CSS GRID LAYOUT

¡HOLA!

Soy **Diana Aceves**

Frontend Developer

Twitter: **@diana_aceves_**

EJEMPLOS DEL CURSO

Colección Codepen:

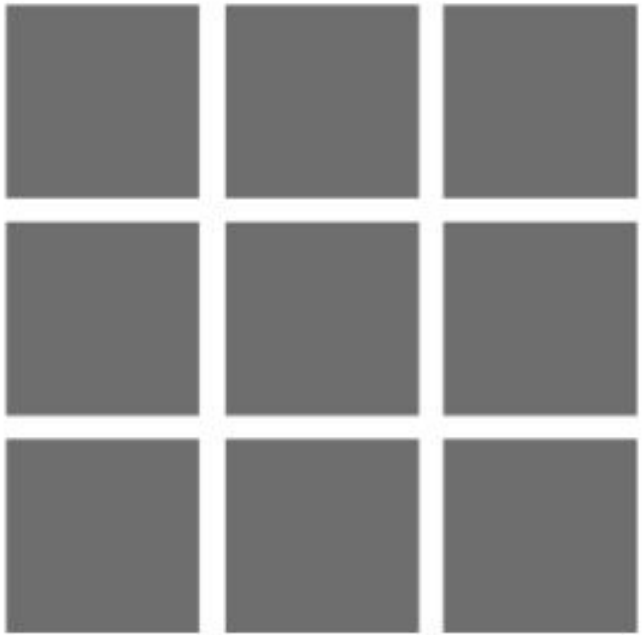
“Escuela IT 2017:

Taller CSS GRID LAYOUT”

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

¿QUÉ ES GRID LAYOUT ?

