Grillas

Módulo 04

Grillas

Al trabajar con **flex,** este era el valor que inicialmente colocamos en la propiedad display de nuestro contenedor principal. De la misma forma, el trabajo con **grillas** comienza reemplazando este valor por el de **grid.**

```
div { display: grid;}
```

No debemos, reemplazar el trabajo con **flex,** ni siquiera con **float.**

La intención de este apartado es comprender las potencialidades que nos brinda esta propiedad utilizada de forma complementaria en nuestra interfaz para lograr maquetar el diseño prototipado.

Grillas: comenzar a trabajar

Al implementar **grid** en el contenedor, por ejemplo en la siguiente estructura, no veríamos grandes cambios,

```
<div>

  <p>1 parrafo</p>
  <p>2 parrafo</p>
  <p>3 parrafo</p>
  <p>4 parrafo</p>

  </div>
```

Necesitamos implementar más propiedades para empezar a trabajar. Veamos un ejemplo de utilización simple y los resultados que obtenemos.

Primeras propiedades

En un sistema de **grillas es fundamental, entender qué tenemos filas y columnas.** El trabajo para determinar cuántas columnas tendremos es **grid-template-columns.**

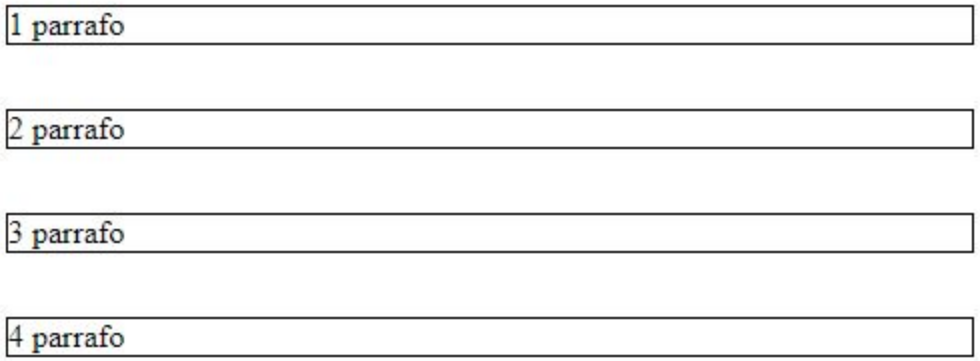
Esta propiedad **toma valores de longitud así como también la palabra auto para adaptarse al espacio disponible.**

También es importante saber cuántos elementos tenemos para comprender el resultado final. En la estructura anterior, **un formato de 4 elementos, con los siguientes estilos:**

```
div { display: grid; }

p { border: 1px solid black;}
```

Primeros resultados

No dará el siguiente resultado. Este resultado no es del todo prometedor y no implica demasiados cambios,



Trabajo con columnas

Sin embargo si implementamos la propiedad **grid-template-columns empezamos a notar cómo se genera un sistema de grilla como el esperado,**

```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;   }

p { border: 1px solid black;}
```

Dependiendo cuantas medidas indiquemos serán la cantidad de columnas que se trabajarán,

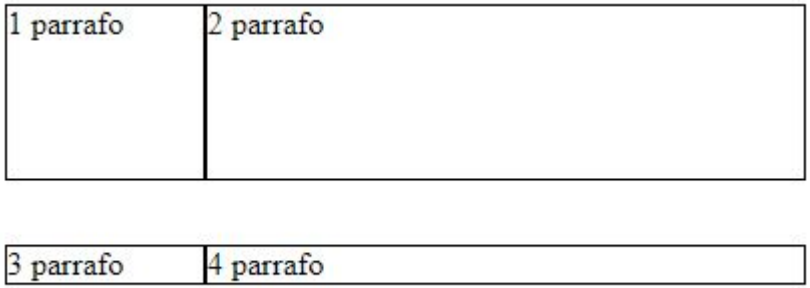
1 parrafo	2 parrafo

3 parrafo	4 parrafo

Trabajo con filas

El trabajo con filas, es similar. Lo realizamos a través de la propiedad **grid-template-rows.**

