

INTRODUCCIÓN A CSS GRID LAYOUT

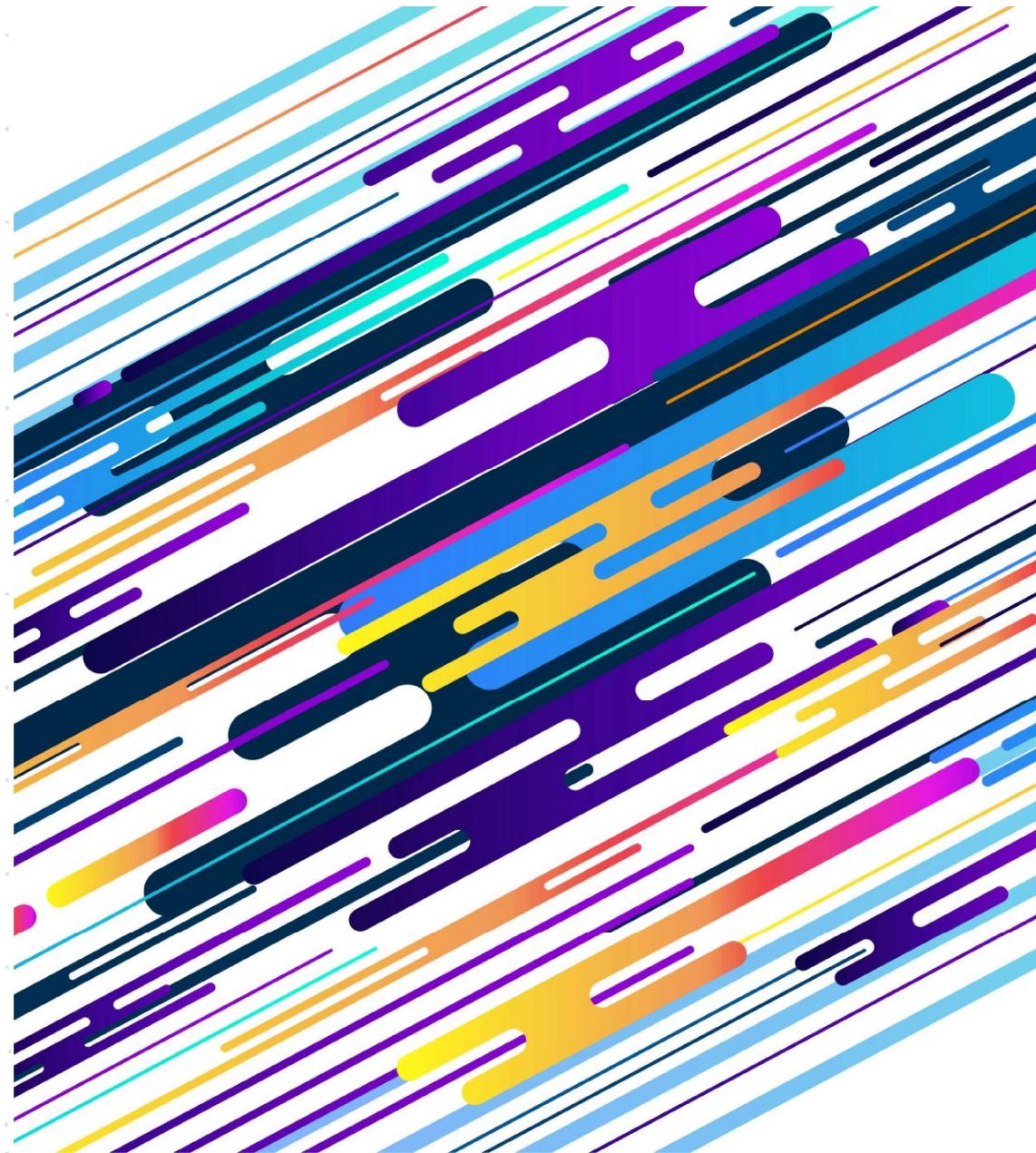
UF1- Programació amb XML

M04-Llenguatges de marques

CFGs DAW

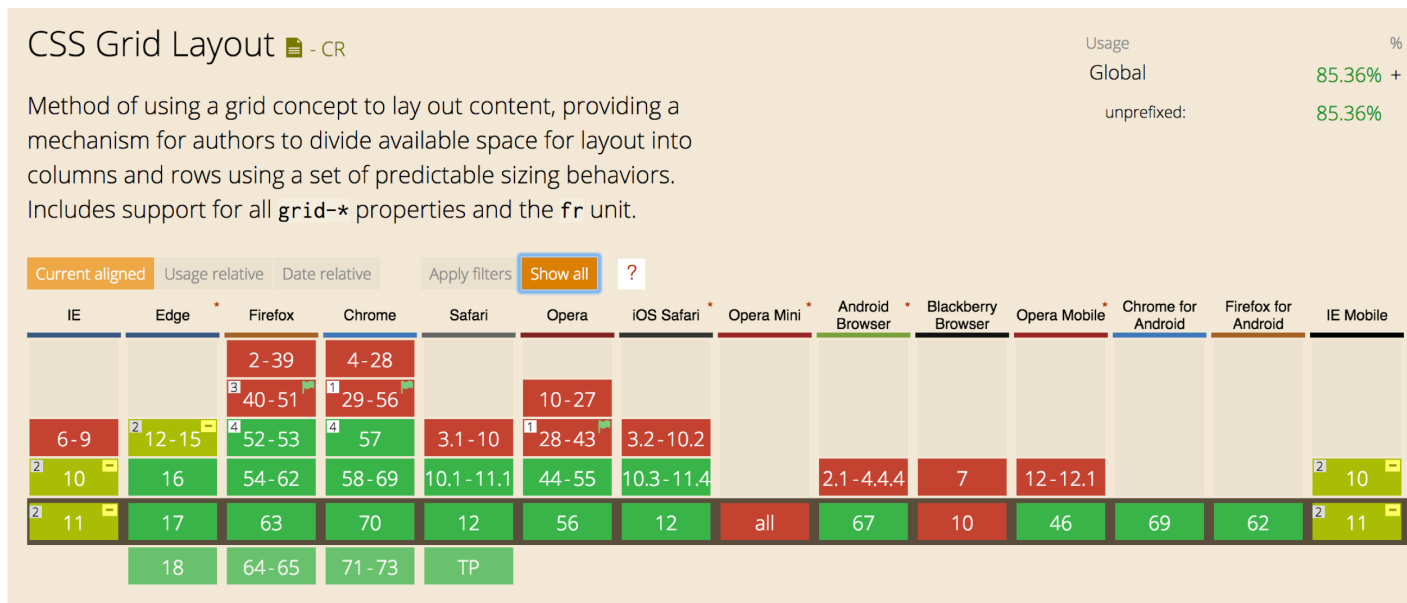
