

# Layout

Grid CSS



# CSS Grid

Ofrece un sistema de layout basado en filas y columnas.  
Facilita el diseño web de páginas sin tener que usar floats y posicionamiento.

## html

```
<div class="my-grid-container">  
  <div class="item">...</div>  
  <div class="item">...</div>  
  <div class="item">...</div>  
  ...  
  ...  
  <div class="item">...</div>  
</div>
```

## display

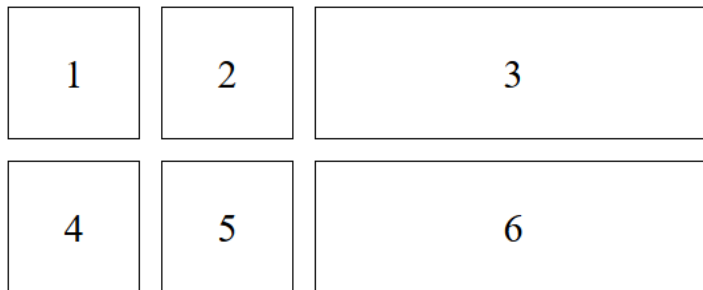
```
.my-grid-container {  
  display: grid | inline-grid;  
}
```

# CSS Grid

grid-template-columns

grid-template-rows

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

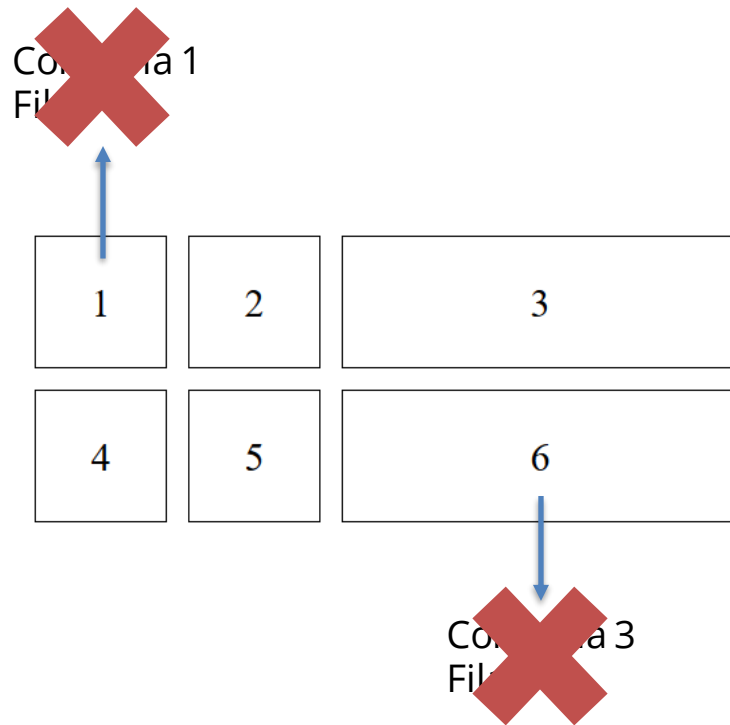


# CSS Grid

## grid-template-columns

## grid-template-rows

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

