**CSS Grid**

Used for two dimensional layouts. Usamos distintas Rows y podemos especificar cuanto espacio ocupa cada elemento dentro de distintas rows.

**Intro**

Se necesita un **Parent container** (display: grid ) also some **Children Items.**

To the **Parent:**

Display: Grid

Grid-template-columns: 200px 300px

🡪 GTC: Size first - Size second.

* Grid is intelligent: If we add **10 Children**, but keep the layout with only **2 columns**, it will automatically créate **Rows** making the content fit for the values of each column.

🡪 IMPLICIT GRID

**Units – Auto**

We can use PX, REM, EM, %, Auto, FR

**Auto**: Responsive. The column is responsive, if we make the screen bigger, the column will be bigger. As well as smaller.

**Grid Rows**

**Grid-template-rows**. When we are using **Implicit Grid**, the row will become **Auto by default.** It till **fill** the selected **height** **value**.

**Fractions Units (FR)**

Se reparte la pantalla en fracciones y debemos elegir la cantidad

de espacio a utilizar.

Grid-Template-Columns:

1fr 🡪 100%

1fr 1fr 🡪 ½ First Col, ½ Second Col

2fr 1fr 🡪 2/3 First Col, 1/3 Second Col

Lo mismo se puede aplicar a Grid-Template-Rows

**Gap**

Column-Gap: 20px 🡪 Establecemos el espacio entre columnas

Row-Gap: 20px 🡪 Espacio entre Rows

Gap: 50px 🡪 Se aplica a ambas: Rows y Columns.

10px 0 🡪 Rows luego Columns

**Fractions Versus Percentage**

Lets have differents case scenarios:

**Template-Columns**: 200px 200px 200px 200px

**Gap:** 50px 100px

The total screen will be really big 800px + 300px gaps. We better be using Fractions.

**Template-Columns:** 50% 50%

**Gap:** 50px 100px

This will also be adding the Gap in between, making the screen bigger tan 100%.

**Template-Columns: 1fr 1fr**

**Gap: 50px 100px**

**This will be the best option. It wont be adding the gap space to the total! Being the screen size 100% including the gap.**

**Firefox developer Tools**

Visto que usamos Chrome no le dí mucha pelota a este tut. Se

trata de que cuando usamos Grid la dev tools de Firefox nos

muestra la grid que creamos. Nos da los números de cada

columna y fila que usaremos. (Grid Lines)

Will help us visualize the kind of grid we have.

**Grid-Lines**

Se refiere a las líneas entre el contenido de cada CELL, además de las Lines están los Tracks (el espacio ocupado por la Cell) y a su vez, si seleccionamos un par de Cells… Dicha selección se llamará Grid Area.

Quiero que mi Cell 1 ocupe dos espacios de columnas y dos de filas (2x2)

.cell-1 {

  grid-column-start: 1;

  grid-column-end: 3;

  grid-row-start: 1;

  grid-row-end: 3;

}

* Automáticamente “empujamos” a los ítems de al lado.
* Hay un Shortcut:

.cell-1 {

  grid-column: 1/3;

  grid-row: 1/3;}

Quiero que mi Cell 4 ocupe in Columns desde la 1 al final (sea cual sea la cantidad de columnas)

.cell-4 {

  grid-column: 1/-1;

}

* -1 Representa la Última Columna, sea cual sea su número (como index in Python)

**Naming Grid-Lines**

Les podemos dar nosotros mismos un nombre para manejarnos mas fácilmente.

**G-T-C**: **[start]** 100px 100px **[Line-3]** 100px **[end]**

**G-T-R:** **[start]** 100px 100px 100px **[end]**

**.cell-1** {grid-column: start/line-3; grid-row: start/end}

🡪 Ocupará 2/3 de las columnas (1/3)

🡪 Ocupará toda el área de las filas (1/4)

* Podemos elegir usar números o darles nombre y llamarlas por su nombre**. CUIDADO! Se da nombre a las LINEAS y NO a los TRACKS.**

**Grid Template Areas**

Podemos establecer distintas áreas dentro de la grid y asignar cada elemento al área deseada.

  grid-template-areas:

    "a a b"

    "a a b"

    "c c b"

    "d d d "

    "e f f";