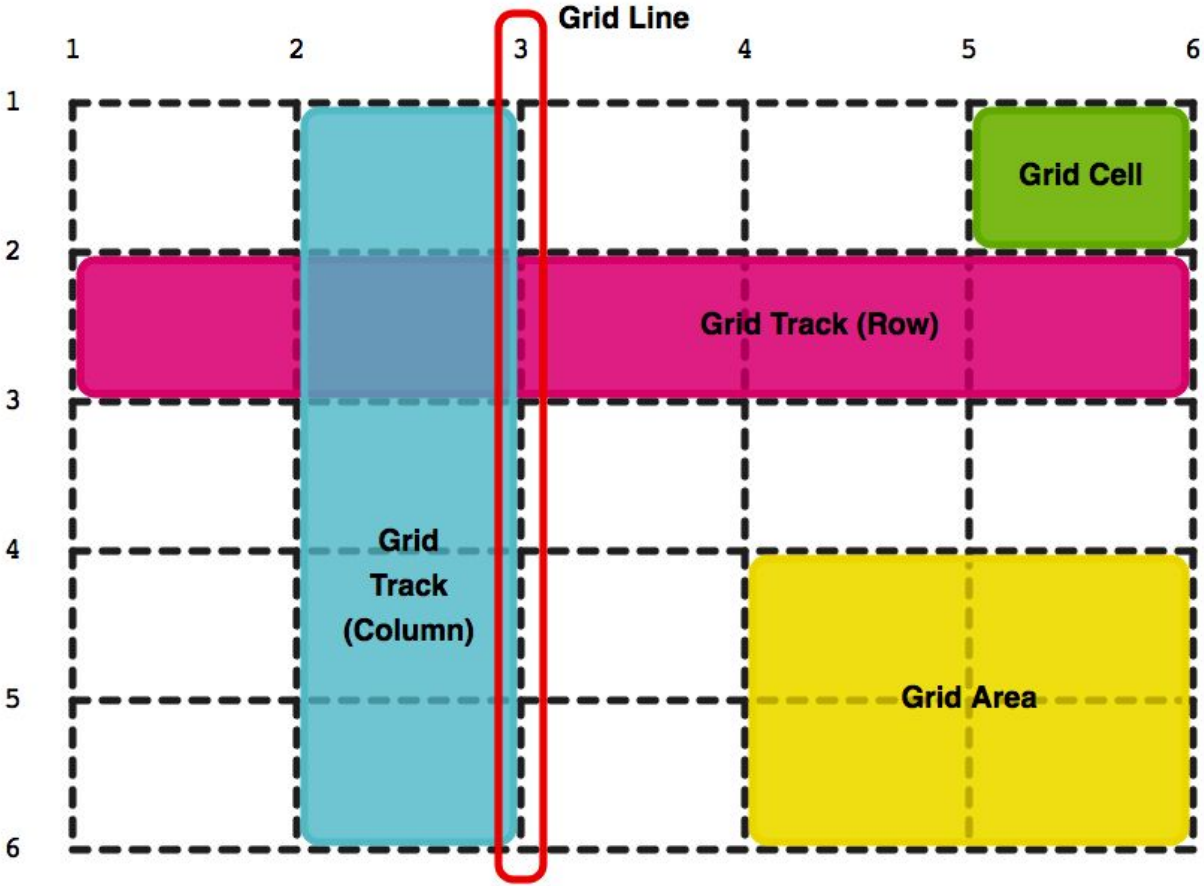
- Sistema de rejilla en
  2 DIMENSIONES



¿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:
EJ.1 - DISPLAY:GRID

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.**

EJ.2 - GRID-TEMPLATE-COLUMNS/ROWS

Funciones y keywords MUY ÚTILES

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

NO

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

EJ.3.1 - REPEAT / MINMAX
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.

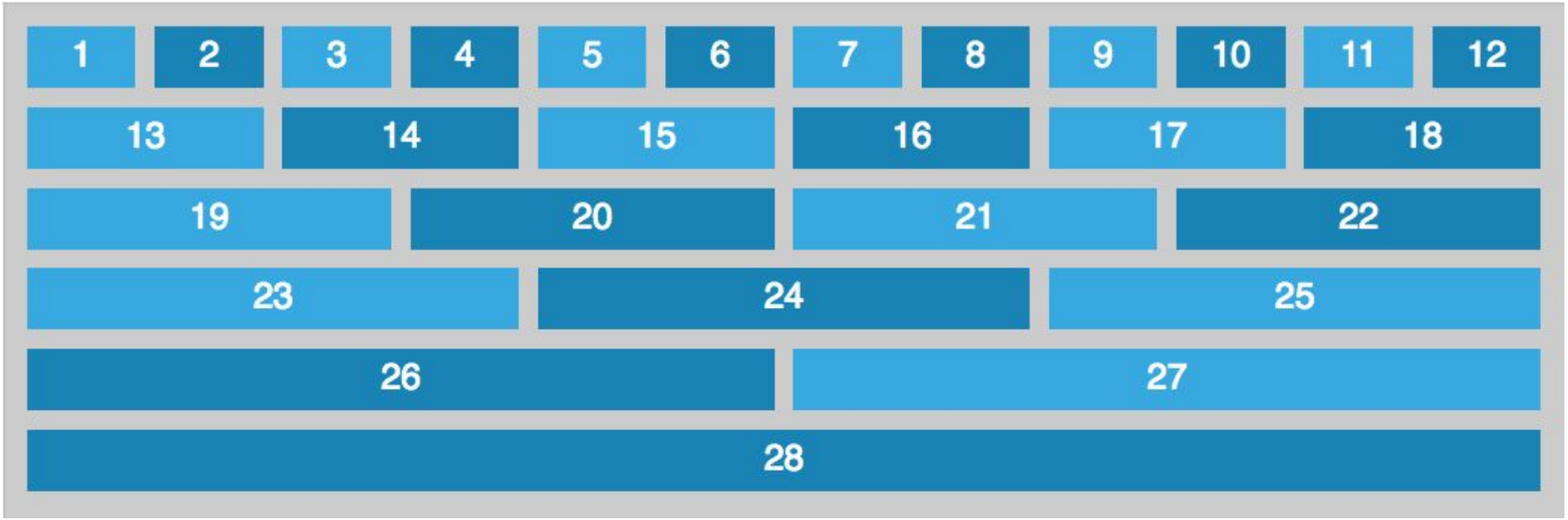
DEFINO UN GRID DONDE DESPUÉS PUEDO POSICIONAR LOS ITEMS.