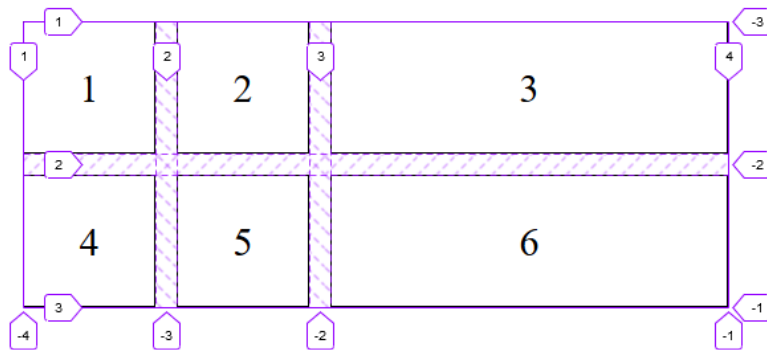


# CSS Grid

grid-template-columns

grid-template-rows

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

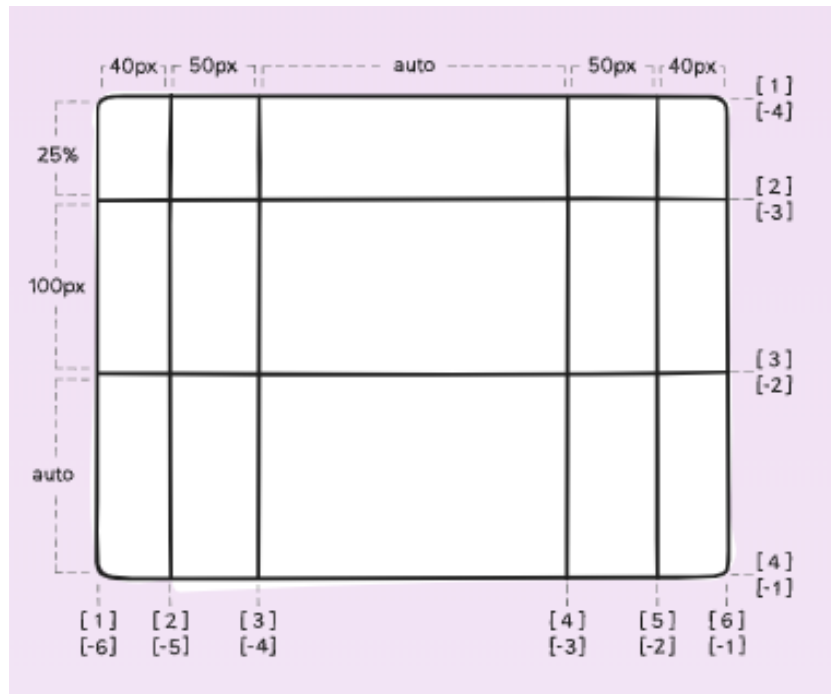


# CSS Grid

grid-template-columns

grid-template-rows

```
.my-grid-container {  
  grid-template-columns: 40px 50px auto 50px 40px;  
  grid-template-rows: 25% 100px auto;  
}
```

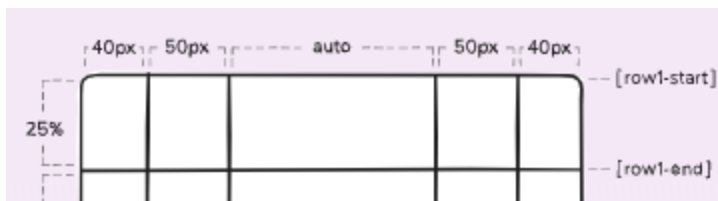


# CSS Grid

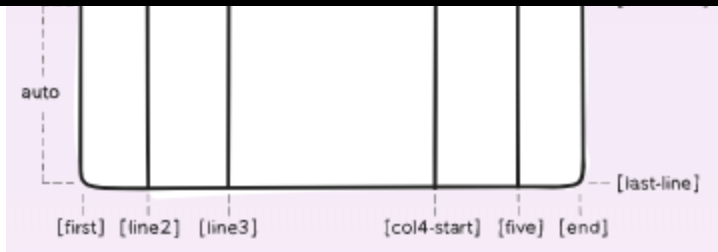
## grid-template-columns

## grid-template-rows

```
.my-grid-container {  
  grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];  
  grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];  
}
```



```
.container {  
  grid-template-rows: [row1-start] 25% [row1-end row2-start] 25% [row2-end];  
}
```



