Flexbox vs Grid

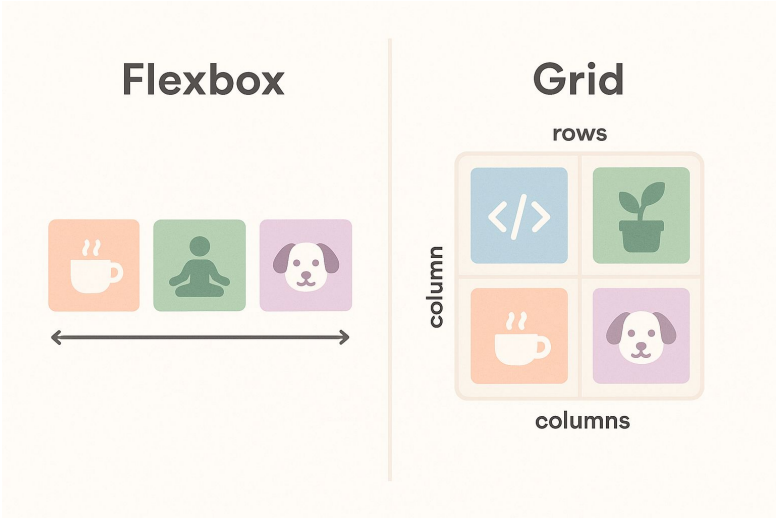
¿cual usar y cuando?

Introducción

¿Que son Flexbox y Grid?

Contenido:

- Flexbox: Sistema de diseño unidimensional (ejes).
- Grid: Sistema de diseño bidimensional (filas y columnas).
- Ambos forman parte de CSS3 y permiten crear layouts más eficientes y adaptativos.



¿Qué es CSS?
Explicación sencilla:

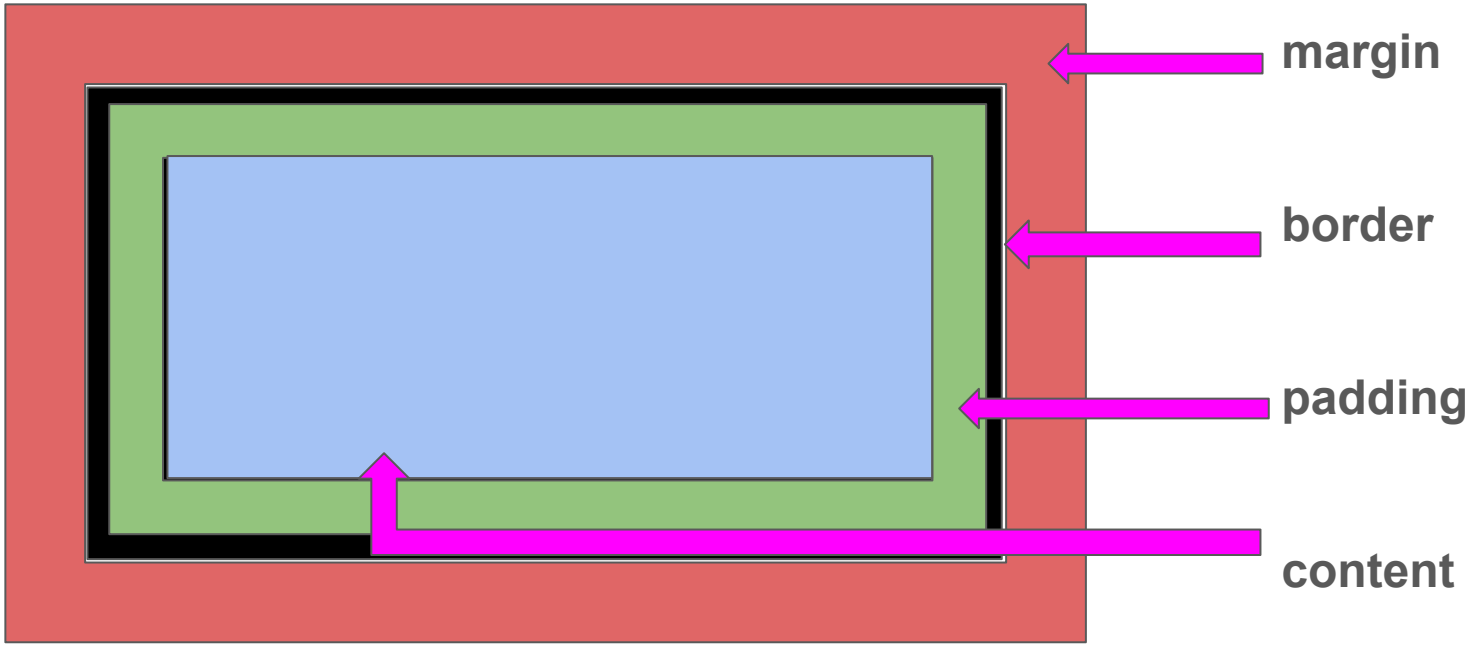
- CSS significa “Cascading Style Sheets”, en español: “Hojas de estilo en cascada”.
- Es lo que usamos para darle estilo a una página web: colores, tamaños, alineaciones,etc.
- Pero además de los colores, también usamos CSS para acomodar los elementos en pantalla(como cajas, botones, imagenes).

¿Que es un “layout”?

Explicación:

- “Layout” significa la forma en que se acomodan los elementos en la página.
- Piensa en una revista: tiene un título arriba, imágenes a un lado, textos debajo.
- En una página web pasa lo mismo: tenemos que decirle al navegador donde va cada cosa.

Modelo de Caja



¿Que es Flexbox?

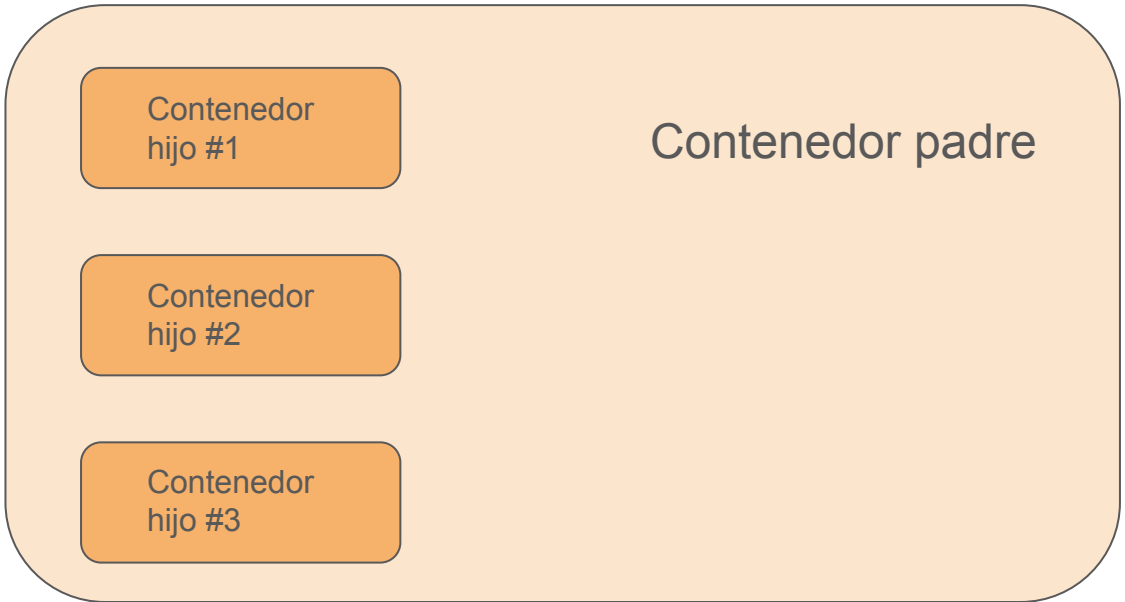
Contenido:

- Diseño unidimensional: se trabaja en **fila** o **columna**.
- Ideal para alinear elementos en una dirección.
- Propiedades clave: `display: flex`, `justify-content`, `align-items`, `flex-wrap`.

Ejemplo práctico:
Alinear un menú horizontal o centrar una tarjeta verticalmente.

¿Cómo utilizar Flexbox?

Al ser Flexbox un módulo completo, podemos utilizarlo modificando una serie de propiedades. Algunas de ellas deben aplicarse en el contenedor “padre”, mientras otras deben aplicarse en los contenedores que hay en el interior del padre, los “hijos”.



Propiedades del contenedor padre

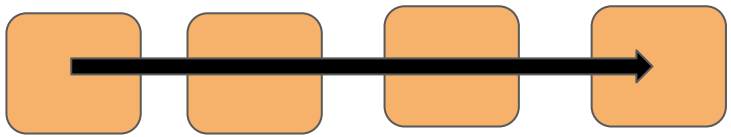
display

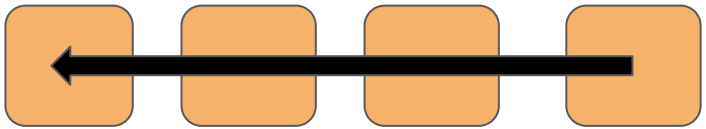
Define un contenedor padre. Puede ser en línea o en bloque dependiendo del valor dado. Además, permite un contexto flexible para todos sus hijos directos.

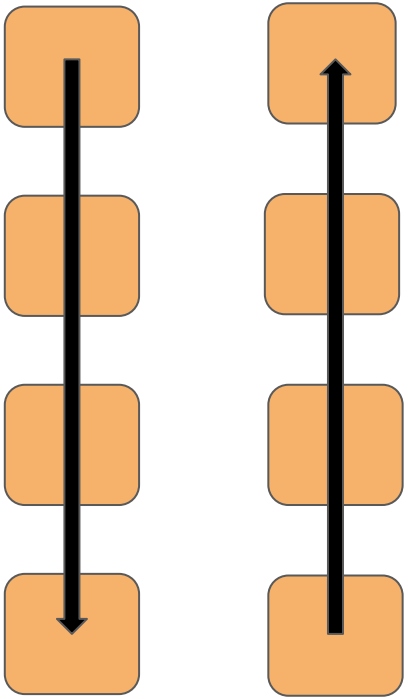
```
.padre {
  display: flex;
  display: inline-flex;
}
```

flex-direction

Establece el eje principal, definiendo así la direccion en la que se colocan los hijos en el contenedor padre. Los hijos se pueden disponer principalmente en filas horizontales o columnas verticales



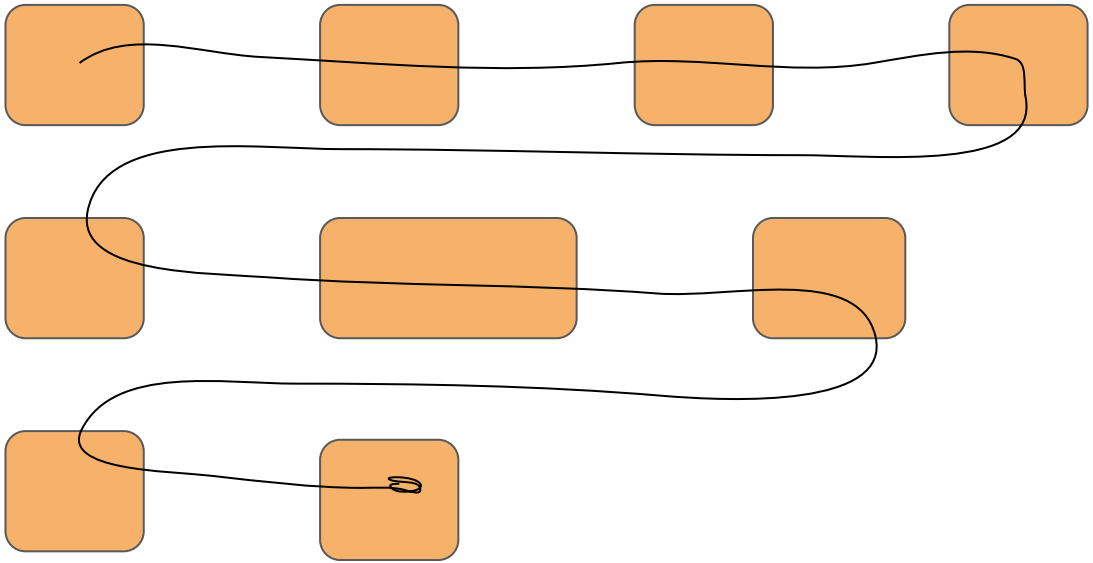




```
.padre {
  flex-direction: row;
  flex-direction: row-reverse;
  flex-direction: column;
  flex-direction: column-reverse;
  }
```

Flex-wrap

Por defecto, todos los hijos intentarán mantenerse alineados. Esta propiedad permite que los hijos se ajusten según la posición necesaria.



```
.padre {
  flex-wrap: nowrap;
  flex-wrap: wrap;
  flex-wrap: wrap-reverse;
}
```

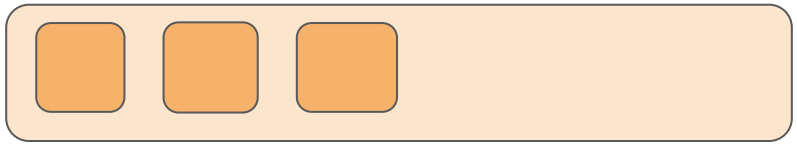
flex-flow

Una abreviatura de las dos propiedades anteriores, flex-direction y flex-wrap, que juntas definen los ejes principal y transversal del contenedor padre.