EJ.4 - COLOCACIÓN + Nº DE LÍNEA

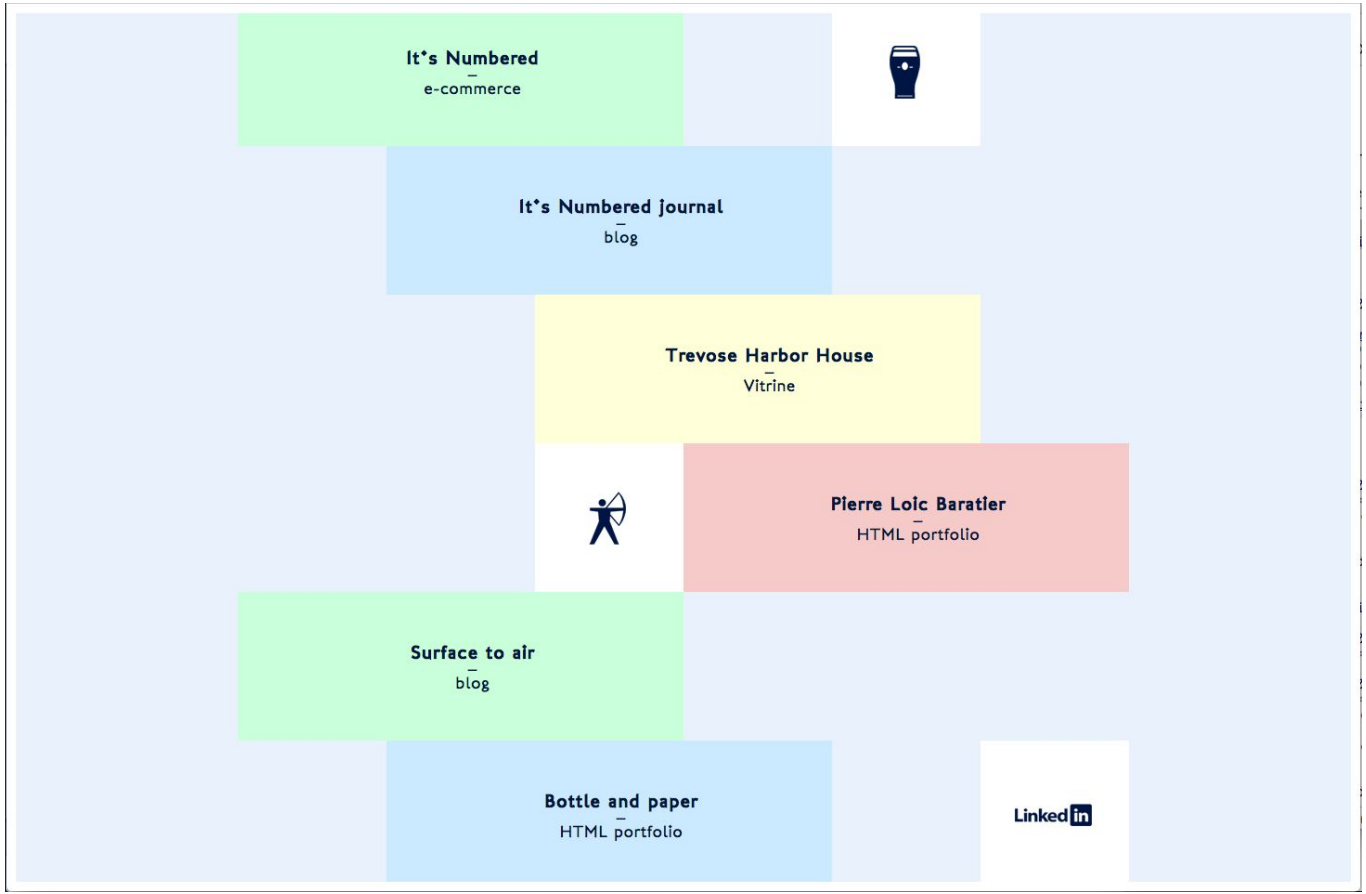
EJ.APLICACIÓN. 1 - GRID 12 COLUMNAS

EJ.APLICACIÓN. 2 - LAYOUT REAL thomasrobin

EJ.APLICACIÓN. 1 - GRID 12 COLUMNAS



EJ.APLICACIÓN. 2 - LAYOUT REAL thomasrobin



Posicionando GRID-ITEMS : LÍNEAS NOMBRADAS

- Me ayudan a recordar cómo van los tracks en layouts complejos.
- En **RESPONSIVE** me evitan sobreescribir la colocación de algunos items.
- Si cambia el número de tracks **NO TENGO QUE REPOSICIONAR TODOS LOS ELEMENTOS.**

EJ.5.1 - LÍNEAS NOMBRADAS

EJ.5.2 - LINEAS NOMBRADAS (nombres múltiples)

EJ.5.3 - LÍNEAS NOMBRADAS (variación número de tracks)

EJ.5.4 - LÍNEAS NOMBRADAS (nombres repetidos)