# CSS Grid

grid-column-start

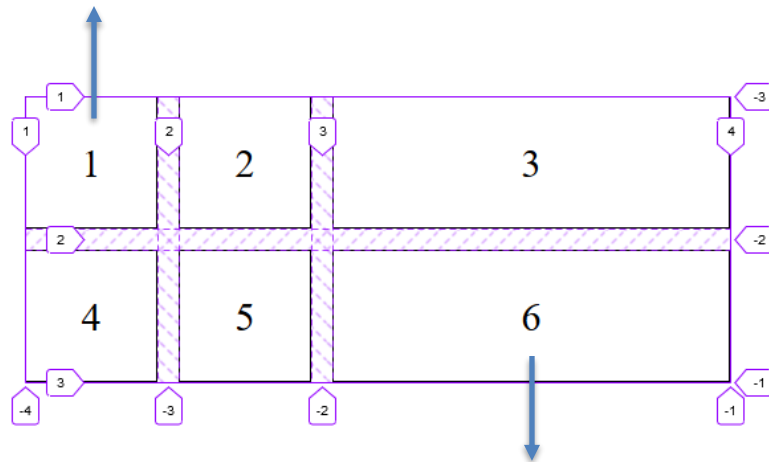
grid-column-end

grid-row-start

grid-row-end

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

```
.item-1 {  
  grid-column-start: 1;  
  grid-column-end: 2;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}
```



```
.item-6 {  
  grid-column-start: 3;  
  grid-column-end: 4;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```



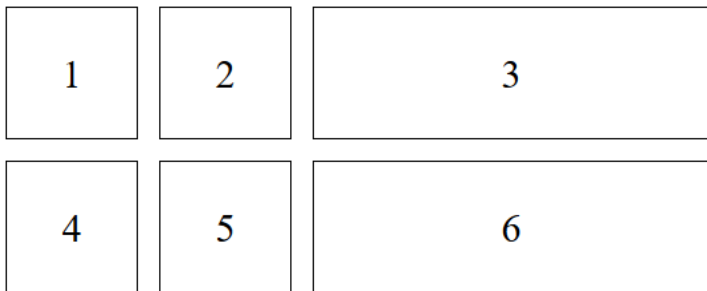
# CSS Grid

grid-column-start

grid-column-end

grid-row-start

grid-row-end



```
.item-1 {  
  grid-column-start: 1;  
  grid-column-end: 2;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}
```

```
.item-2 {  
  grid-column-start: 2;  
  grid-column-end: 3;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}
```

```
.item-3 {  
  grid-column-start: 3;  
  grid-column-end: 4;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}
```

```
.item-4 {  
  grid-column-start: 1;  
  grid-column-end: 2;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```

```
.item-5 {  
  grid-column-start: 2;  
  grid-column-end: 3;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```

```
.item-6 {  
  grid-column-start: 3;  
  grid-column-end: 4;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```

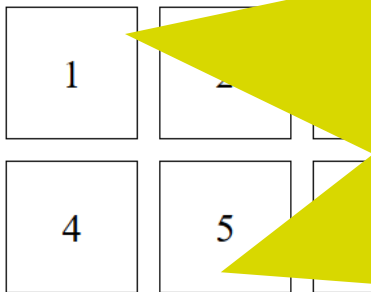
# CSS Grid

SHORT HAND:  
grid-column  
grid-row

```
.item-1 {  
  grid-column: 1 / 2;  
  grid-row: 1 / 2;  
}
```

Esto sólo es el ejemplo de  
los valores que tomarían  
cada item por defecto:  
AUTO

Para este resultado no es  
necesario indicar cada celda

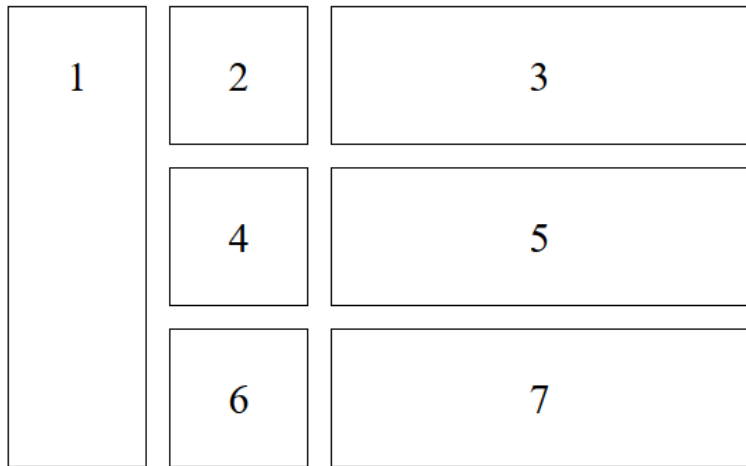


```
.item-6 {  
  grid-column: 3 / 4;  
  grid-row: 2 / 3;  
}
```