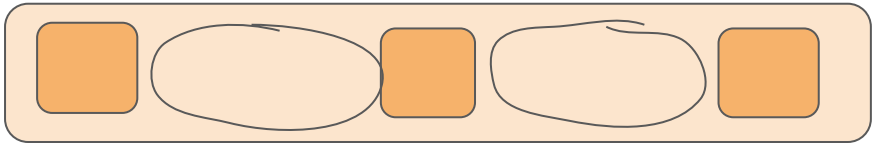
```
.padre {
  flex-direction: column;
  flex-wrap: wrap;
---------------------------
  flex-flow: column wrap;
}
```

justify-content

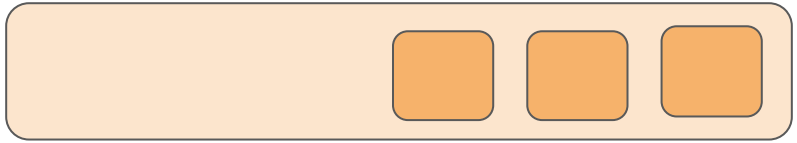
flex-start



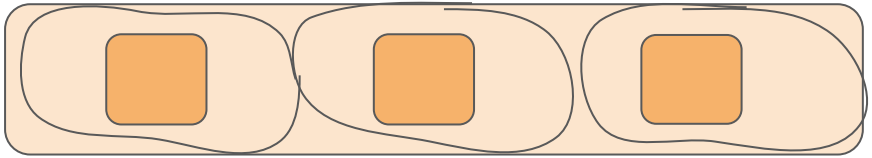
space-between



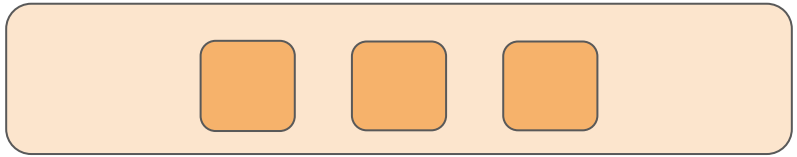
flex-end



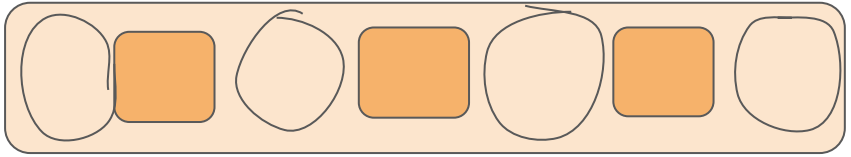
space-around



center



space-evenly

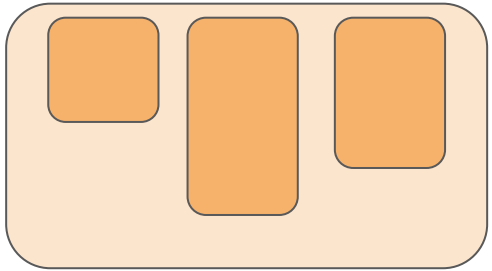


Define la alineación a lo largo del eje principal. Ayuda a distribuir el espacio libre adicional sobrante cuando todos los hijos de una línea son inflexibles o son flexibles pero han alcanzado su tamaño máximo. También ejerce cierto control sobre la alineación de los hijos cuando desbordan la línea

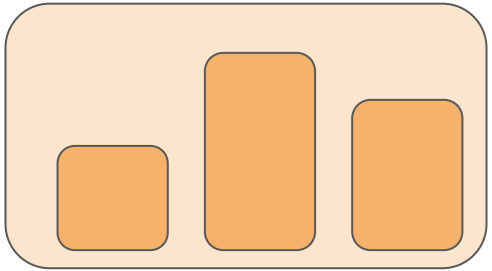
```
.padre {
  justify-content: flex-start,
  justify-content: flex-end;
  justify-content: center;
  justify-content: space-between;
  justify-content: space-around;
  justify-content: space-evenly;
}
```

align-items

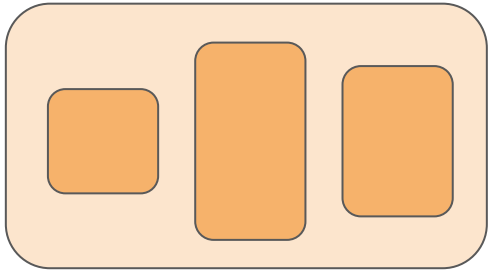
flex-start



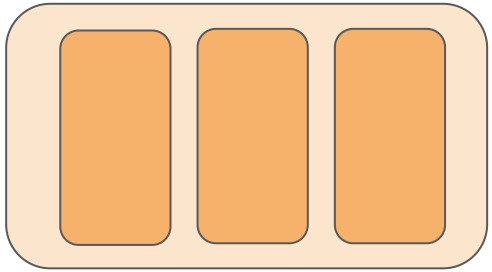
flex-end



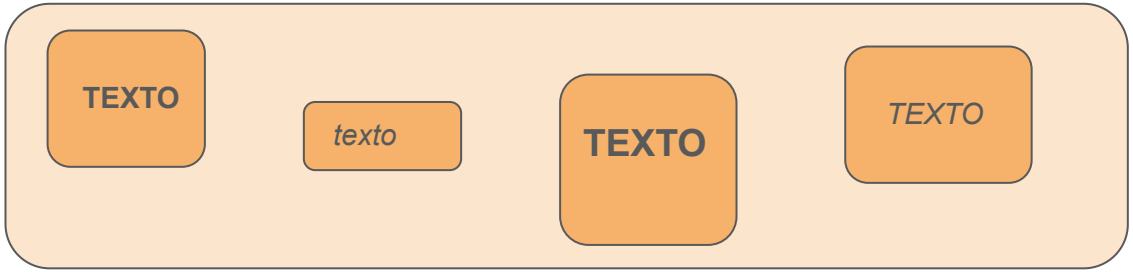
center



stretch



baseline

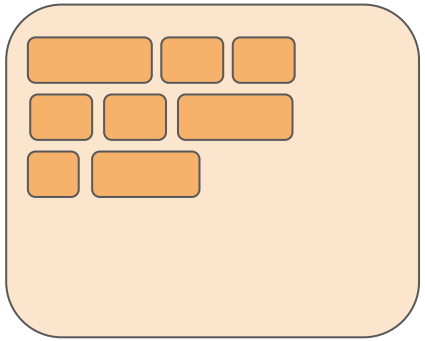


Define el comportamiento predeterminado de cómo se distribuyen los hijos a lo largo del eje transversal de la línea actual. Puede considerarse como la versión de justificación de contenido para el eje transversal (perpendicular al eje principal).

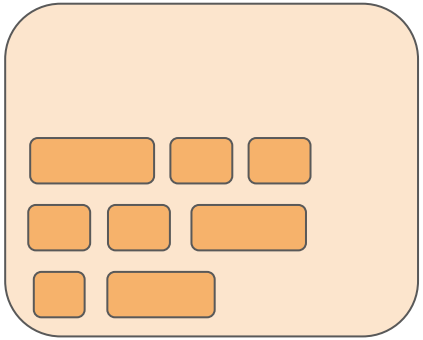
```
.padre {
  align-items: stretch;
  align-items: flex-start;
  align-items: flex-end;
  align-items: center;
  align-items: baseline;
  align-items: first baseline;
  align-items: last baseline;
}
```

align-content

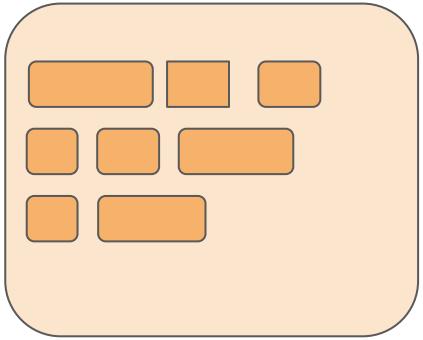
flex-start



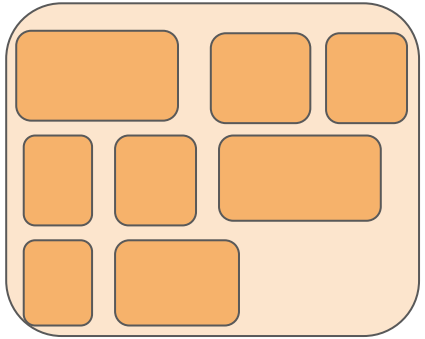
flex-end



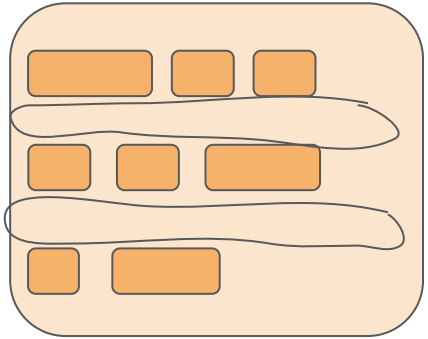
center



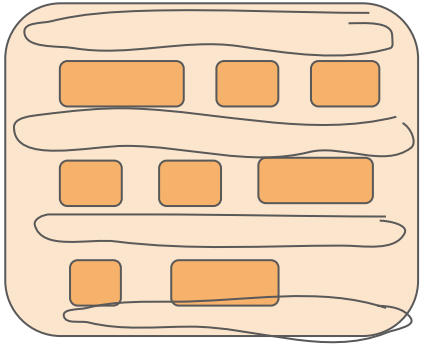
stretch



space-between



space-around

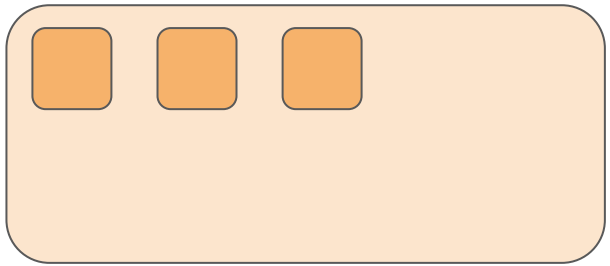


Alinea las líneas de un contenedor padre cuando hay espacio adicional en el eje transversal, similar a como justificar contenido alinea hijos individuales dentro del eje principal.

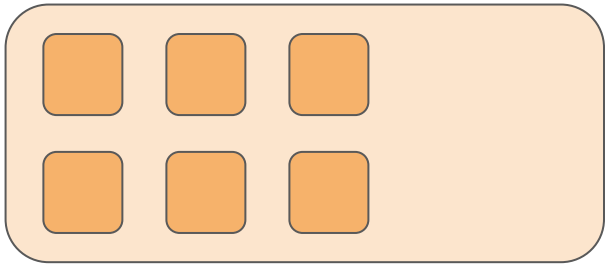
```
.padre {
  align-content: flex-start;
  align-content: flex-end;
  align-content: center;
  align-content: stretch;
  align-content: space-between;
  align-content: space-around;
}
```

gap, row-gap, column-gap

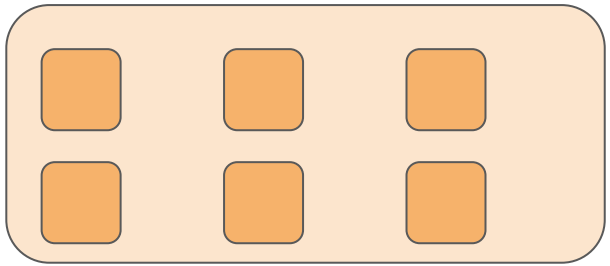
gap:10px



gap:10px



gap:10px 30px

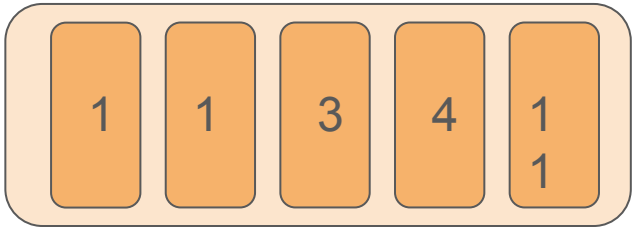


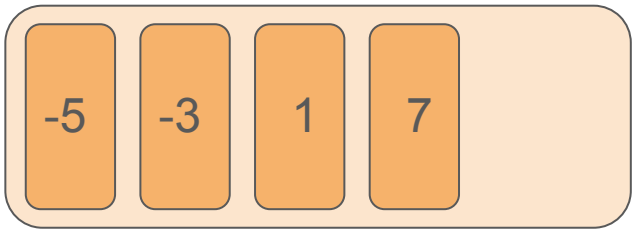
La propiedad gap o “hueco” controla explícitamente el espacio entre hijos. Se aplica ese espacio solo entre hijos y no en los bordes exteriores.