Esta propiedad indica el alto de las filas, luego de ya determinadas las columnas con la propiedad **grid-template-column.**




```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;    }

p { border: 1px solid black;}
```

Generar espacios en las grillas

Para generar espacios trabajamos con **column-gap o row-gap.**

La propiedad column-gap representa el espacio **entre las columnas. La propiedad row-gap,** por otro lado, representa el **espacio entre las filas.**

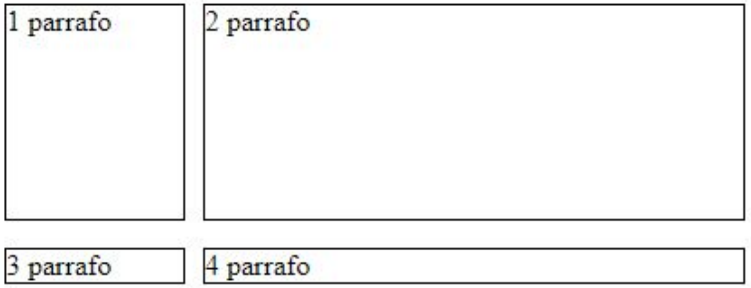
```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    column-gap:10px;
    row-gap:5px;
     }
```

Generar espacios en las grillas: una línea

Para generar espacios trabajamos con **column-gap o row-gap.** De todas maneras, siempre es interesante saber cómo trabajar de forma más ágil y sencilla.

La propiedad grid-gap (actualmente reemplazada por la propiedad gap) nos permite en una línea indicar el **column-gap y el row-gap. Siempre el primer valor responde a las filas y el segundo a las columnas,** por tanto el ejemplo anterior en una sóla línea sería equivalente a:

```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    gap: 5px 10px;
     }

p { border: 1px solid black;
    margin: 0; }
```



Alineación dentro de la grilla

Para alinear nuestro contenido dentro de la grilla, debemos trabajar con **justify content.** Esta propiedad contiene valores gracias a los cuales podemos orientar nuestro contenido hacia el **comienzo del contenedor, gracias al valor start:**

```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    grid-column-gap:10px;
    justify-content: start; }

p { border: 1px solid ■black;}
```

O hacia el final del mismo, donde trabajamos con **end** como vemos en la imagen de la derecha.

```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    grid-column-gap:10px;
    justify-content: end; }
```

Alineación dentro de la grilla

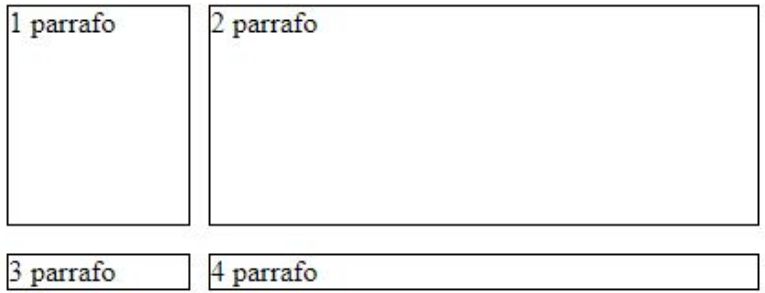
Para alinear nuestro contenido también contamos con valores como **center**. Este permite centrar el contenido como así también podemos trabajar con **space-evenly** para espacios iguales entre los elementos contenidos:

```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    gap: 15px 10px;
    justify-content: space-evenly;
     }

p { border: 1px solid black;
    margin: 0; }
```

Alineación dentro de la grilla: valores

Con la intención de generar espacios antes y después de los elementos podemos trabajar con el valor **space-around**. También existe **between** donde sólo generamos espacio entre los mismos:

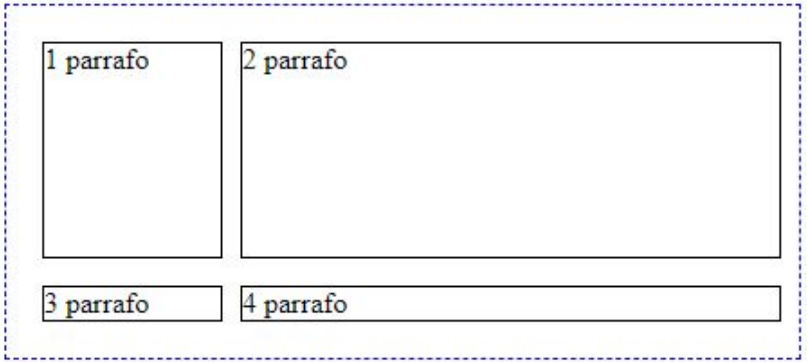
```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    gap: 15px 10px;
    justify-content: space-between;
     }

p { border: 1px solid black;
    margin: 0; }
```



Alineación dentro de la grilla: vertical

La propiedad de alineación vertical es **align-content.** Si por ejemplo queremos centrar el contenido trabajamos con **center**,

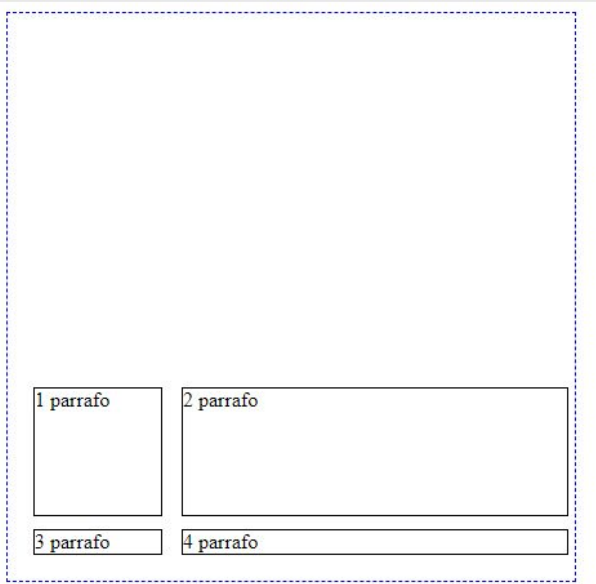
```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    gap: 15px 10px;
    justify-content: space-between;
    align-content: center;
    border: 1px dashed blue;
    padding: 20px;

    }

p { border: 1px solid black;
    margin: 0; }
```



Alineación dentro de la grilla: valores

Otros valores son similares a los utilizados en **justify-content.** Al momento de trabajar se debe notar, como elementos como el alto del contenedor, y el **padding generan y acompañan la vista final,**

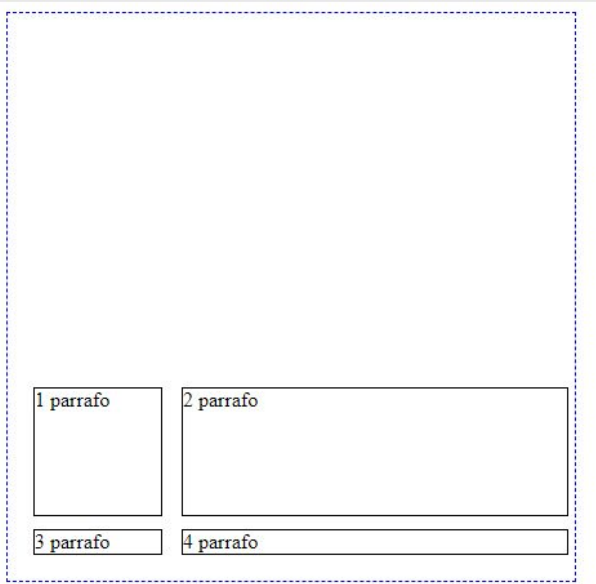
```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:100px;
   padding: 10px;
   gap: 10px 15px;
    justify-content: space-between;
    align-content:end;
    border: 1px dashed blue;
    height: 400px;
    padding: 20px;


    }

p { border: 1px solid black;
    margin: 0; }
```



