiza tal y como se muestra en la siguiente imagen. (No puedes modificar el .html)



# EJERCICIO 4

- ▶ Realiza la página IES AUGUSTÓBRIGA (Ejercicio6 del boletín de ejercicios) utilizando css grid y flex-box para colocar los elementos.