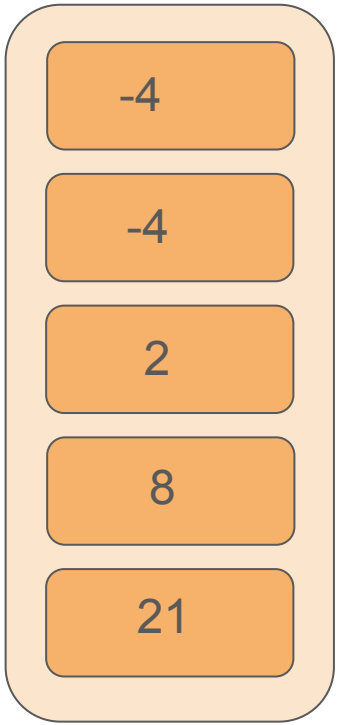
```
.padre {
  display: flex;
  gap: 10px;
  gap: 10px 30px;
  row-gap: 10px;
  column-gap: 30px;
}
```

Propiedades del contenedor hijo

order



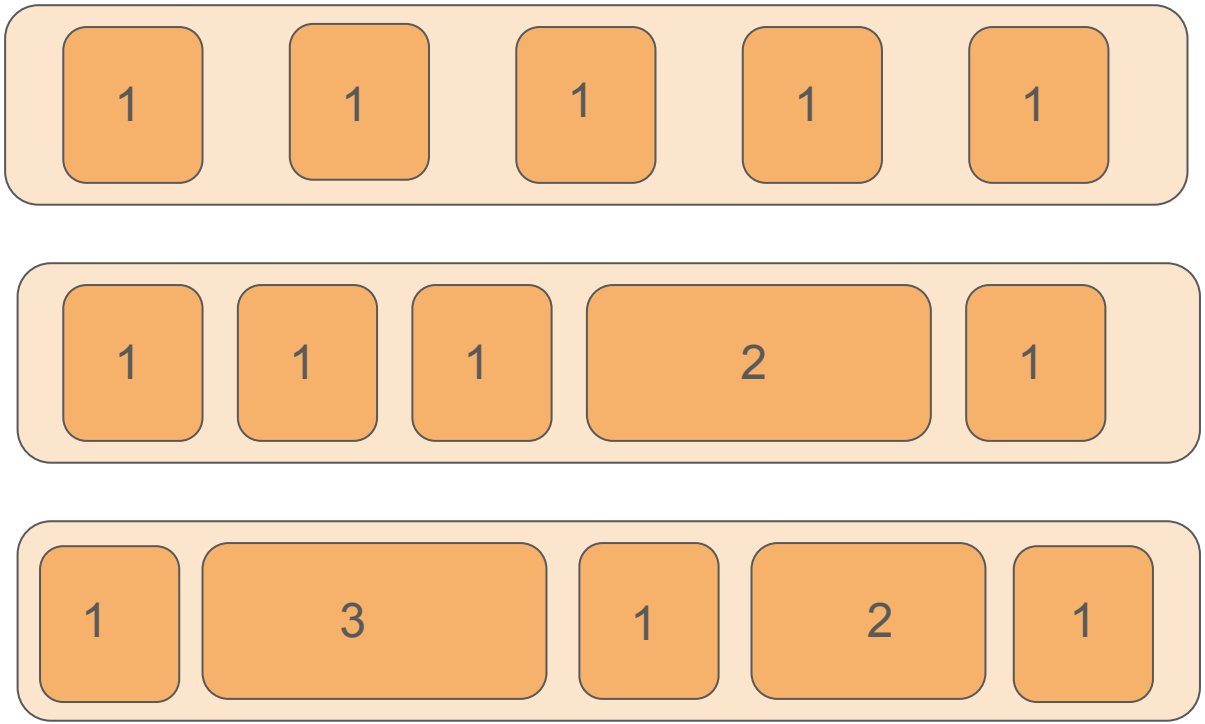




Predeterminadamente los hijos se muestran por orden de origen. La propiedad order controla en que aparecen en el contenedor padre.

```
.hijo {
  order: 5;
}
```

flex-grow



Define la capacidad de un hijo de crecer (aumentar de tamaño) si es necesario. Acepta un valor sin unidades que sirve como proporción. Indica que cantidad de espacio disponible dentro del contenedor padre debe ocupar el articulo.

Si todos los hijos tienen flex-grow configurado en 1, el espacio restante en el contenedor se distribuirá equitativamente entre todos los niños. Si uno de los hijos tiene valor de 2, ese hijo ocuparía el doble de espacio que los demás (o lo intentara, al menos).

```
.hijo {
  flex-grow: 4;
}
```

flex-shrink

Define la capacidad de un hijo de reducir su tamaño si es necesario.

```
.hijo {
  flex-shrink: 3;
}
```

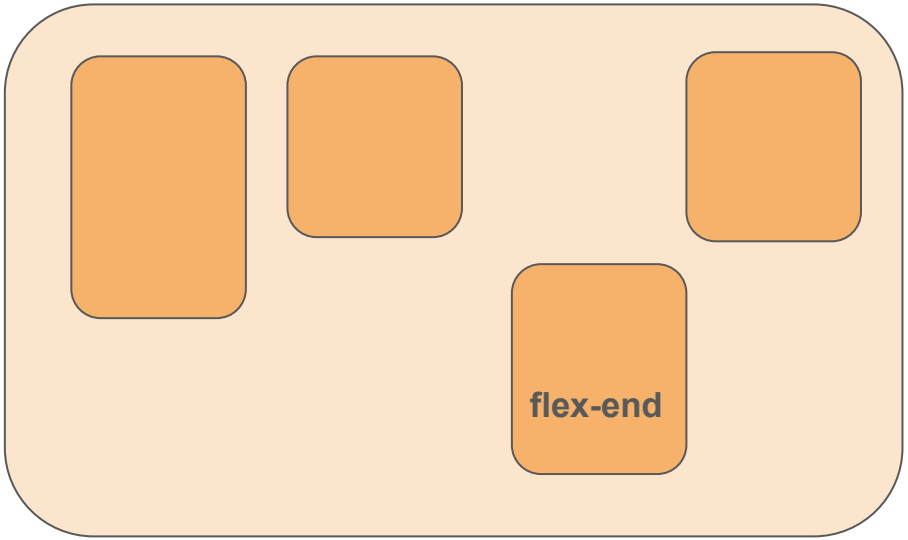
flex-basis

Define el tamaño predeterminado de un hijo antes de que se distribuya el espacio restante. Puede ser una longitud (por ejemplo, 20%, 5rem,etc) o una palabra clave(por ejemplo auto o max-content).

```
.hijo {
  flex-basis: 1rem;
}
```

align-self

flex-start



Permite anular la alineación predeterminada (o la especificada por align-items) individualmente por cada hijo.

```
.hijo {
  align-self: auto;
  align-self: flex-start;
  align-self: flex-end;
  align-self: center;
  align-self: baseline;
  align-self: stretch;
}
```

Aplicaciones de Flexbox en el diseño UX

Diseño de interfaces responsive

CSS Flexbox es una herramienta invaluable para crear interfaces que se adapten a diferentes dispositivos y tamaños de pantalla de manera fluida. Al permitir una distribución automática del espacio, los diseñadores pueden crear diseños que se ajusten sin problemas a cualquier resolución, mejorando así la experiencia del usuario en diversos dispositivos.

Diseño de interfaces intuitivas

También permite crear interfaces con elementos que se ajustan a las acciones del usuario, mejorando la usabilidad. A nivel de navegación, Flexbox también permite crear una navegación web que se adapta a diferentes tamaños de pantalla, mostrando u ocultando elementos según sea necesario.

Cuadrículas flexibles de contenido

Crear cuadrículas de contenido flexibles es una tarea común en el diseño web. Con Flexbox, los diseñadores pueden construir cuadrículas dinámicas que se ajusten automáticamente al contenido, garantizando una presentación visualmente atractiva y una experiencia de usuario fluida.

Animaciones y transiciones

Flexbox facilita la creación de animaciones y transiciones fluidas entre diferentes estados de la interfaz. El módulo permite definir con precisión la posición y el comportamiento de los elementos, lo que es fundamental para crear animaciones y transiciones fluidas y coherentes.

Ejemplos prácticos

Menús de navegación flexibles

Los menús de navegación son elementos fundamentales en la experiencia del usuario, Con Flexbox, es posible diseñar menús que se expandan o contraigan según sea necesario, optimizando el espacio disponible y facilitando la navegación en dispositivos móviles.

Diseño de formularios.

Los formularios son componentes esenciales en muchos sitios web y aplicaciones. Utilizando Flexbox, es posible diseñar formularios que se adapten a diferentes longitudes de campo y dispositivos, mejorando la usabilidad y accesibilidad para los usuarios.

Galerias de imagenes flexibles.

Las galerias de imagenes son otro ejemplo donde Flexbox brilla. Los diseñadores pueden crear galerías que se ajustan dinámicamente al tamaño de la pantalla, manteniendo la proporción de las imagenes y ofreciendo una experiencia visualmente agradable en cualquier dispositivo.

Tarjetas de contenido

Al utilizar Flexbox, las tarjetas de contenido pueden diseñarse para adaptarse fácilmente a diferentes tamaños de pantalla y dispositivos móviles, lo que mejora la experiencia del usuario en dispositivos móviles y tabletas También es posible alinear y espaciar los elementos dentro de las tarjetas de contenido de manera flexible y personalizada, lo que permite crear diseños visualmente atractivos y bien organizados.

Cuando usar Flexbox

Usos recomendados:

- Menús horizontales o verticales.
- Centrado de elementos.
- Cards o componentes individuales.
- Espacios distribuidos entre elementos.

Ejemplo:

```
.container {
display: flex;
justify-content: space-between;
align-items: center;
}
```

Cuando usar Grid

Usos recomendados:

- Layouts complejos.
- Tablas, galerías o dashboards.
- Responsive sin media queries explícitas.
- Control preciso de filas y columnas.

Ejemplo:

```
.container {
display: grid;
grid-template-columns:  1fr 2fr;
grid-template-rows: auto 1fr auto;
}
```

Diferencias clave

Características	Flexbox	Grid
Dimensión	Unidimensional	Bidimensional
Dirección principal	Eje principal	Filas y columnas
Control del layout	Por contenido	Por contenedor
Complejidad	Mas simple	Mas estructurado
Uso recomendado	Componentes	Estructura general

¿Qué es Grid?

Contenido:

- Diseño bidimensional: **filas y columnas**.
- Ideal para layouts completos y estructurados.
- Propiedades clave: `display: grid`, `grid-template-columns`, `grid-template-rows`, `grid-area`.
- Propiedades específicas: `grid-column-start, grid-column-end, grid-row-start,grid-row-end.`
- Abreviaturas: `grid-column:, grid-row:, grid-template:.`.

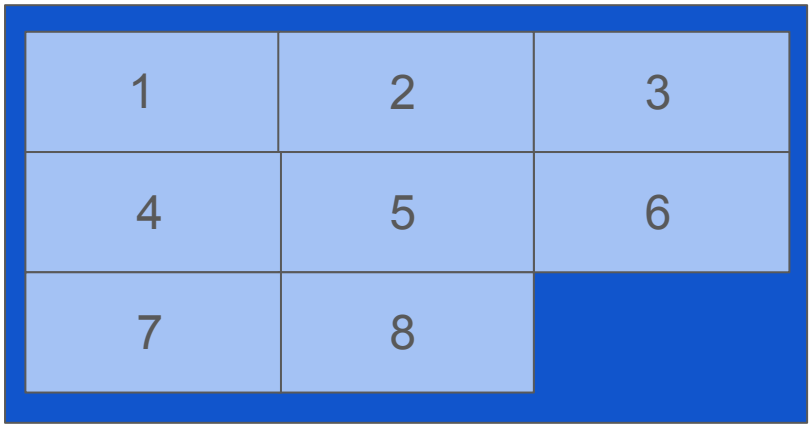
Contenedor de cuadrícula y elementos de cuadrícula

Un diseño de cuadrícula consta de un elemento principal (el contenedor de la cuadrícula) con uno o más elementos de cuadrícula.

Todos los elementos secundarios directos del contenedor de la cuadrícula se convierten automáticamente en elementos de la cuadrícula.

