educación IT
CODE YOUR FUTURE

# Bootcamp Java Developer

## Fase 2 - Java Web Developer

Módulo 14

educación IT
CODE YOUR FUTURE

# Uso básico y avanzado de Grid

# Grid

## Grillas en CSS

**`Grid`** es otro valor de **`display`** posible.

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

La intención de este apartado es comprender las potencialidades que nos brinda esta propiedad cuando es **utilizada de forma complementaria**, en nuestra interfaz, para lograr maquetar el diseño prototipado. **No es para nada un reemplazo de `flex`**.

Cada una de esas dos propiedades tiene su contexto. Parte de la experiencia que ganamos con desarrolladores o maquetadores es entender cuándo conviene más una forma u otra.

El hecho de conocer el funcionamiento de todas las formas de trabajo es suficiente para poder, con el tiempo, hacerlo correctamente.

## `display: grid`

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 comenzar a trabajar.

Veamos un ejemplo de utilización simple y los resultados que obtenemos.

En un **sistema de grillas** es fundamental tener presente que **tenemos filas y columnas**.

Para **determinar cuántas columnas** tendremos, trabajamos con **`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 en total para comprender el resultado final. En la estructura anterior, un formato de **4 elementos,** con los siguientes estilos como vemos en la imagen:

```
div { display: grid; }

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

**Resultado**

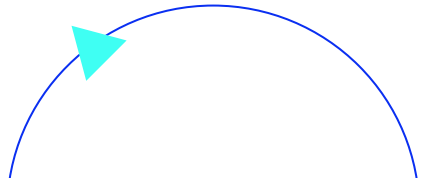
1 párrafo

2 párrafo

3 párrafo

4 párrafo

No es del todo prometedor y no implica demasiados cambios, veamos una imagen:



Sin embargo, si al código anterior le sumamos:

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

p { border: 1px solid ■black;}
```

**Resultado**



| | |
|---|---|
| 1 párrafo | 2 párrafo |
| 3 párrafo | 4 párrafo |

Dependiendo de cuántas medidas indiquemos será la cantidad de columnas que se trabajarán.

Para **alinear o distribuir los elementos en forma horizontal o vertical,** podemos trabajar con las mismas propiedades que usamos en el trabajo con **`display flex: justify-content`** o **`align-items`**:

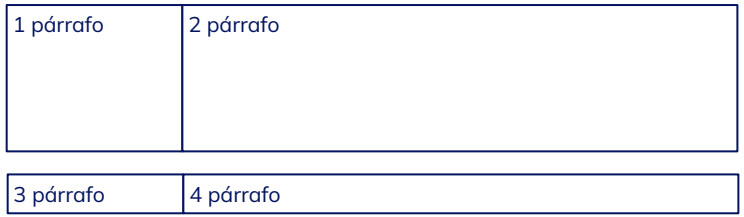
```
div { display: grid; width: 400px;
    grid-template-columns: 100px 300px;
    grid-template-rows:120px;
    gap: 15px 10px;
    justify-content: space-evenly;
     }

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

## Trabajar 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 que ya estén 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;}
```

**Resultado**



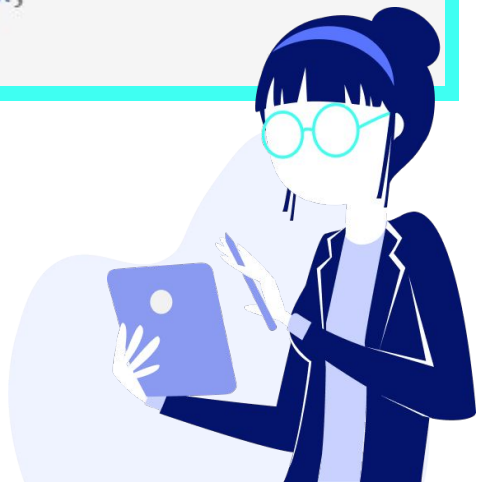
# Generar espacios en la grilla

## Column-gap y row-gap

Para generar espacios trabajamos con:

- **La propiedad `column-gap`:** representa el espacio entre las columnas.
- **La propiedad `row-gap`:** 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;
    }
```



## Grid-gap y gap

Como vimos, 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 sería:

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

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

Resultado

# Alinear el contenido

Para **alinear el contenido dentro de la grilla,** debemos trabajar con **`justify-content`**.

El valor **`start`** de esta propiedad permite orientar el contenido hacia el comienzo del contenedor:

```
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;}
```

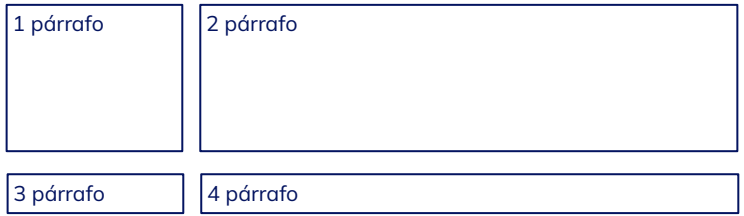
También, es posible alinear hacia el final, como podemos ver en la imagen, con el valor **`end`**:

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

Otra similitud con **`flex`**, es que también encontramos valores como **`space-around`** y **`space-between`**:

```
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; }
```

**Resultado**

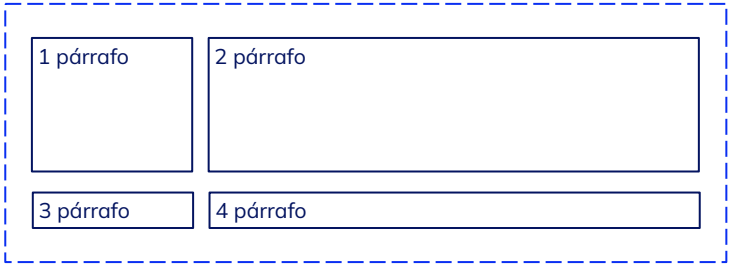


Para **alinear en forma vertical** se utiliza **`align-content`**.

Podemos centrar el contenido 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; }
```

**Resultado**



Como ocurre en la alineación horizontal, en la vertical contamos con palabras como **end, start, y center.**

```
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; }
```

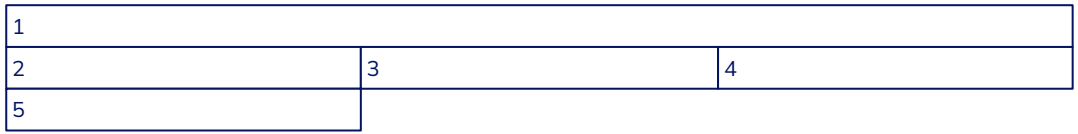
# Expandir celdas

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

En un sistema de tres columnas, si queremos que nuestra celda ocupe todo el espacio de la primera 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; }
```

**Resultado**



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

En este caso, si queremos que nuestro elemento tome la segunda fila también lo haremos como se muestra en el ejemplo debajo.

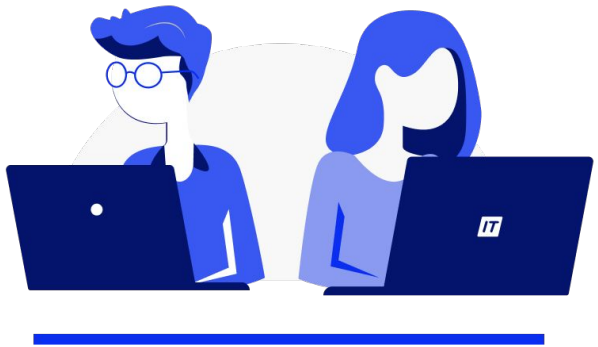
Nótese que hemos hecho algunos cambios en las propiedades anteriores para lograr que esta nueva propiedad se efectivice:

```
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 ;  }
```

# ¿Sabías qué...?

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

En nuestro siguiente ejemplo, nombramos múltiples áreas para construir una interfaz gráfica. Para lograr este proceso nuestros estilos se verán como se muestra en la diapositiva a continuación.

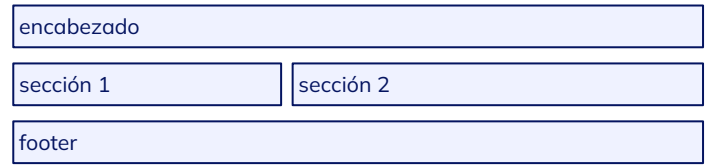


```
main { display: grid;
    grid-template-areas: 'encabezado encabezado encabezado' 'seccion1 seccion2 seccion2' 'footer footer footer';
    grid-gap: 4px;}
div { border: 3px solid black; background-color: lightblue; }

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

Con la siguiente estructura **generada en nuestro .html:**

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

</main>
```

**Resultado**



# Revisión

- Repasar los conceptos vistos de **`grid`**.
- Implementar estas nuevas propiedades.
- Ver todos los videos y materiales necesarios antes de continuar

# ¡Sigamos trabajando!