* Es como dar una imagen de la grid y nombrar las áreas a ocupar. **A** forma un cuadrado, **B** un rectángulo vertical, **C** rectángulo pequeño, **D** rectángulo horizontal largo, **E** un pequeño cuadrado.
* Deben coincidir:
  + Cantidad de Rows con las G-T-R.
  + Cantidad de Columns con las Fractions en G-T-C.
* Luego se declarará las Cells a ocupar dichas áreas:

Cell-1 {grid-area: a}

Cell-2 {grid-area: b}

Cell-3 {grid-area: c}

Cell-4 {grid-area: d}

Cell-5 {grid-area: e}

**Order Property**

We can change order property with:

Order: 1 🡪 Change numbers (could be negatives too)

Order: -1

**Repeat Function**

All the time we wanted to repeat value we had to manually write

it out. We have the Repeat Function.

Supongamos que queremos 5 columnas de 100px

Grid-Template-Columns: **repeat** (4, 100px)

* Unicamente sirve para el mismo valor! No 100 y 200px.

**Justify-Content**

There are some Properties that we can “borrow” from Flexbox. Alinea los elementos **Horizontalmente**

Flex-Start 🡪 Default

Flex-End 🡪 Todo toca el borde derecho.

Center 🡪 Todo se centra horizontalmente

Space-Between 🡪 Espacio en medio, columnas tocan bordes

Space-Evenly 🡪 Espacio igual entre bordes y columnas

Space-Around 🡪 Harbá márgenes para cada columna.

Conviene usar PX o medidas exactas, ya que las Fracciones Llenan todo el espacio.

**Align-Content**

Se encarga de alinear elementos **Verticalmente.** Podemos alinear sin especificar la cantidad de columnas. Necesitamos establecer una **Height.**

Stretch 🡪 Default. Se estiran desde Top a Bot, SI hay más Rows… Se dividirán el espacio.

Flex-Start 🡪 Todo comienza del Top

Flex-End 🡪 Comienzan desde Bot

Space-Around 🡪 Espacio entre ítems y bordes

Space-Between 🡪 Espacio enter ítems, filas tocan bordes Top y Bot

Space-Evenly 🡪 Espacios iguales entre Rows y bordes.

**Align-Items, Justify-Items, Align-Self, Justify-Self, Self.**

Align-Items : Center 🡪 Mete el **contenido** de una Cell en el medio **Verticalmente.** Dejando espacios arriba y abajo por igual

Align-Items: Flex-End 🡪 **Contenido** toca el **Bot** de la Cell.

Justify-Items: Center 🡪 Centra el **contenido Horizontalmente** dentro de la Cell.

Cell-1 {Justify-self: end} 🡪 Hace que el **Contenido toque** el borde **Derecho** de la Cell seleccionada.

Cell-1 {Align-Self: flex-end} 🡪 El **Contenido** **toca el Borde Bot.**

**Min-Max ()**

Supongamos que a medida que la screen is smaller, I wouldn’t want all the cells to become smaller. Maybe i want to declare a Min-Width for the second column, giving it a cap.

grid-template-columns: 1fr minmax(300px, 1fr) 1fr

* Con esto aclaramos que la Segunda Col tenga como: Minimo 300px, Maximo 1fr.
* Es muy útil para cuando querramos poner unas Column Layout sin usar media query.

**Auto-Fit and Auti-Fill**

Valores que me ayudarán para set-up a CSS Grid without using Query.

Imaginemos que queremos 3 Column Layout. But imagine we have even more ítems or the Database is giving us 1 thousand ítems.

There must be a way to set up the columns as many as i need without knowing.

grid-template-columns: repeat(auto-fill, 100px)

* **Creará columnas** as it is **needed.** Si la screen is getting smaller it will créate Rows. As the screen gets Bigger it will fit them un 1 Row only by **creating lots of Columnsm even more as needed to be filled.**

grid-template-columns: repeat(auto-fit, 100px)

* Only Set up as many columns as needed. **NO MORE.**
* We can use the **Minmax(auto-fit,** 1fr**)**. A medida que agrandemos la pantalla (al tener MAX 1fr) **las columnas ocuparan todo el espacio de la pantalla** (sean la cantidad de cols que sean)