Ejemplo

```
<div class =”container”>
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
        <div>8</div>
</div>
```

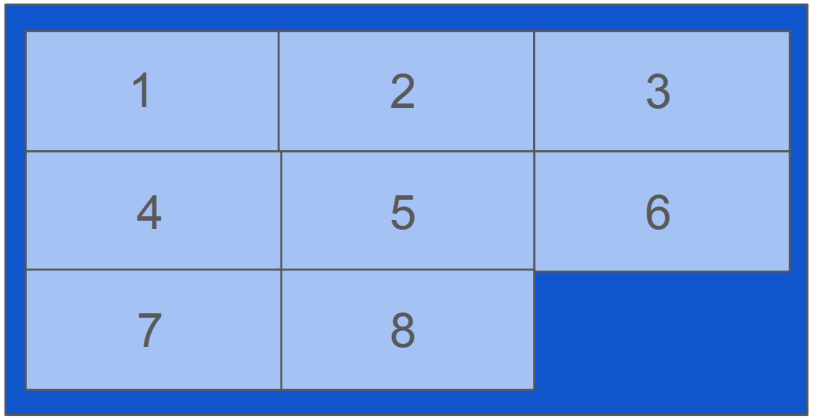


Propiedad de cuadrícula de visualización

El `<div>`elemento se convierte en un contenedor de cuadrícula cuando su `display` propiedad se establece en `grid` o `inline-grid`.

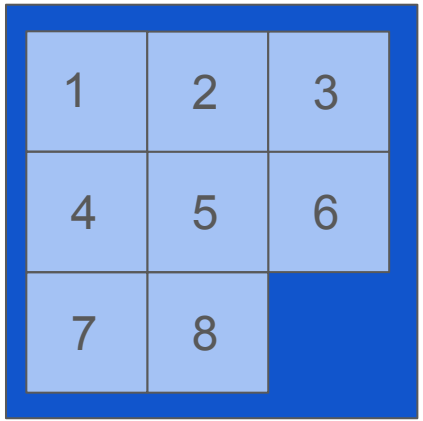
Ejemplo

```
.container {
        display: grid;
}
```



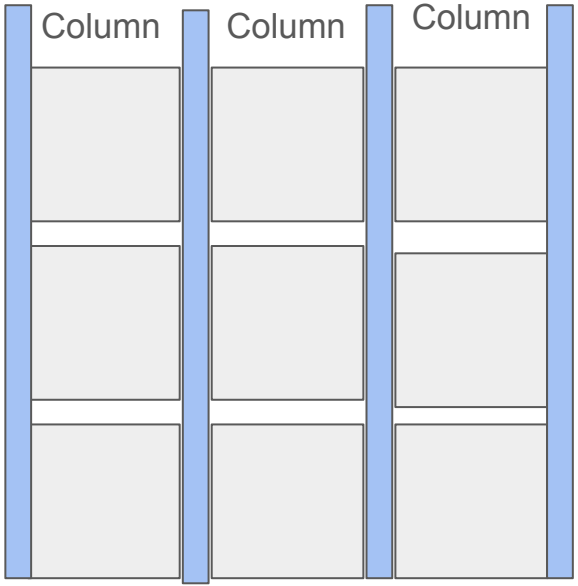
Ejemplo:

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



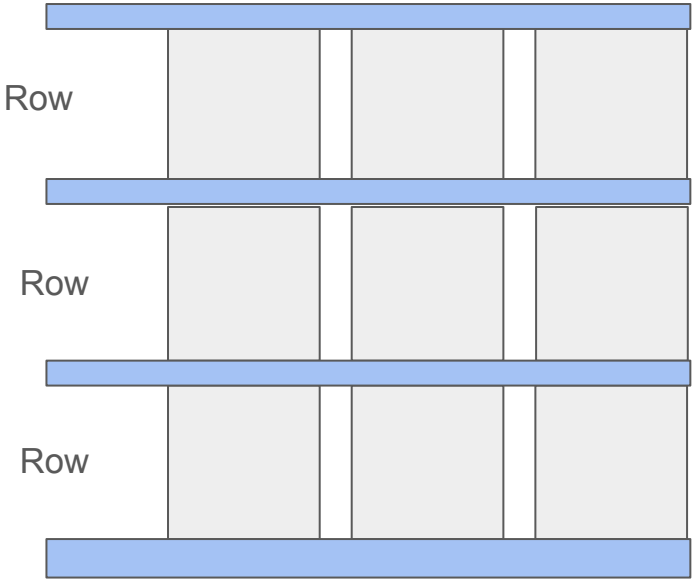
Grid Columns

Las líneas verticales de los elementos de la cuadrícula se denominan columns.



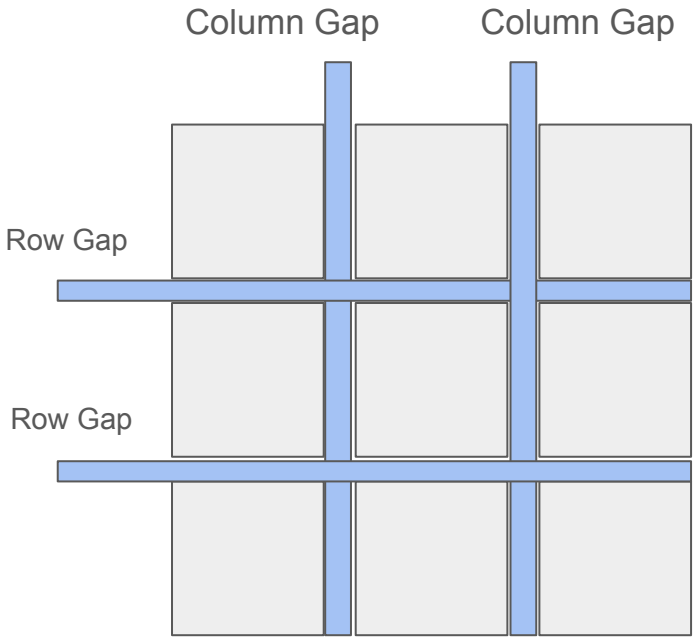
Grid Rows

Las líneas horizontales de los elementos de la cuadrícula se denominan rows.



Grid Gaps

Los espacios entre cada columna/fila se llaman Gaps.



Puede ajustar el tamaño del espacio utilizando una de las siguientes propiedades:

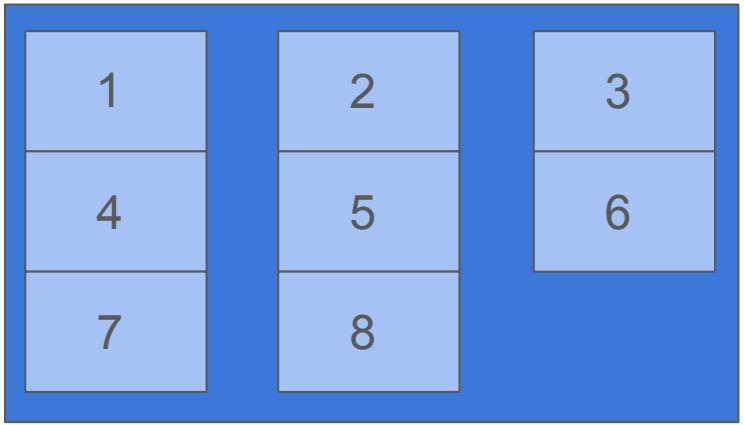
- Column-gap
- row-gap
- gap

La propiedad de column-gap

La column-gap propiedad especifica el espacio entre las columnas de una cuadrícula.

Ejemplo
Especifique un espacio de 50 píxeles
entre las columnas de la cuadrícula:

```
.container {

        display: grid;

        column-gap: 50px;

}
```



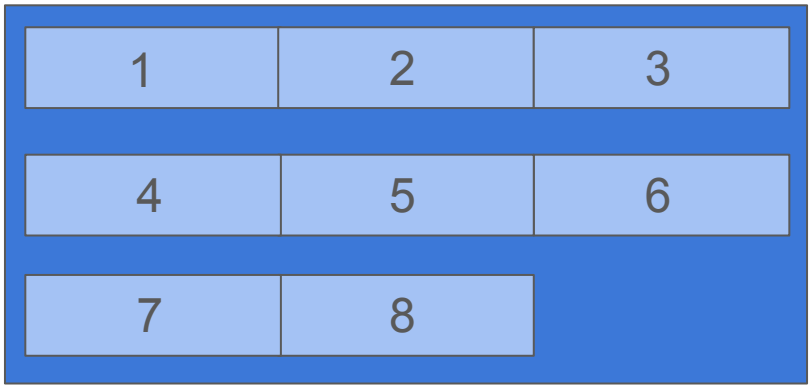
La propiedad de row-gap

La row-gap propiedad especifica el espacio entre las filas de una cuadrícula.

Ejemplo

Especifique un espacio de 50 píxeles entre las filas de la cuadrícula:

```
.container {
        display: grid;
        row-gap: 50px;
}
```



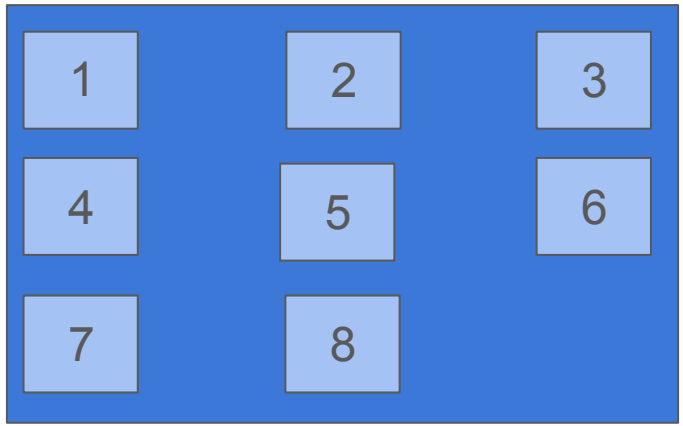
La propiedad de column-gap

La gap propiedad es una abreviatura de propiedad row-gap y column gap:

Ejemplo

Establezca el espacio entre filas en 50px y el espacio entre columnas en 100ps en la cuadricula:

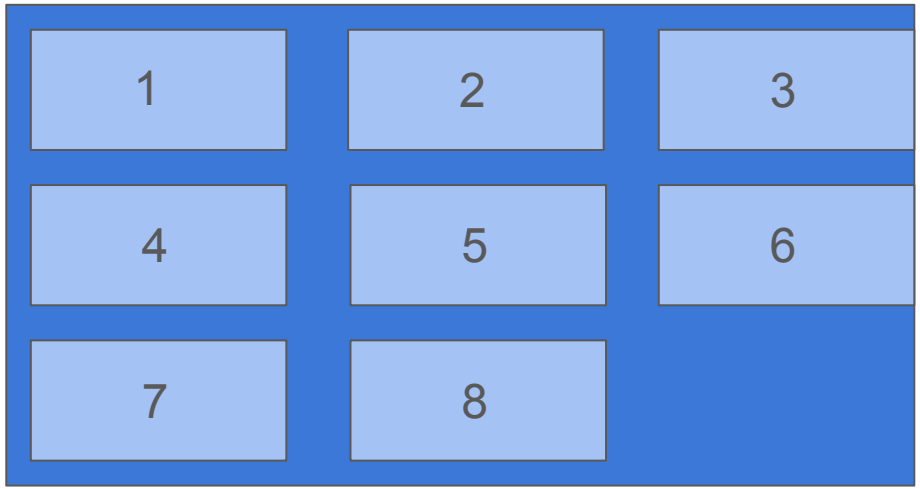
```
.container {
        display: grid;
        gap: 50px 100px;
}
```



Ejemplo

Especifique el espacio entre filas y columnas en 50 píxeles en la cuadrícula:

```
.container {

        display: grid;

        gap: 50px;

}
```

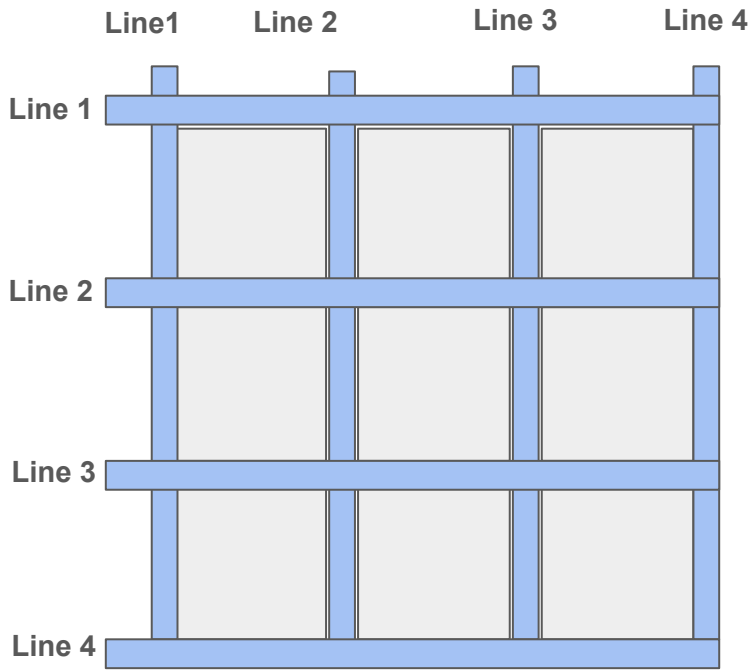


Grid Lines

Las líneas entre columnas se llaman column lines.
Las líneas entre filas se llaman rows lines

Column Lines

Row lines



Podemos especificar dónde comenzar y finalizar un elemento de la cuadrícula utilizando las siguientes propiedades:

- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end
- grid-column
- grid-row

Puede hacer referencia a los numeros de linea al colocar un elemento de la cuadrícula en un contenedor de la cuadrícula.