# CSS Grid

## SHORT HAND: grid-column grid-row

```
.my-grid-container {  
  display: grid;  
  grid-template-columns: 100px 100px 300px;  
  grid-template-rows: [row1-start] 100px 100px 100px [last-row-line];  
  gap: 1rem;  
}
```



```
.item-1 {  
  grid-row: 1 / 4;  
}
```

```
.item-1 {  
  grid-row: 1 / span 3;  
}
```

```
.item-1 {  
  grid-row: row1-start / span 3;  
}
```

```
.item-1 {  
  grid-row: row1-start / last-row-line;  
}
```

# CSS Grid

## Notación <repeat>

```
.my-grid-container {  
  display: grid;  
  grid-template-columns: 100px 100px 100px 100px 100px 300px [last-col-line];  
  grid-template-rows: [row-start] 100px [row-start] 100px [last-row-line];  
}
```

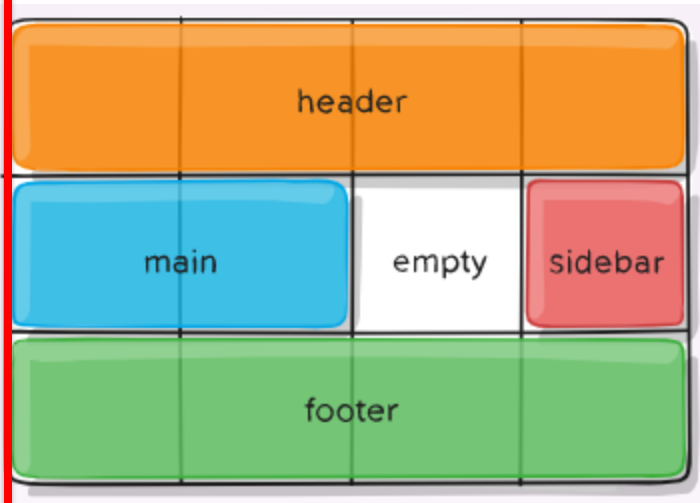
```
.my-grid-container {  
  display: grid;  
  grid-template-columns: repeat(5, 100px) 300px [last-col-line];  
  grid-template-rows: repeat(2, [row-start] 100px) [last-row-line];  
}
```

Si varias líneas comparten el mismo nombre,  
Se pueden referenciar por el nombre y el índice

```
.nav {  
  grid-row: row-start 2 / last-row-line;  
}
```

# Grid CSS

## AREAS



Las líneas se nombran automáticamente para el inicio y fin de las filas y columnas. Esto implica que una línea puede tener múltiples nombres. En este ejemplo, la primera línea vertical izquierda puede referenciarse como: **header-start**, **main-start** y **footer-start**.

## Contenedor:

```
.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";  
}
```

## Areas:

```
.item-a {  
  grid-area: header;  
}  
.item-b {  
  grid-area: main;  
}  
.item-c {  
  grid-area: sidebar;  
}  
.item-d {  
  grid-area: footer;  
}
```