Professor: Marc Callejón

ETP Xavier



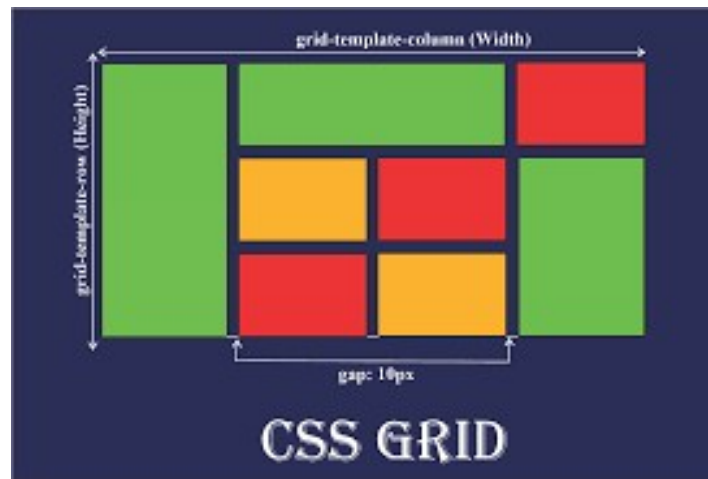
Grid Layout

- Soporte mayoritario en todos los navegadores

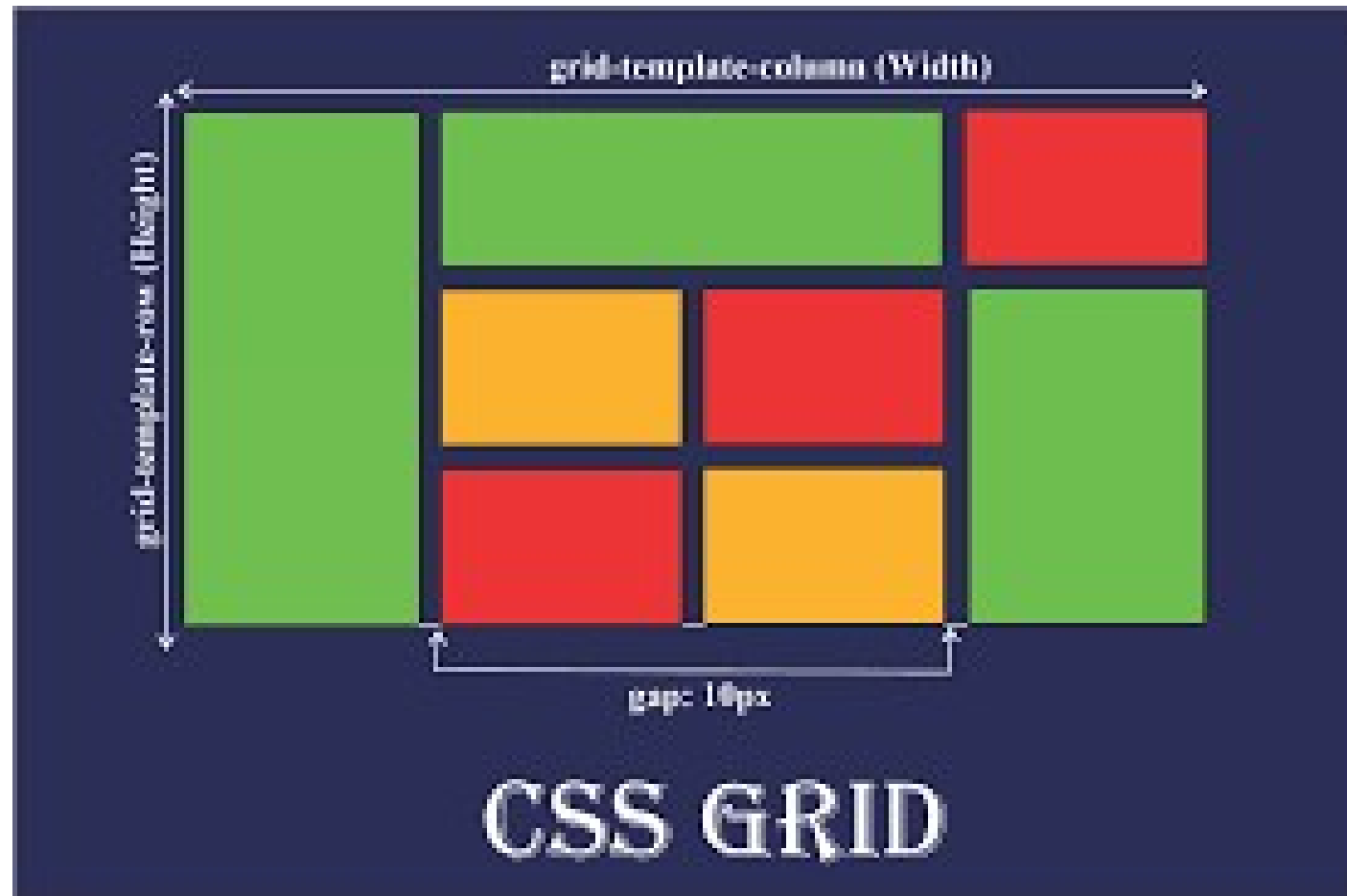


Grid Layout

- Módulo de CSS3 permite definir un grid (parrilla) de dos dimensiones en el layout.
- Permite definir el comportamiento del eje de las x y de la y