Alineación dentro de la grilla: valores

Otros valores son similares a los utilizados en **justify-content.** Al momento de trabajar se debe notar, como elementos como el alto del contenedor, y el **padding generan y acompañan la vista final,**

```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:100px;
   padding: 10px;
   gap: 10px 15px;
    justify-content: space-between;
    align-content:end;
    border: 1px dashed blue;
    height: 400px;
    padding: 20px;


    }

p { border: 1px solid black;
    margin: 0; }
```

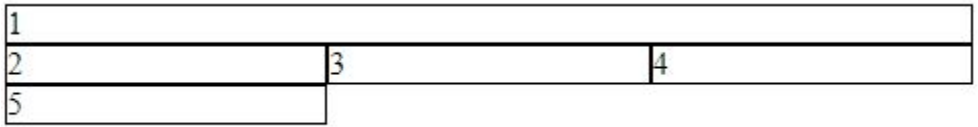


Expandir celdas

Para poder **expandir celdas** debemos trabajar con **grid-column-start y grid-column-end.** Estas propiedades permiten indicar dónde comienza y termina la celda.

Debemos pensarlo como una simple observación donde en un sistema de tres columnas, si queremos que nuestra celda ocupe todo el **espacio de la primer fila,** el valor será **1 en start y 4 en end.**

```
main { display: grid; grid-template-columns: auto auto auto ;}
div { border: 1px solid black; }

#expandir { grid-column-start: 1; grid-column-end: 4; }
```



Expandir celdas:filas

Para poder **expandir celdas** avanzando entre las filas, debemos trabajar con grid-row-start y grid-row-end. El concepto es exactamente el mismo.

En este caso **si queremos que nuestro elemento tome la segunda fila también lo haremos de la siguiente forma.**

Nótese que hemos hecho algunos cambios en las propiedades anteriores para lograr que esta nueva propiedad se efectivice.

```
main { display: grid; grid-template-columns: auto auto auto ;
       grid-template-rows: auto auto auto ;}
div { border: 1px solid black; }

#expandir { grid-column-start: 1; grid-column-end: 3;
            grid-row-start: 1; grid-row-end:3 ;  }
```

Propiedades Avanzadas

Existen propiedades muy interesantes como **grid-area** que nos permite nombrar áreas para construir nuestra grilla.

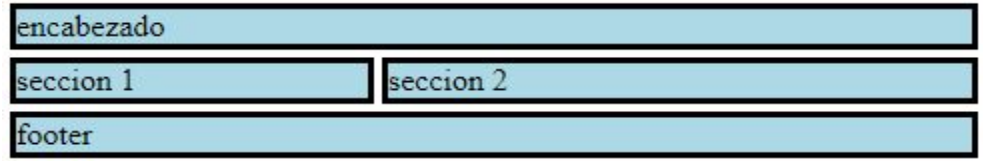
Esta forma de trabajar es sumamente interesante,en nuestro siguiente ejemplo hemos nombrado a múltiples áreas para construir una interfaz gráfica. Para lograr este proceso nuestros estilos se verán de la siguiente forma,

```
main { display: grid;
    grid-template-areas: 'encabezado encabezado encabezado' 'seccion1 seccion2 seccion2' 'footer footer footer';
    grid-gap: 4px;}
div { border: 3px solid black; background-color: lightblue; }

#encabezado { grid-area: encabezado;}
.seccion1{ grid-area:seccion1;}
.seccion2 { grid-area: seccion2;}
#footer { grid-area:footer;}
```

Propiedades Avanzadas

El estilo anterior luego se trabajará en nuestra estructura,

```
<main>
    <div id="encabezado">encabezado</div>
    <div class="seccion1">seccion 1</div>
    <div class="seccion2">seccion 2</div>
    <div id="footer">footer</div>

</main>
```



Revisión

- Repasar la propiedad **display con su valor grid.**
- Trabajar con todas las **opciones de grid.**
- Combinar las propiedades con **margin y padding**
- Complementar el trabajo en **nuestra interfaz con flex y float.**
- Experimentar con **grid-area** y sus opciones posibles.

¡Muchas gracias!

¡Sigamos trabajando!