# Grid CSS

## SHORT HAND: grid-template

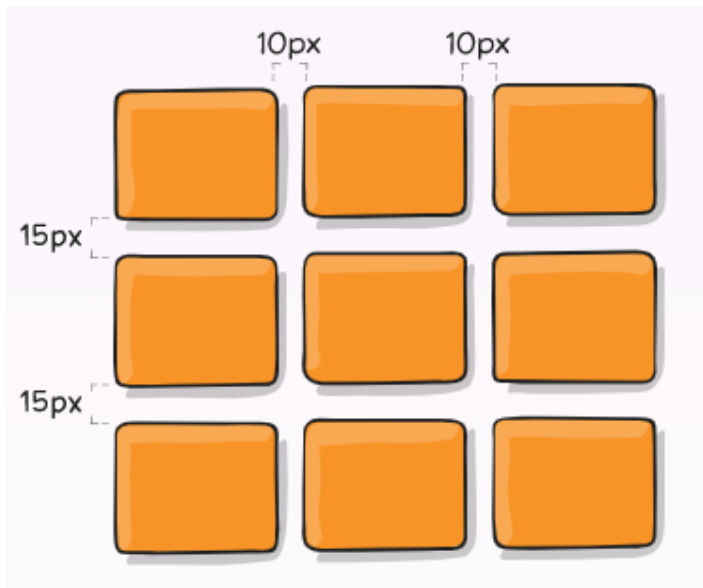
```
.container-1 {  
  display: grid;  
  grid-template-rows: [row1-start] 1fr [row1-end row2-start] 4fr [row2-end row3-start] 1fr [row3-end];  
  grid-template-columns: 1fr 50px 4fr;  
  grid-template-areas:  
    "header header header"  
    "main . sidebar"  
    "footer footer footer";  
}
```

```
.container {  
  display: grid;  
  grid-template:  
    [row1-start] "header header header" 1fr [row1-end]  
    [row2-start] "main . sidebar" 4fr [row2-end]  
    [row3-start] "footer footer footer" 1fr [row3-end]  
    / 1fr 50px 4fr;  
}
```

# Grid CSS

column-gap

row-gap



SHORT HAND

gap: <row-gap> <column-gap>

```
.container {  
  grid-template-columns: 100px 50px 100px;  
  grid-template-rows: 80px auto 80px;  
  gap: 15px 10px;  
}
```

CSS

# Grid CSS

## justify-self

Organiza los "items" en referencia al eje *inline* (fila, horizontal)  
El tamaño de los items dentro del contenedor  
en el eje vertical)

```
.container {  
  justify-items: center;  
}
```

Este comportamiento puede ser aplicado a  
un item concreto con la propiedad  
**justify-self**





# Grid CSS

## align-items

Organiza los "items" en referencia al eje vertical (columna)  
El contenido se alinea dentro del contenedor  
(se alinea en el eje horizontal)

```
.container {  
  align-items:  
}
```

Este comportamiento puede ser aplicado a  
un item concreto con la propiedad  
**align-self**

```
.container {  
  align-items:  
}
```



# Grid CSS

## place-items

### SHORT HAND: place-items

Define los valores de las dos propiedades anteriores en una sólo declaración:

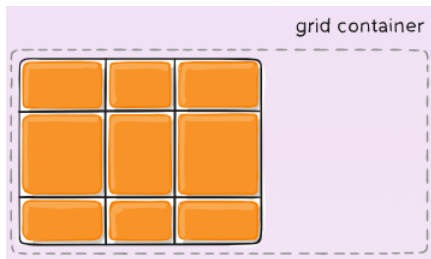
`place-items: <align-items> / <justify-items>`

Si sólo se define un valor, se asigna a las dos propiedades.