Las propiedades de grid-column-start y grid-column-end

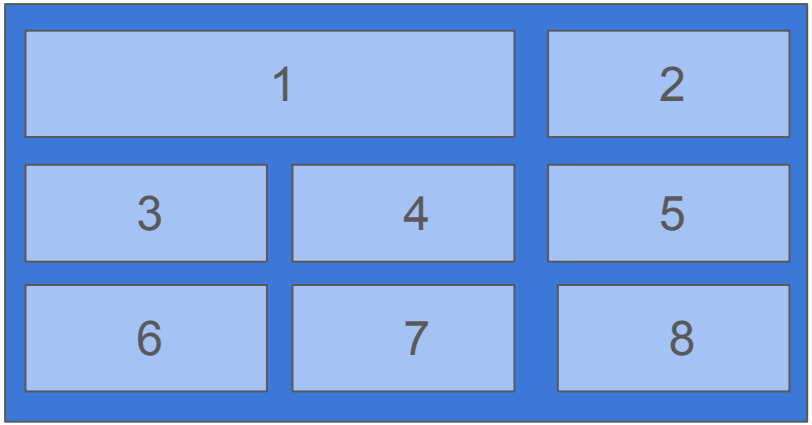
La grid-column-start propiedad específica donde iniciar un elemento de la cuadrícula.

La grid-column-end propiedad específica donde finalizar un elemento de la cuadrícula.

Ejemplo

Coloque el primer elemento de la cuadrícula en la línea de columna 1 y deje que finalice en la línea de columna 3:

```
.item1 {
        grid-column-start: 1;
        grid-column-end: 3;
}
```



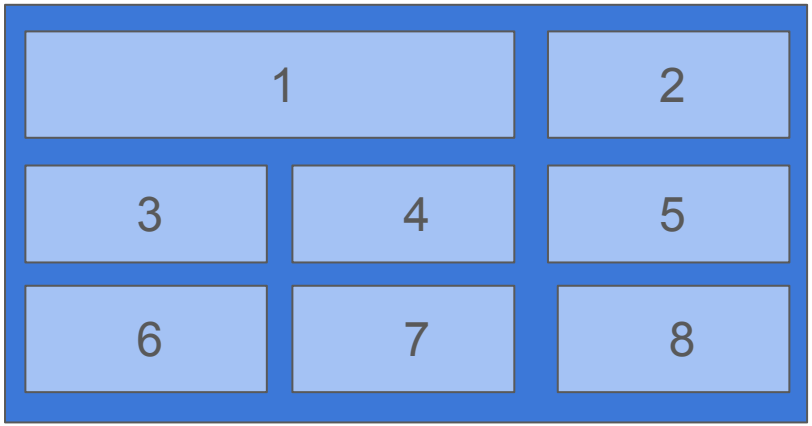
La propiedad grid-column

La grid-column propiedad es una forma abreviada de propiedad para las grid-column-start y las grid-column-end propiedades.

Ejemplo

Coloque el primer elemento de la cuadrícula en la línea de columna 1 y deje que abarque 2 columnas:

```
.item1 {
        grid-column: 1 / span 2;
}
```



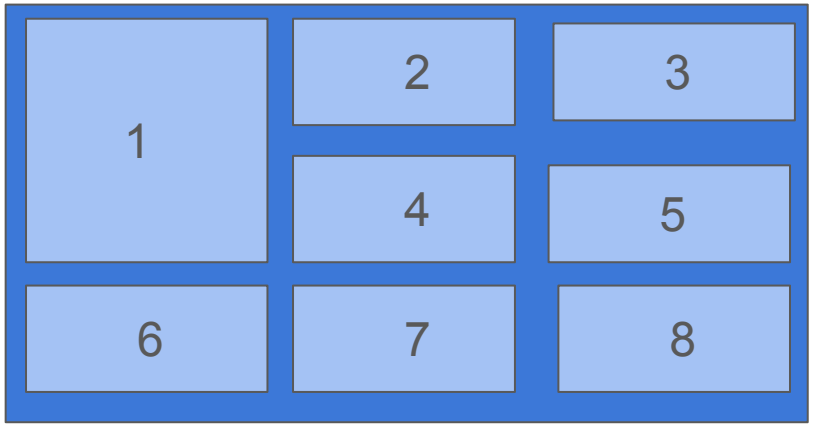
Las propiedades grid-row-star y grid-row-end

La grid-row-start propiedad específica donde iniciar un elemento de la cuadrícula.
La grid-row-end propiedad específica donde finalizar un elemento de la cuadricula.

Ejemplo

Coloque el primer elemento de la cuadrícula en la línea de fila 1 y déjelo terminar en la línea de fila 3:

```
.item1 {
        grid-row-star: 1 ;
        grid-row-end: 3;
}
```



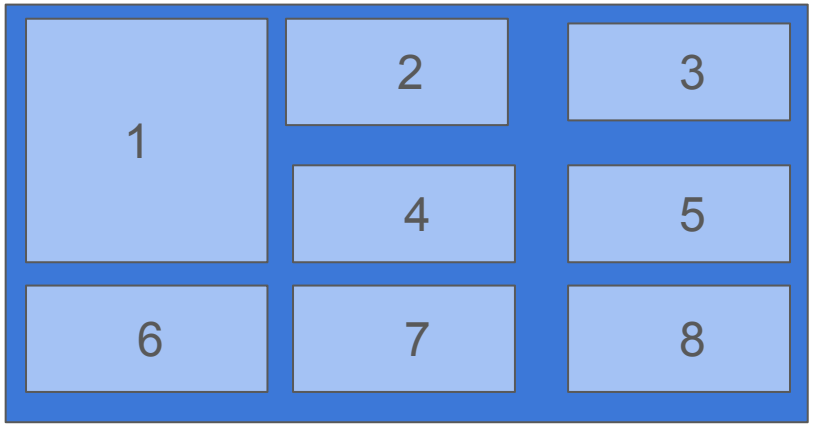
Las propiedades grid-row

La grid-row propiedad es una forma abreviada de la propiedad para las grid-row-start y las grid-row-end propiedades.

Ejemplo

Coloque el primer elemento de la cuadrícula en la línea de fila 1 y deje que abarque 2 filas:

```
.item1 {
        grid-row: 1 / span 2 ;
}
```



Grid Container

Un contenedor de cuadrícula contiene uno o más elementos de cuadrícula organizados en columnas y filas.

Los elementos secundarios directos del contenedor de la cuadrícula se convierten automáticamente en elementos de la cuadrícula.

un elemento se convierte en un contenedor de cuadrícula cuando su display propiedad se establece en grid o inline-grid.

Grid Tracks

Las filas y columnas de una cuadrícula se definen con las grid-template-rows propiedades y grid-template-columns(o la abreviatura grid o grid-template propiedades).

Estos definen las pistas de la cuadrícula. una posta de la cuadrícula es el espacio entre dos líneas de cuadrícula adyacentes.

La propiedad grid-template-columns

La grid-template-columns propiedad define la cantidad de columnas en el diseño de la cuadrícula y puede definir el ancho de cada columna.

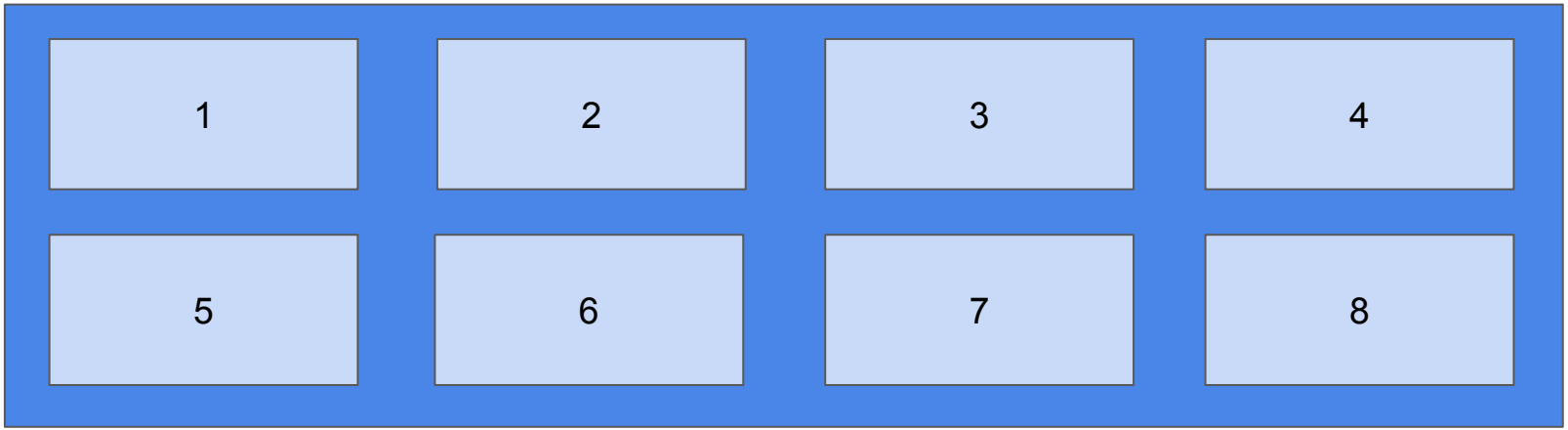
El valor es una lista separada por espacios, donde cada valor define el ancho de la columna respectiva.

Si desea que el diseño de su cuadrícula contiene 4 columnas, especifique el ancho de las 4 columnas o “automático” si todas las columnas deben tener el mismo ancho.

Ejemplo

Haz una cuadrícula con 4 columnas de igual ancho:

```
.grid-container {

        display: grid;

        grid-template-columns: auto auto auto auto;
}
```

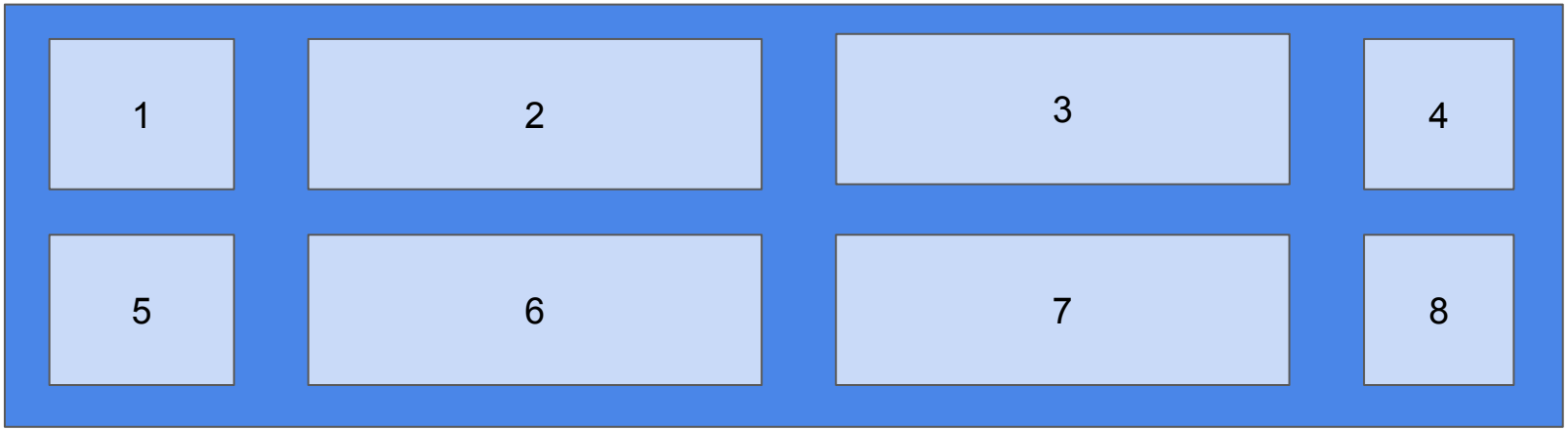


La grid-template-columns propiedad también se puede utilizar para especificar el tamaño exacto (ancho) de las columnas.. o una combinación de tamaño fijo y automático.

Ejemplo
Haz una cuadrícula con 4 columnas de igual ancho:

```
.grid-container {

    display: grid;

    grid-template-columns: 80px 200px auto 40px;
}
```

NOTA:
Si tiene más de 4 elementos de cuadrícula en una cuadrícula de 4 columnas, la cuadrícula agregara automáticamente una nueva fila para colocar los elementos.



Dimensionamiento de celda con la unidad fr

La fr unidad se puede utilizar al definir cuadrículas, como cualquier otra longitud CSS como %, px o em.