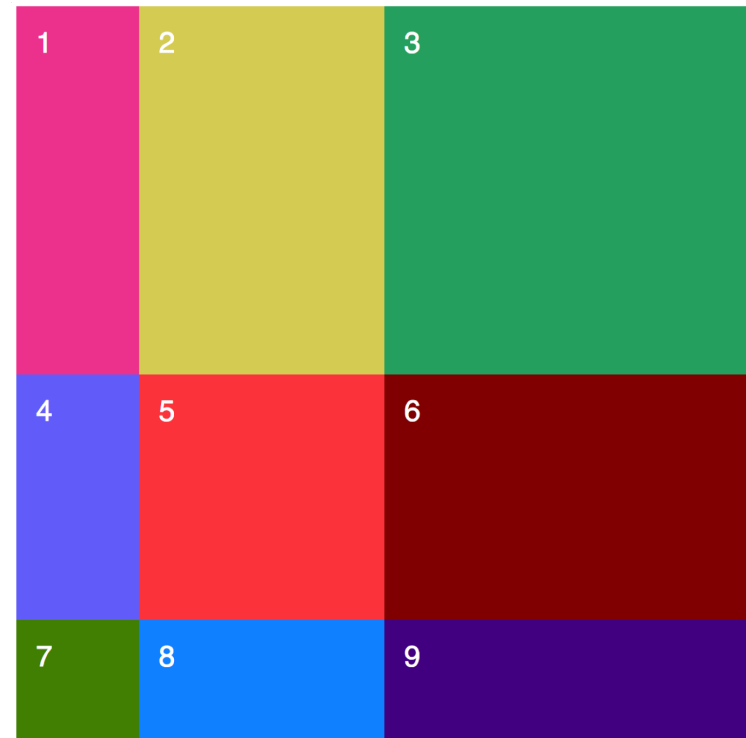
Grid Layout



Grid Layout

- Grid-template-<columns | rows>

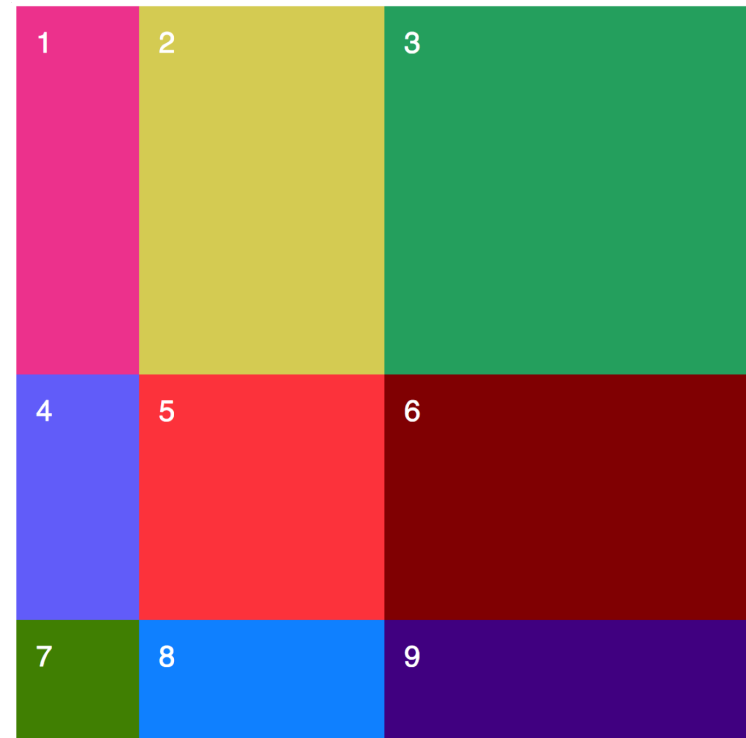
```
.container{  
  display: grid;  
  grid-template-columns: 100px 200px 300px;  
  grid-template-rows: 300px 200px 100px;  
}
```



Grid Layout

- Grid-template

```
.container{  
  display: grid;  
  grid-template: 300px 200px 100px / 100px 200px 300px;  
}
```