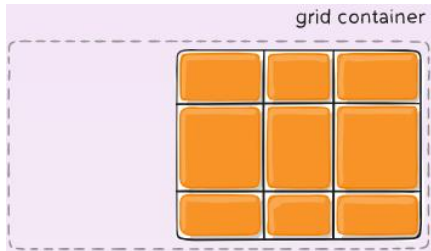
# Grid CSS

## justify-content

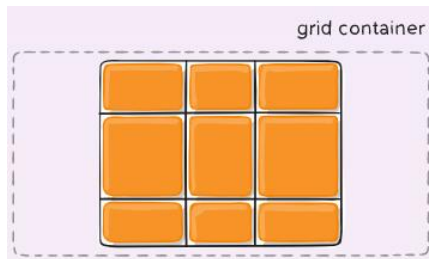
justify-content: start



justify-content: end



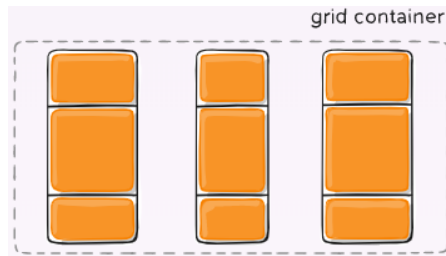
justify-content: center



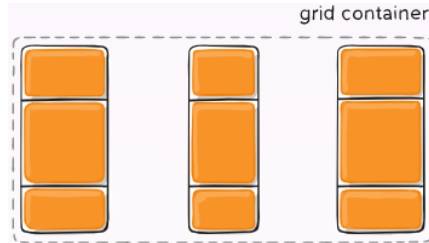
justify-content: stretch



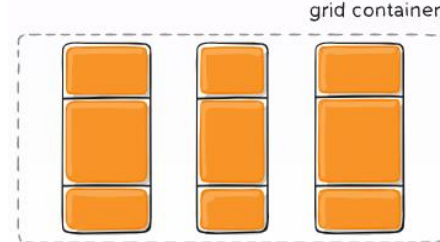
justify-content: space-around



justify-content: space-between



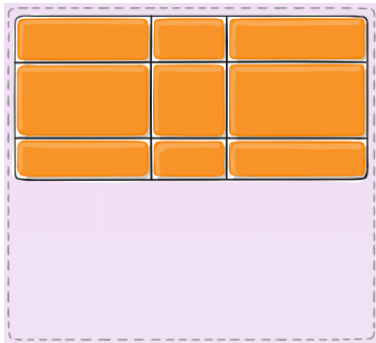
justify-content: space-evenly



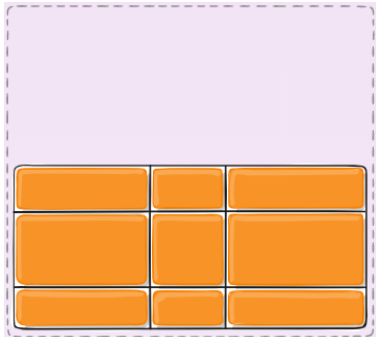
# Grid CSS

## align-content

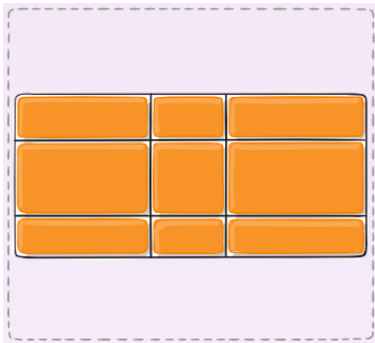
align-content: start



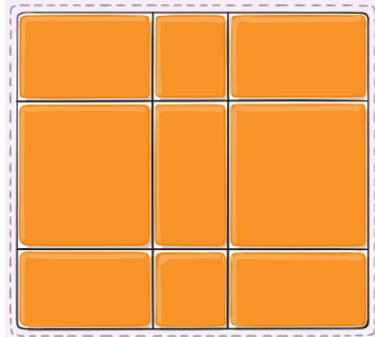
align-content: end



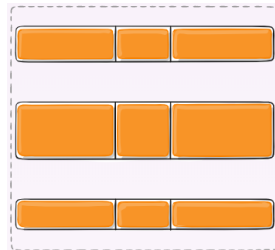
align-content: center



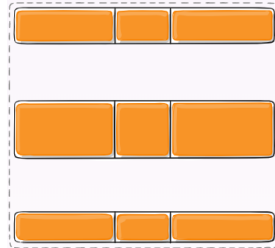
align-content: stretch



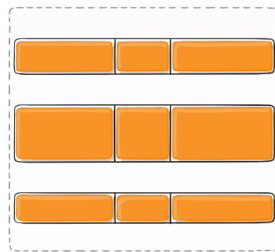
align-content: space-around



align-content: space-between



align-content: space-evenly



# Grid CSS

## place-content

SHORT HAND: place-content

Define los valores de las dos propiedades anteriores en una sólo declaración:

**place-content: <align-content> / <justify-content>**

Si sólo se define un valor, se asigna a las dos propiedades.



[lemoncode.net](https://lemoncode.net)



[@lemoncoders](https://twitter.com/lemoncoders)



[github.com/lemoncode](https://github.com/lemoncode)



[facebook.com/lemoncoders](https://facebook.com/lemoncoders)