

# CSS GRID LAYOUT



**¡HOLA!**

**Soy Diana Aceves**

Frontend Developer

Twitter: **@diana\_aceves\_**

## EJEMPLOS DEL CURSO

Colección Codepen:

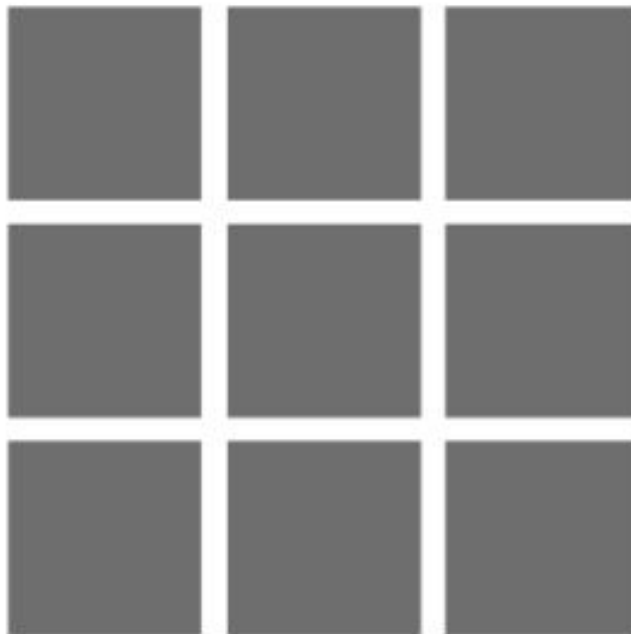
“Escuela IT 2017:

**Curso CSS GRID LAYOUT”**

<https://codepen.io/collection/712dfdc773b2532beaff9157a2ffe194/>

# ➔ ¿QUÉ ES **GRID LAYOUT** ?

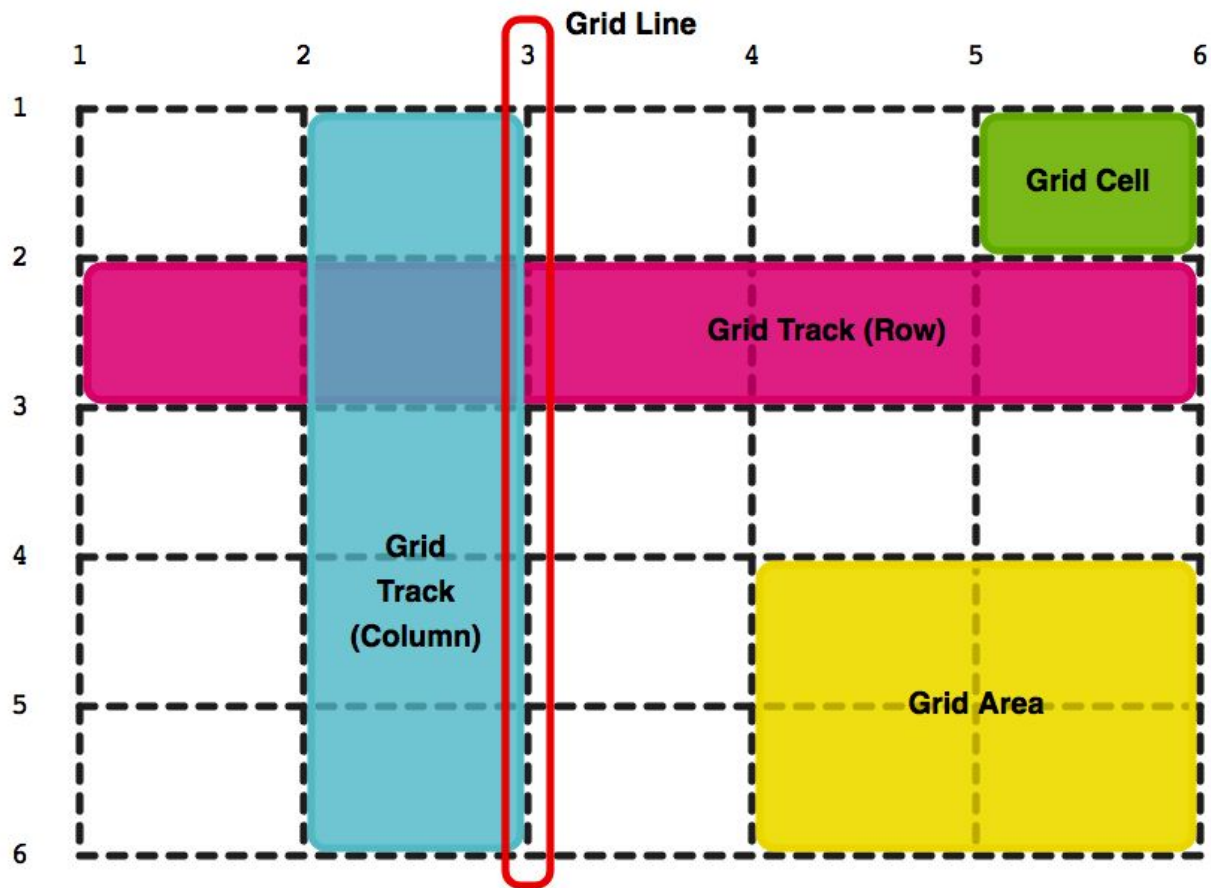
- ▶ Sistema de rejilla en **2 DIMENSIONES**