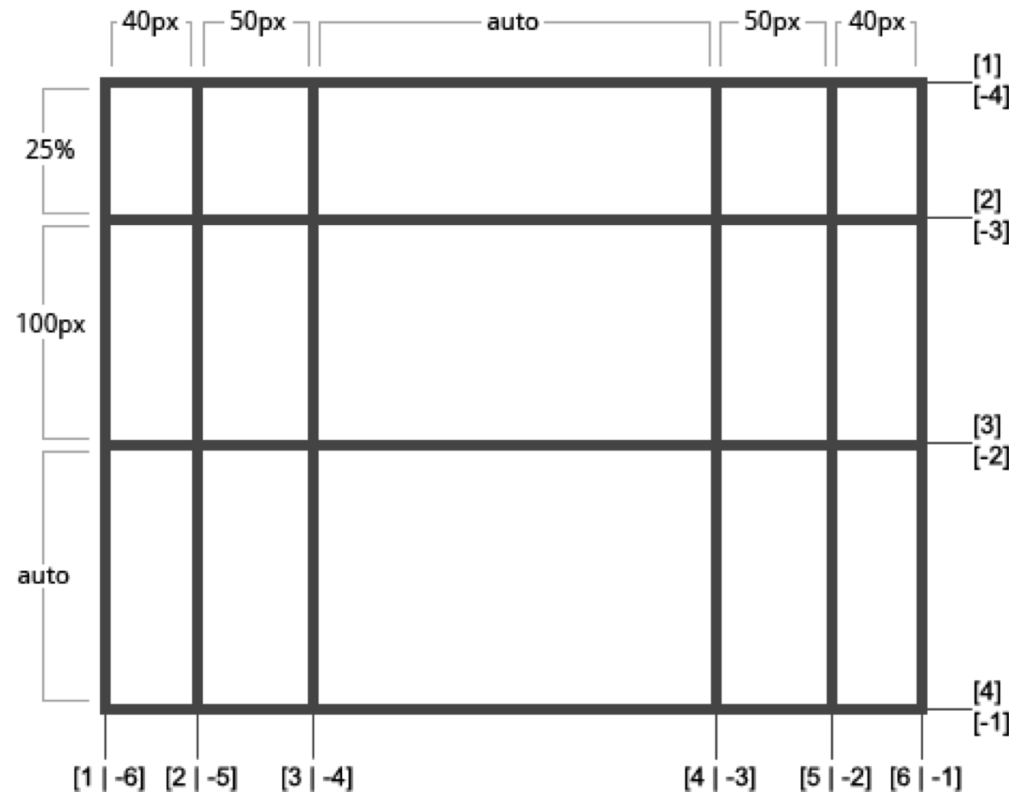
Fraction Unit FR

```
.container{  
  display: grid;  
  grid-template-columns: 1fr 3fr 1fr;  
  grid-template-rows: 3fr 1fr 1fr;  
}
```



Grid Layout

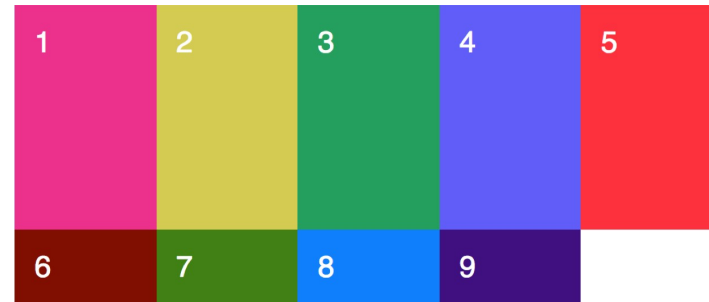
- Nomenclatura filas y columnas (default)



Grid Layout

- Repeat

```
.container{  
  display: grid;  
  grid-template-columns: repeat(5, 1fr);  
  grid-template-rows: 3fr 1fr 1fr;  
}
```



```
.container{  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-template-rows: repeat(3, 1fr);  
}
```



Grid Layout

- Repeat auto-fit: estira los elementos para que ocupen el ancho disponible

```
.container{  
  display: grid;  
  grid-template-columns: repeat(auto-fit,minmax(100px, 1fr));  
  grid-template-rows: 3fr 1fr 1fr;  
}
```

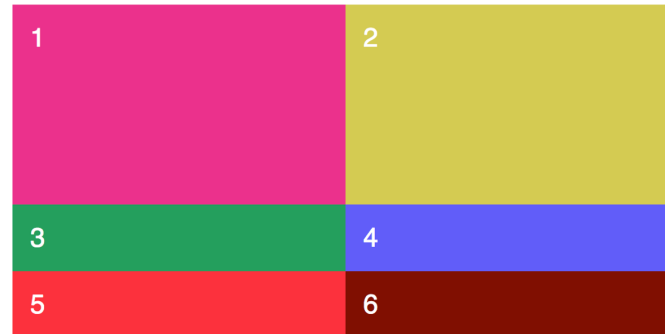
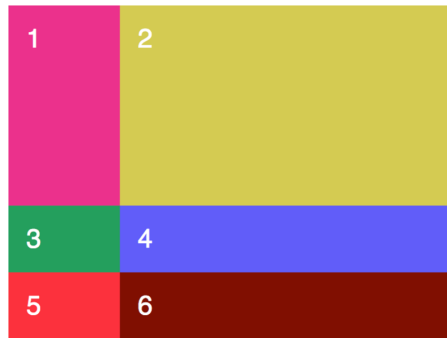