Posicionando GRID-ITEMS : ÁREAS NOMBRADAS

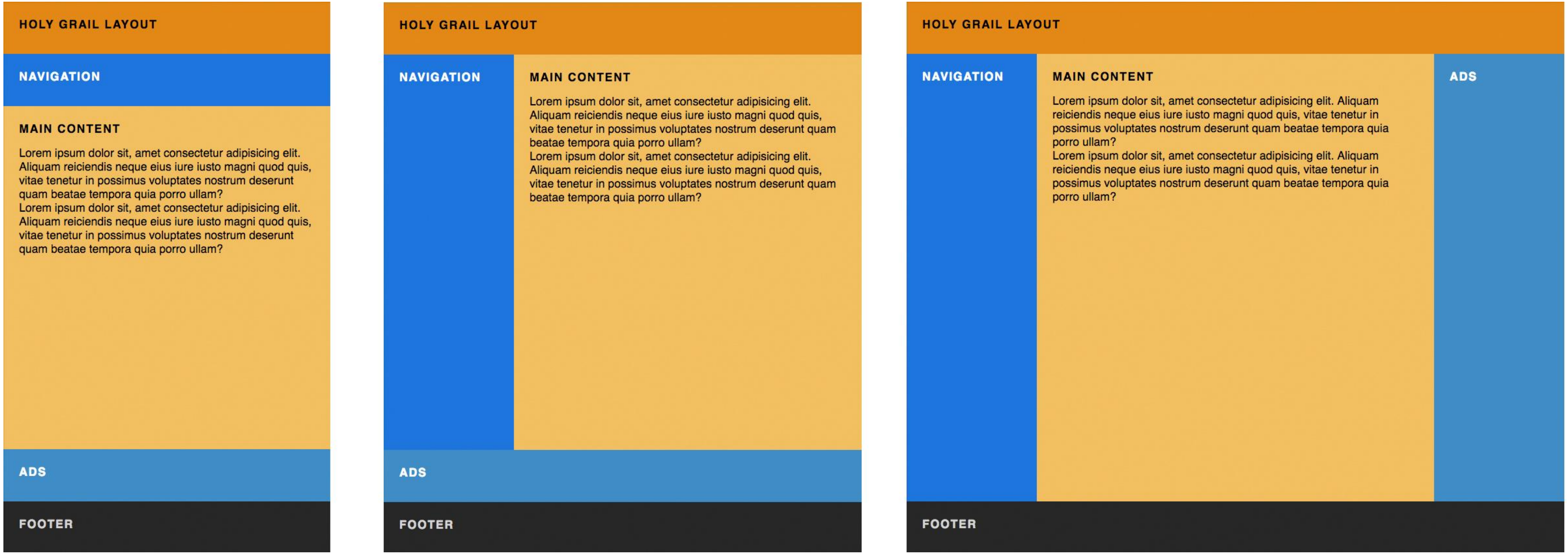
Dos nuevas propiedades :

- CONTENEDOR -> **GRID-TEMPLATE-AREAS**
- ITEMS -> **GRID-AREA**

EJ.6 - ÁREAS NOMBRADAS

EJ.APLICACIÓN. 3 - HOLY GRAIL LAYOUT

EJ.APLICACIÓN. 3 - HOLY GRAIL LAYOUT



MAGIC LINES

- **GRID AREAS** crean **LINE NAMES IMPLÍCITOS**

- **LINE NAMES** con una determinada sintaxis crean **AREA NAMES IMPLÍCITOS**

EJ.7.1 - MAGIC LINES: ÁREAS -> LÍNEAS NOMBRADAS

EJ.7.2 - MAGIC LINES: LÍNEAS NOMBRADAS -> ÁREAS

BOX ALIGNMENT
ALINEACIÓN DE LOS ÍTEMS

ALINEACIÓN EN **COLUMN/BLOCK AXIS**:

- **align-self**
- **align-items**

ALINEACIÓN EN **ROW/INLINE AXIS**:

- **justify-self**
- **justify-items**

EJ.8 - ALINEACIÓN ÍTEMS

BOX ALIGNMENT
ALINEACIÓN DE LOS TRACKS

ALINEACIÓN EN **COLUMN/BLOCK AXIS**:

- **align-content**

ALINEACIÓN EN **ROW/INLINE AXIS**:

- **justify-content**

EJ.9 - ALINEACIÓN TRACKS

ALGORITMO AUTO-PLACEMENT

- Reglas que controlan **dónde se colocan los items** cuando no establecemos su posición o por colocar otro, pierde su sitio natural.
- Aunque no les dé información de dónde colocarse, ellos buscarán sitio en celdas libres o crearán los tracks necesarios para hacerlo.
- Lo primero que determina cómo funciona AUTO-PLACEMENT es el valor de la propiedad:

  grid-auto-flow

PROPIEDAD GRID-AUTO-FLOW

Grid-auto-flow determina **2 ASPECTOS**:

- **Dirección**: Define la dirección en la que el grid va a crecer si necesita colocar items por auto-placement.
    - **ROW** (default)
    - **COLUMN**
- **Modo**: Determina si el algoritmo intentará rellenar huecos vacíos para colocar items por auto-placement.
    - **SPARSE** (default)
    - **DENSE**

EJEMPLOS: AUTO-PLACEMENT

EJ.10.1 - GRID-AUTO-FLOW: COLUMN

EJ.10.2 - GRID-AUTO-FLOW: DENSE

EJ.10.3 - ADIVINANDO QUÉ PASA VOL.2

Creemos que lo hemos entendido...