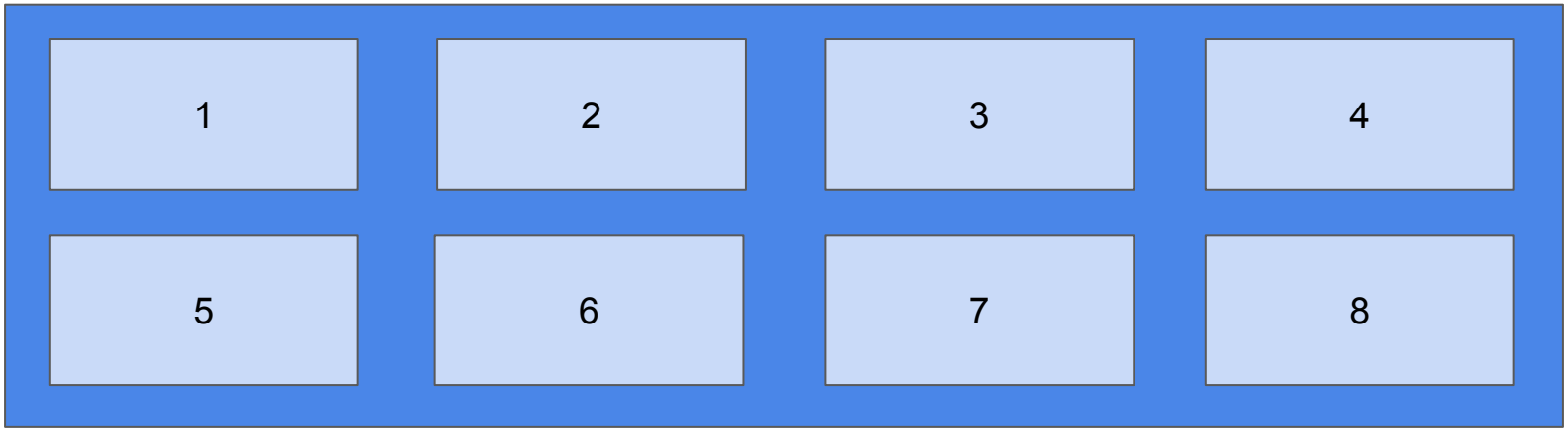
La fr unidad significa “fracción”. Esta unidad divide automáticamente el espacio disponible en fracciones.

Ejemplo: 1 fr tomará 1 fracción del espacio disponible, mientras que 2 fr tomara 2 fracciones del espacio disponible.

Ejemplo

Aquí, cada columna ocupará el 25% del ancho del contenedor, dividiéndolo en partes iguales:

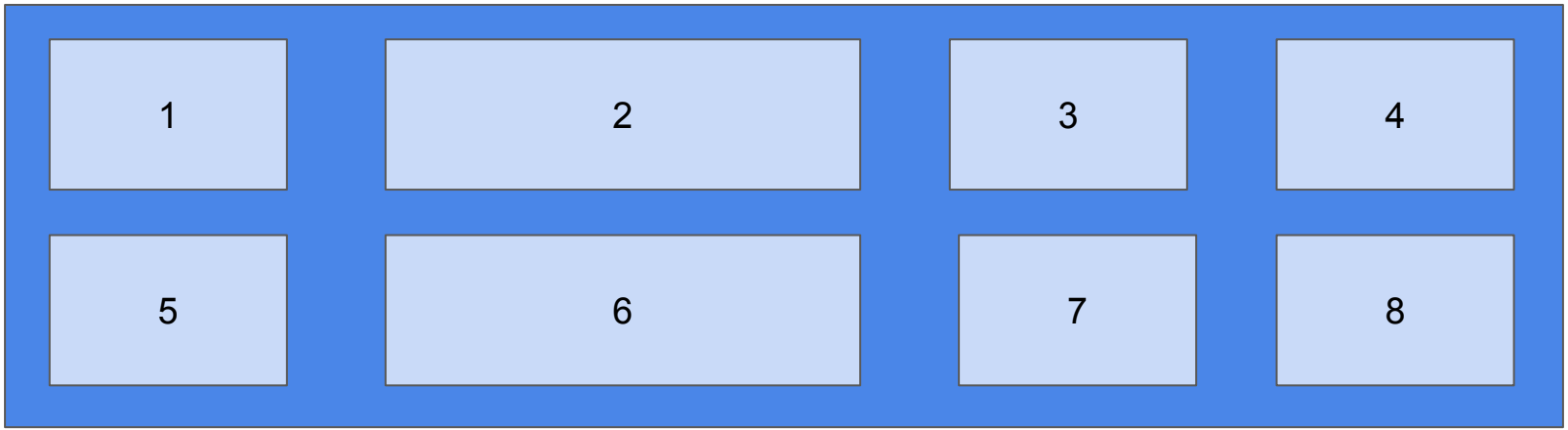
```
.grid-container {
        display: grid;
        grid-template-columns: 1fr 1fr 1fr 1fr;
}
```



Ejemplo

Aquí, la segunda column será el doble de grande que las demás:

```
.grid-container {
        display: grid;
        grid-template-columns: 1fr 2fr 1fr 1fr;
}
```



La propiedad grid-template-rows

La grid-template-rows propiedad define la altura de cada fila-

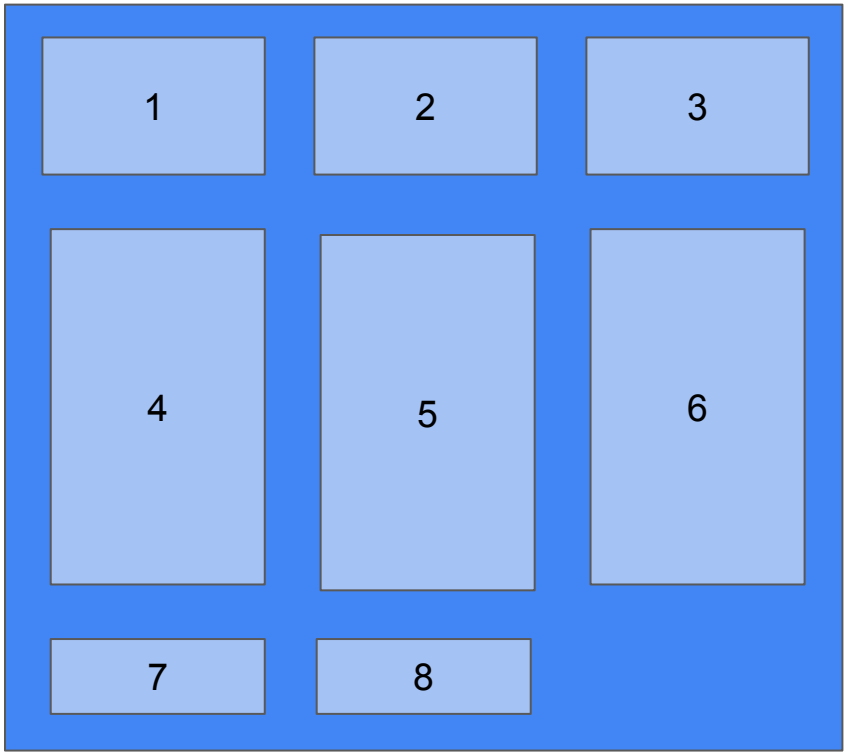
El valor es una lista separada por espacios, donde cada valor defina la altura de la fila respectiva:

Ejemplo

Aqui, la segunda column será el doble de grande que las demás:

```
.grid-container {
        display: grid;

        grid-template-rows: 80px 200px;
}
```

Observe que la primera fila de la cuadrícula anterior tiene una altura de 80px y la segunda de 200px. Las siguientes filas usarán la opción “auto”como predeterminada.



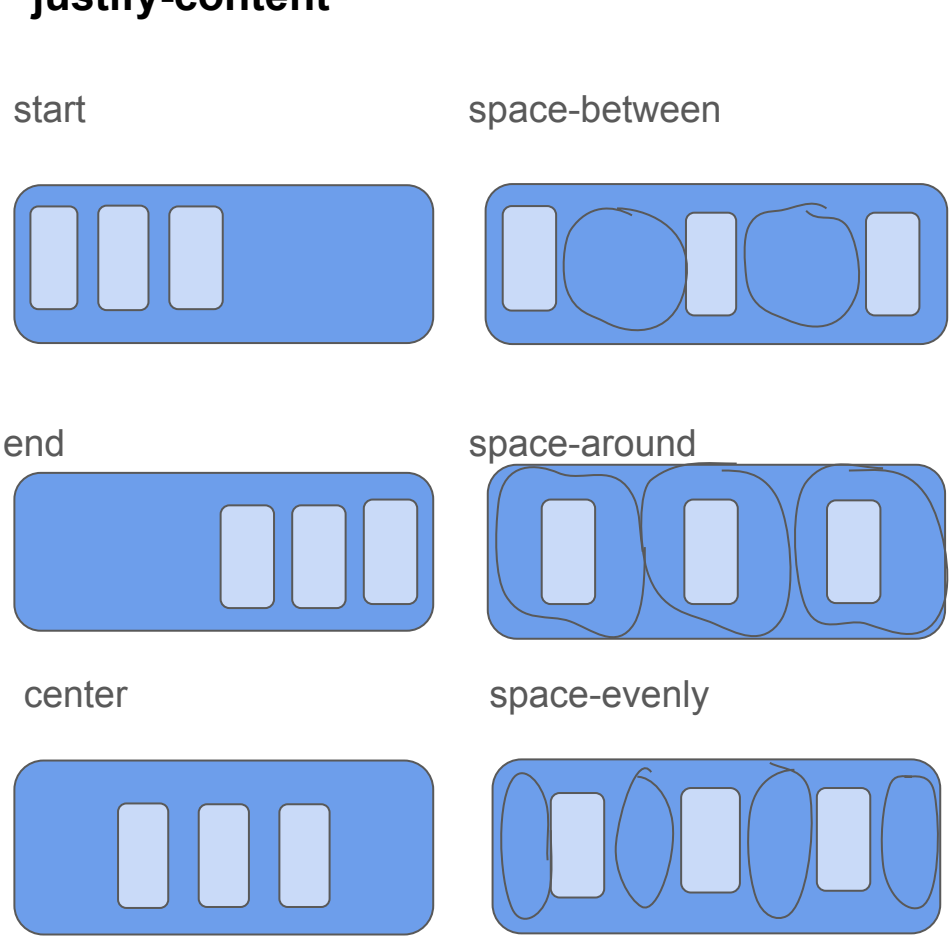
La propiedad Justify-content

Esta justify-content propiedad se utiliza para alinear los elementos de la cuadrícula cuando no utilizan todo el espacio disponible en el eje principal (horizontalmente).

La justify-content propiedad puede tener uno de los siguientes valores:

- start
- end
- center
- space-between
- space-around
- space-evenly

Nota: El ancho total del elemento de la cuadrícula debe ser menor que el ancho del contenedor para que la justify-content propiedad tenga algún efecto,

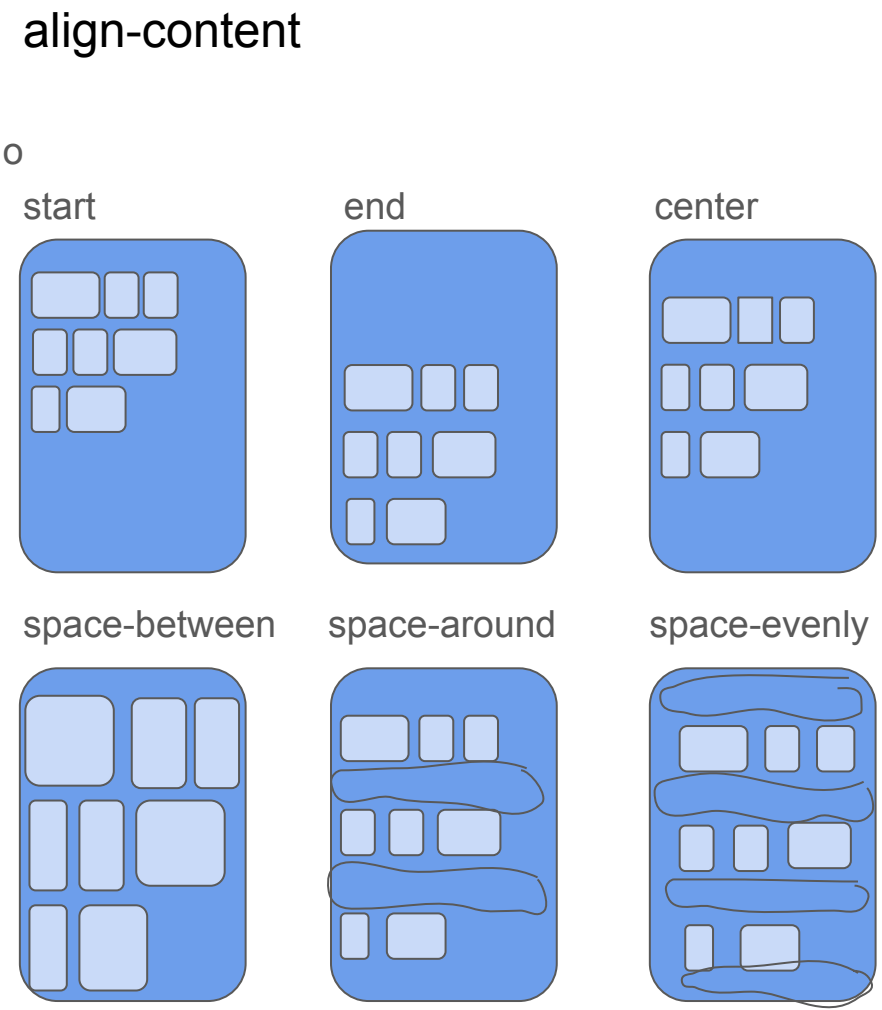


La propiedad align-content

Esta align-content propiedad se utiliza para alinear los elementos de la cuadrícula cuando no utilizan todo el espacio disponible en el eje transversal (verticalmente).