Lo que nos dice esta norma viene a ser algo tipo “pinta tantas columnas como puedas que vayan de 100px hasta 1fr (el máximo espacio que esté disponible)”

Grid Layout

- MinMax

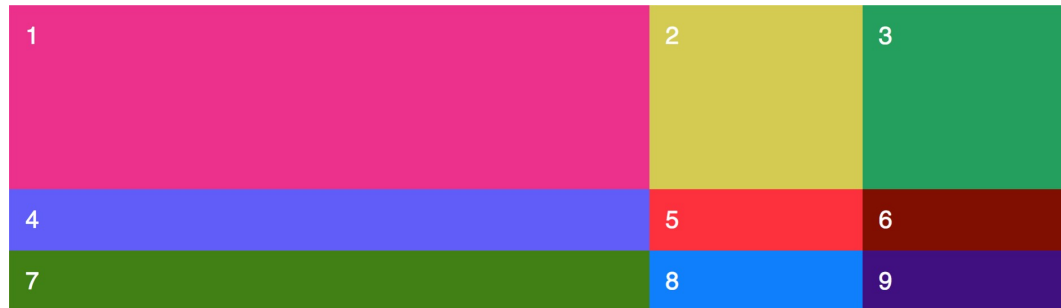
```
.container{  
  display: grid;  
  grid-template-columns: minmax(100px, 300px) 300px;  
  grid-template-rows: 3fr 1fr 1fr;  
}
```



Grid Layout

- MinMax

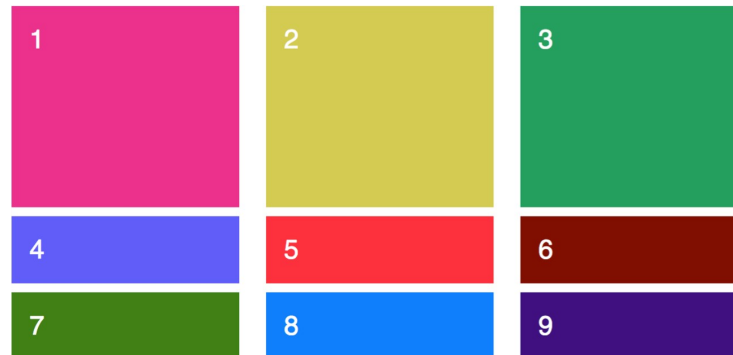
```
.container{  
  display: grid;  
  grid-template-columns: minmax(200px, 60%) repeat(2,1fr);  
  grid-template-rows: 3fr 1fr 1fr;  
}
```



Grid Layout

- Grid-column-gap / Grid-row-gap

```
.container{  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-template-rows: 3fr 1fr 1fr;  
  grid-column-gap: 1.5rem;  
  grid-row-gap: .5rem;  
}
```



Grid Layout

- Grid-template-areas

```
.container{  
  display: grid;  
  
  grid-template-areas:  
    'header header header header header header'  
    'menu main main main right right'  
    'menu footer footer footer footer footer';  
  gap: 10px;  
}
```

```
.container > div{  
  border: solid 1px red;  
}
```



The diagram illustrates a CSS Grid layout with 5 items. The grid is defined by the following CSS rules:

```
.container{  
  display: grid;  
  
  grid-template-areas:  
    'header header header header header header'  
    'menu main main main right right'  
    'menu footer footer footer footer footer';  
  gap: 10px;  
}
```

```
.container > div{  
  border: solid 1px red;  
}
```

The items are represented by colored rectangles with numbers 1 through 5 in the top-left corner:

- Item 1: Red rectangle, spanning the entire width of the grid (header area).
- Item 2: Blue rectangle, spanning the first column of the second row (menu area).
- Item 3: Yellow rectangle, spanning the next three columns of the second row (main area).
- Item 4: Green rectangle, spanning the last two columns of the second row (right area).
- Item 5: Grey rectangle, spanning the entire width of the third row (footer area).

```
.item1{background: red; grid-area: header; }  
.item2{background: blue; grid-area: menu; }  
.item3{background: yellow; grid-area: main;}  
.item4{background: green; grid-area: right;}  
.item5{background: grey; grid-area: footer;}
```