## ➡ ¿QUÉ TIENE **DIFERENTE** ?

- ▶ Voy a poder colocar los items **DONDE QUIERA...**
- ▶ ...**PERO** habrá items que **SE COLOQUEN SOLOS** (AUTO-PLACEMENT)
- ▶ Puedo **HACER LO MISMO DE MUCHAS FORMAS...**
- ▶ ...**PERO NO TODAS HACEN LO MISMO.**
- ▶ Controlo las 2 dimensiones, **NO ES FLEXBOX.**
- ▶ La colocación de los items es muy libre, **NO ES UNA TABLA.**
- ▶ Tiene una **EXTENSA SINTAXIS.**
- ▶ Nos va a volver un poco locos, pero **VA A CAMBIAR EL CSS PARA SIEMPRE.**

# ➡ CONCEPTOS BÁSICOS





## EMPECEMOS: **DISPLAY GRID**

En cuanto declaro

**DISPLAY: GRID** o

**DISPLAY: INLINE-GRID**

los hijos directos de ese elemento pasan a ser **GRID ITEMS**.

Vamos a verlo:

**[GRID LAYOUT - EJ.1](#)**

## **DEFINIENDO TRACKS EN EL GRID**

- ▶ Creamos el grid **EXPLÍCITAMENTE**.
- ▶ Es decir, al contenedor GRID, le indico cómo deben ser las filas y las columnas:
- ▶ **EXISTEN MÚLTIPLES SINTAXIS Y FORMAS DE HACERLO.**

**GRID LAYOUT - EJ.2**



## ➔ Funciones y keywords **MUY ÚTILES**

- ▶ Tengo que saber cada track? Cuánto mide?  
Cuántos hay? Escribirlos todos? **MUERO!!!**

**NO**

- ▶ `repeat()`
- ▶ `minmax()`
- ▶ `auto-fill / auto-fit`

**GRID LAYOUT - EJ.3.1 - REPEAT / MINMAX**

**GRID LAYOUT - EJ.3.2 - AUTO-FILL / AUTO-FIT**

## ➔ Visto hasta **AHORA**

- ▶ **DISPLAY: GRID** genera una serie de tracks donde, si no indico más, **los items se colocan automáticamente** mediante el algoritmo de **AUTO-PLACEMENT**.
- ▶ Puedo concretar más la rejilla generada con:
  - **grid-template-columns/rows**
  - **grid-auto-columns/rows**
  - Y todas las **funciones/propiedades/keywords** que hemos visto.

PERO AÚN NO HEMOS TOCADO LOS **GRID-ITEMS**



## Posicionando **GRID-ITEMS** : **NÚMEROS DE LÍNEA**

La enorme potencia de GRID LAYOUT en parte viene dada porque una vez generada la rejilla, **puedo COLOCAR LOS ITEMS DONDE QUIERA** dentro de esta.

DEFINO UN GRID DONDE DESPUÉS PUEDO POSICIONAR LOS ITEMS.

**EJ.4 - COLOCACIÓN + N° DE LÍNEA**

**EJ.APLICACIÓN.1 - GRID 12 COLUMNAS**



The diagram illustrates the merging process of 28 small blocks into a single large block. The blocks are numbered 1 through 28. The merging steps are as follows:

- Step 1: 28 small blocks (1-28) are arranged in a 4x7 grid.
- Step 2: Blocks 1-2 merge to form block 13.
- Step 3: Blocks 3-4 merge to form block 14.
- Step 4: Blocks 5-6 merge to form block 15.
- Step 5: Blocks 7-8 merge to form block 16.
- Step 6: Blocks 9-10 merge to form block 17.
- Step 7: Blocks 11-12 merge to form block 18.
- Step 8: Blocks 13-14 merge to form block 19.
- Step 9: Blocks 15-16 merge to form block 20.
- Step 10: Blocks 17-18 merge to form block 21.
- Step 11: Blocks 19-20 merge to form block 22.
- Step 12: Blocks 21-22 merge to form block 23.
- Step 13: Blocks 23-24 merge to form block 24.
- Step 14: Blocks 25-26 merge to form block 25.
- Step 15: Blocks 27-28 merge to form block 26.
- Step 16: Blocks 24-25 merge to form block 27.
- Step 17: Blocks 26-27 merge to form block 28.