La align-content propiedad puede tener uno de los siguientes valores:

- start
- end
- center
- space-evenly
- space-between
- space-around



La propiedad place-content

La place-content propiedad es una forma abreviada de propiedad para las align-content

y justify-content las propiedades.

Si la place-content propiedad tienen dos valores:

- place-content: start center; -el align-content valor es “inicio” y justify-content el valor es “centro”

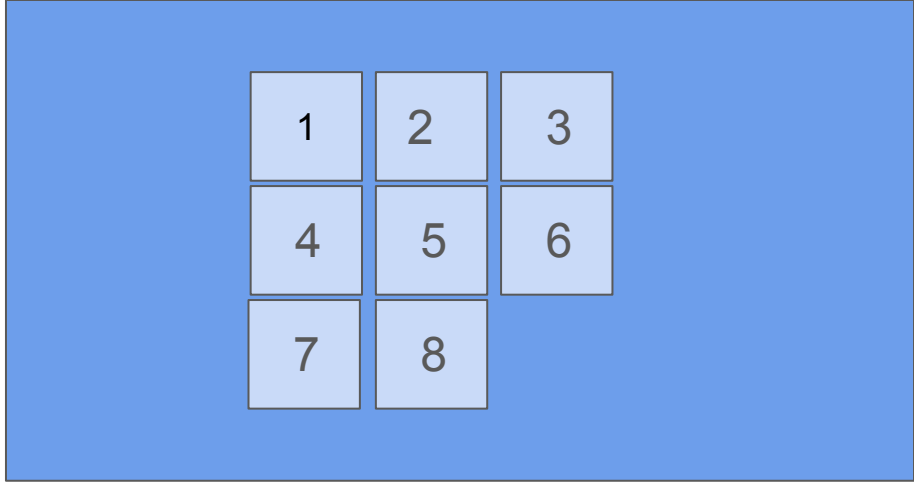
Si la place-content propiedad tienen un valor:

- place-content: end; - Ambos align-content valores justify-content son “fin”

Nota: La altura y el ancho totales del elemento de la cuadrícula deben ser menores que la altura y el ancho del contenedor para que la place-content propiedad tenga algún efecto.

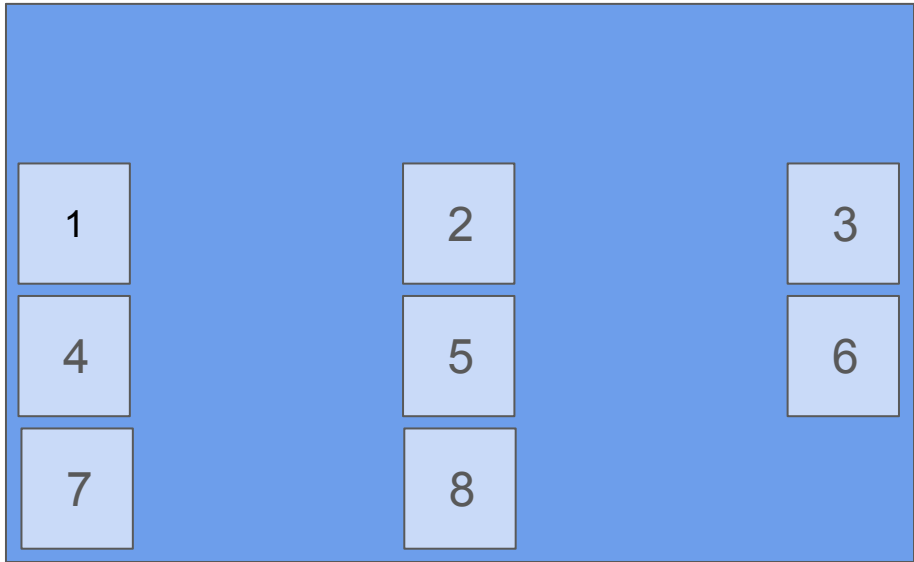
El center valor posiciona los elementos de la cuadrícula en el medio del contenedor (tanto vertical como horizontalmente).

```
.grid-container {
  display: grid;
  height: 400px;
  place-content: center;
```



El end space.between valor alinea las líneas de la cuadrícula hacia la parte inferior del contenedor de la cuadrícula y alinea los elementos de la cuadrícula con el mismo espacio entre ellos horizontalmente.

```
.grid-container {
  display: grid;
  height: 400px;
  place-content: end space-between;
}
```



La propiedad del área de cuadrícula

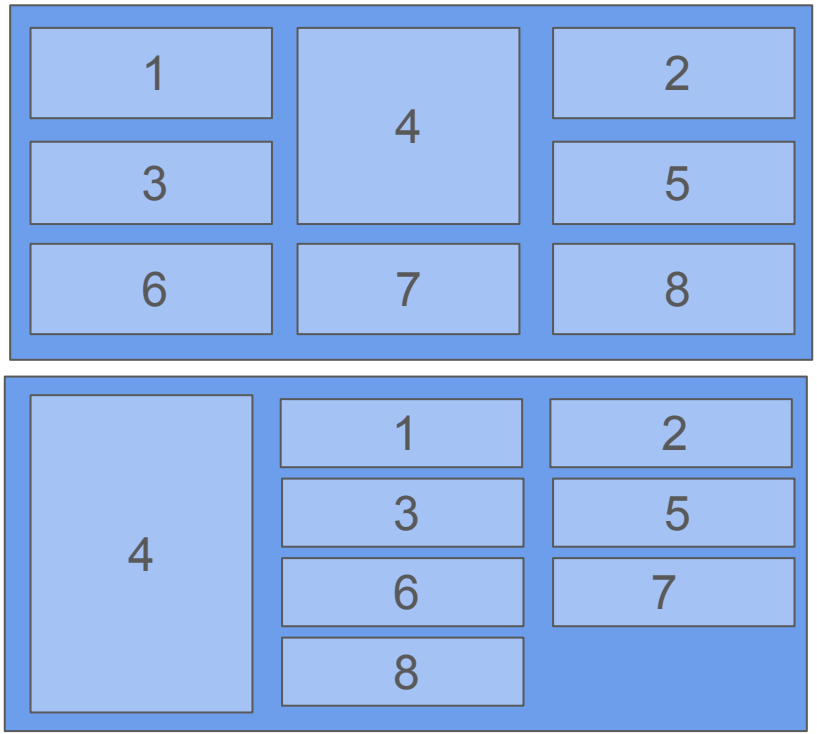
La grid-area propiedad es una abreviatura de propiedad para las propiedades grid-row-start, grid-column-start y grid-row-end, grid-column-end.
La sintaxis es cuadricula-fila-inicio / cuadrícula-columna-inicio / cuadrícula-fila-fin / cuadrícula-columna-fin.
grid-row-start / grid-column-start / grid-row-end / grid-column-end

Ejemplo
Haga que el “ítem 4” comience en la fila-linea 1 y la columna-linea 2, y termine en la fila-linea 3 y la columna-línea 2:

```
.item4 {
  grid-area: 1 / 2 / 3 / 2;
}
```

Hacer que “item 4” comience en la fila-línea 1 y en la columna-línea 1, y abarque 4 filas y 1 columna.

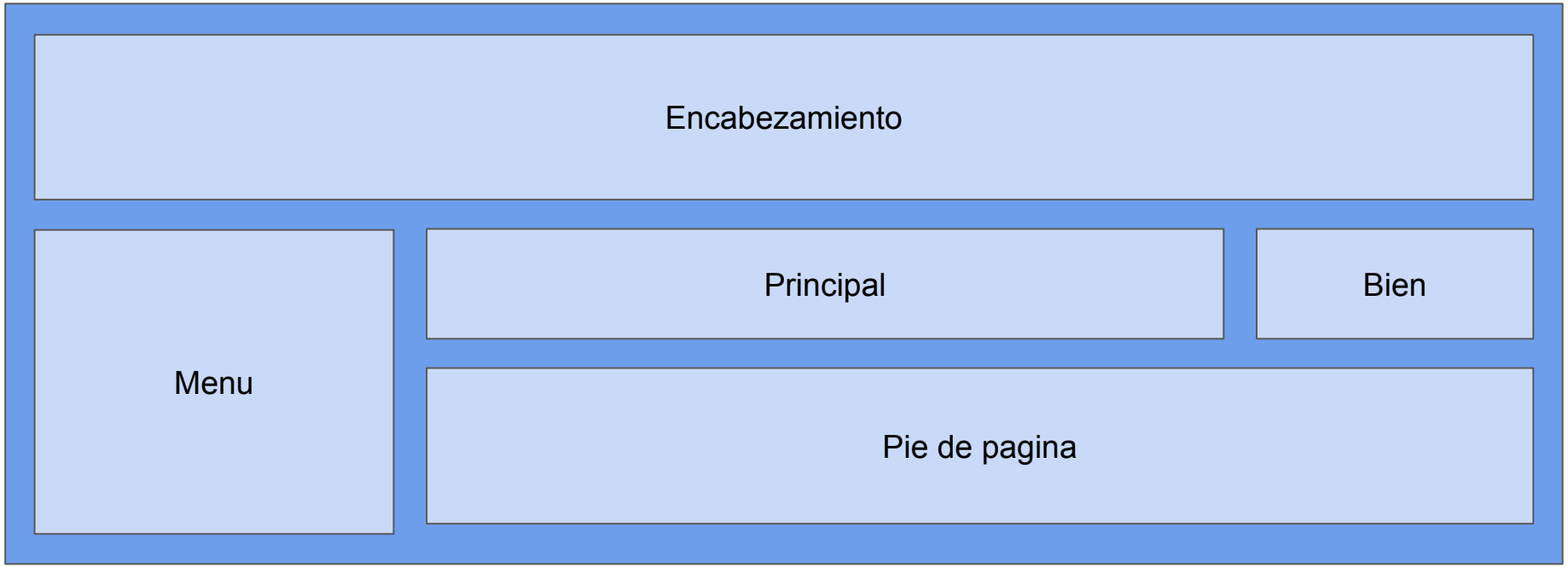
```
.item4 {
  grid-area: 1 / 1 / span 4 / span 1;
}
```



Nombrar elementos de la cuadrícula con grid-area

La grid-area propiedad también se puede utilizar para asignar nombres a los elementos de la cuadrícula.

Luego se puede hacer referencia a los elementos de la cuadrícula nombrados mediante la grid-template-areas propiedad del contenedor de la cuadrícula.

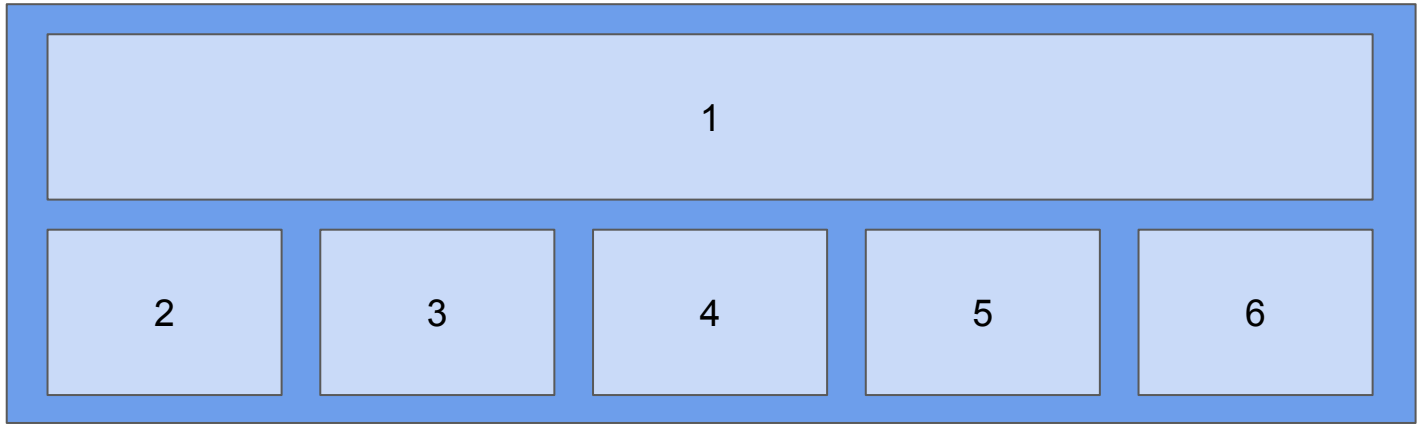


Ejemplo

El elemento 1 recibe el nombre “myArea” y abarca las cinco columnas en un diseño de cuadrícula de cinco columnas:

```
.item1 {
  grid-area: myArea;
}
.grid-container {
  grid-template-areas: 'myArea myArea myArea myArea myArea';
}
```

Cada fila está definida por apóstrofos (‘ ’). Los elementos de la cuadrícula nombrados en cada fila se definen dentro de los apostrofes, separados por un espacio.

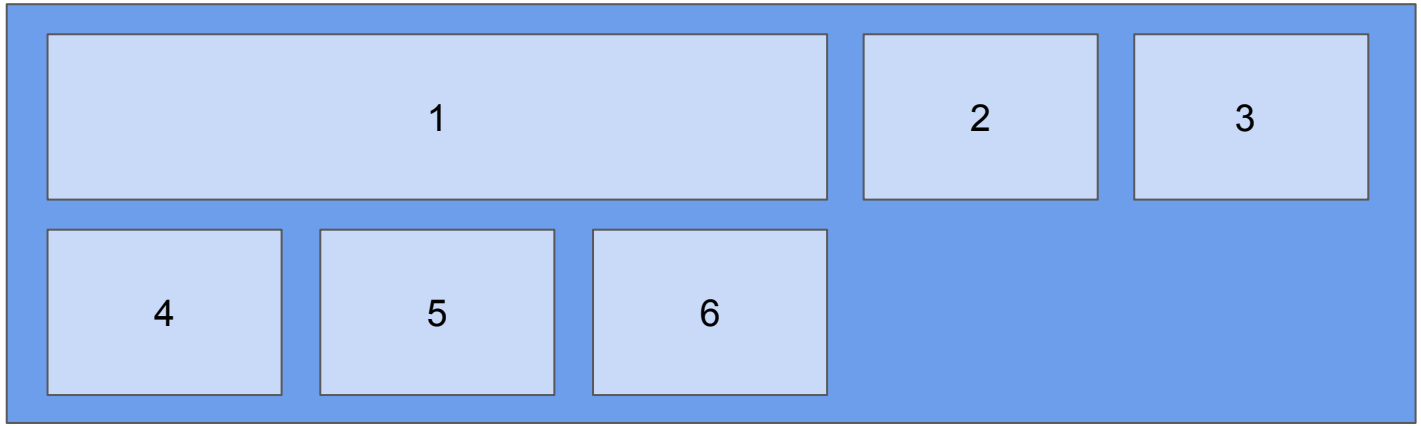


Ejemplo

Deje que “myArea” abarque tres columnas en un diseño de cuadrícula de cinco columnas (los signos de punto representan elementos sin nombre):

```
.item1 {
  grid-area: myArea;
}

.grid-container {
  grid-template-areas: 'myArea myArea myArea . .';
}
```

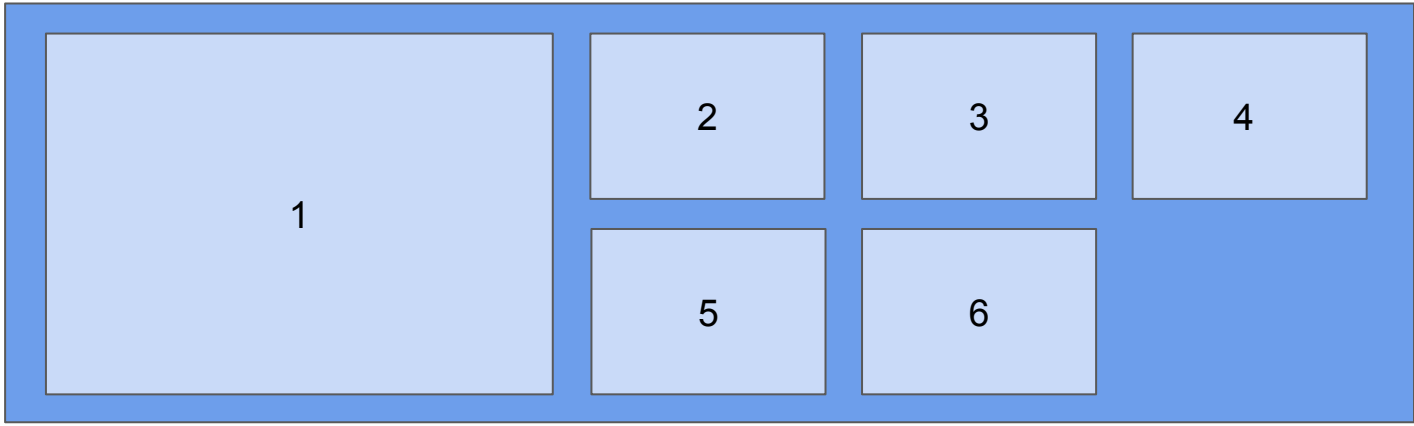
Nota: Un signo de punto representa un elemento de la cuadrícula sin nombre.



Ejemplo

Deje que el “item1” abarque dos columnas y dos filas:

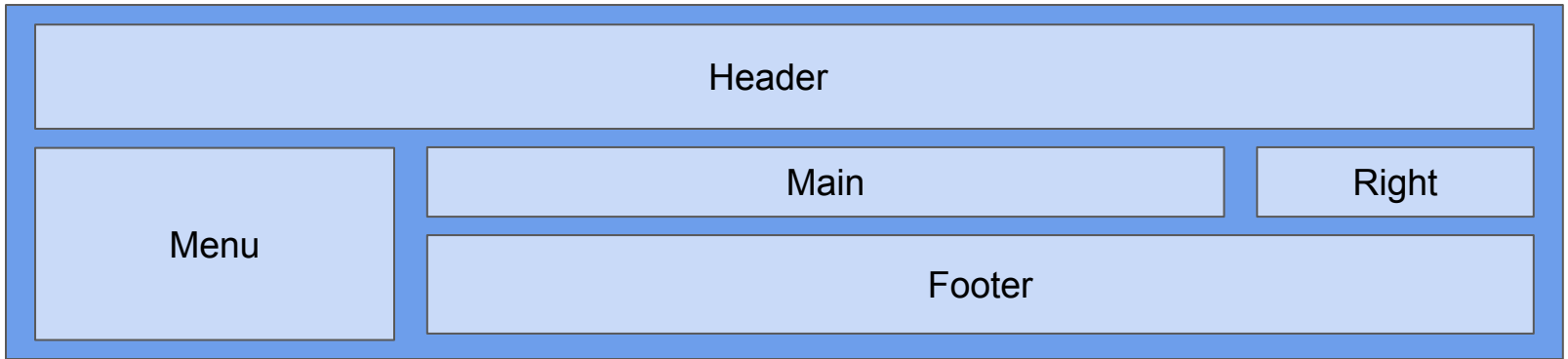
```
.item1 {
  grid-area: myArea;
}

.grid-container {
  grid-template-areas:
        'myArea myArea . . .'
        'myArea myArea . . .';
}
```



Ejemplo

Nombra todos los elementos de la cuadrícula y crea una plantilla de página web lista para usar:

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  grid-template-areas:
        'header header header header header header'
        'menu main main main main right'
        'menu footer footer footer footer footer';
```

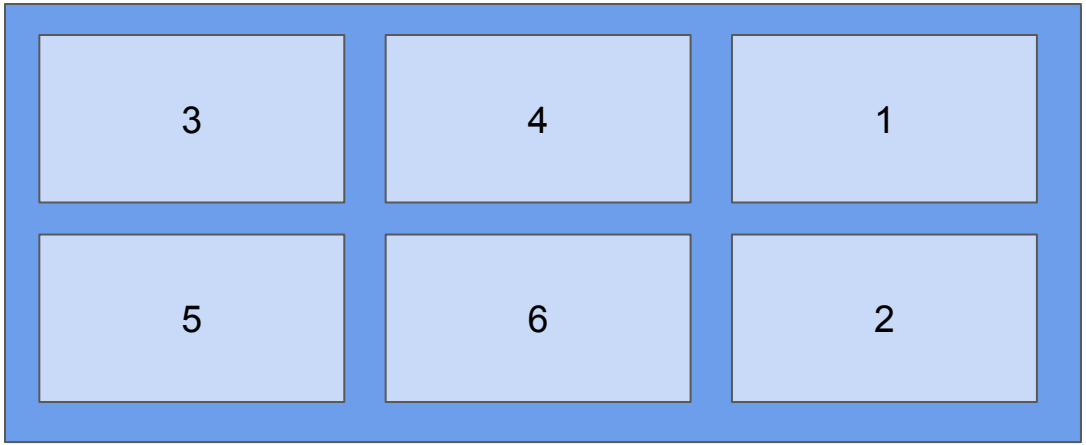


El orden de los elementos de la cuadricula

La grid-area propiedad también se puede utilizar para definir el orden de los elementos de la cuadrícula.

El primer elemento de la cuadrícula en el código HTML no tiene que aparecer como el primer elemento de la cuadrícula.

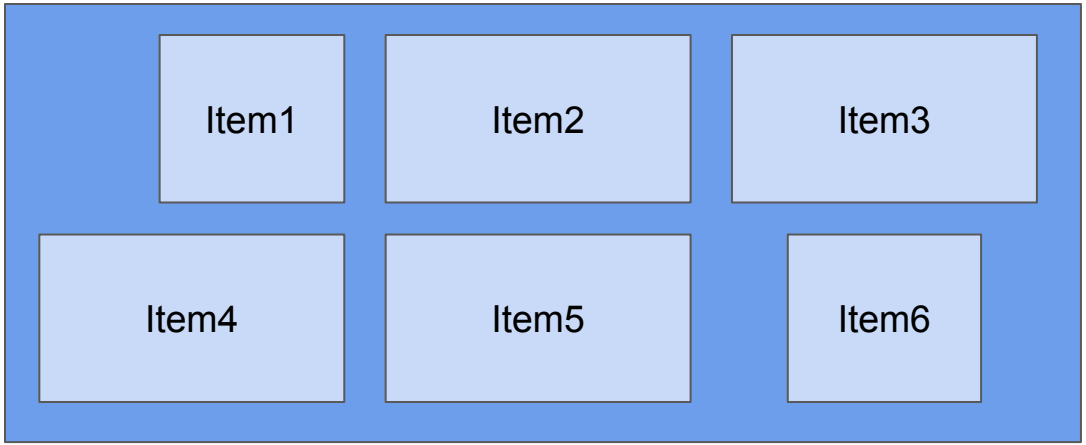
```
/* place in row 1 column 3 */
.item1 {grid-area: 1 / 3;}

/* place in row 2 column 3 */
.item2 {grid-area: 2 / 3;}

/* place in row 1 column 1 */
.item3 {grid-area: 1 / 1;}

/* place in row 1 column 2 */
.item4 {grid-area: 1 / 2;}

/* place in row 2 column 1 */
.item5 {grid-area: 2 / 1;}

/* place in row 2 column 2 */
.item6 {grid-area: 2 / 2;}
```



La propiedad de justificarse a sí mismo

La justify-self propiedad se utiliza para alinear el contenido de un elemento de la cuadrícula a lo largo del eje de la fila.

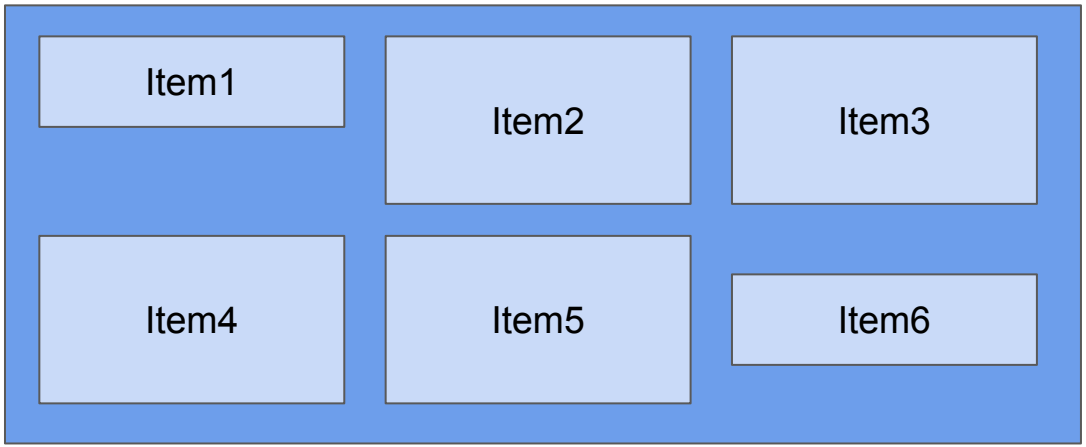
```
.item1 {
  justify-self: right;
}

.item6 {
  justify-self: center;
}
```



La propiedad align-self

La align-self propiedad se utiliza para alinear el contenido de un elemento de la cuadrícula a lo largo del eje de la columna.

```
.item1 {
  align-self: start;
}

.item6 {
  align-self: center;
}
```



THE END