INTRODUCCIÓN AUTO-PLACEMENT + GRID-ROW/COLUMN

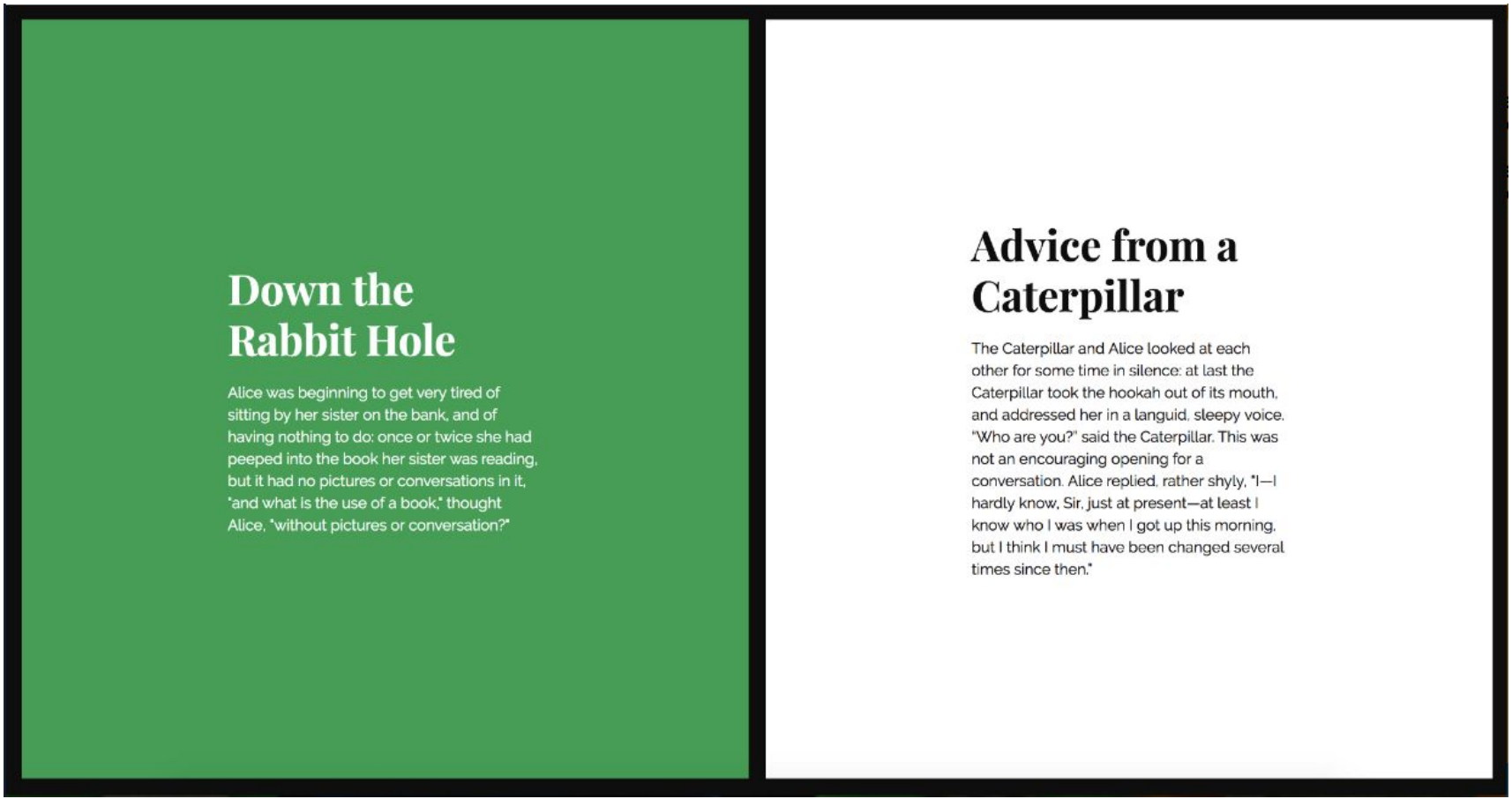
AUTO-PLACEMENT + GRID-ROW/COLUMN

Para colocar los items, el algoritmo también tiene en cuenta qué nivel de posicionamiento tienen:

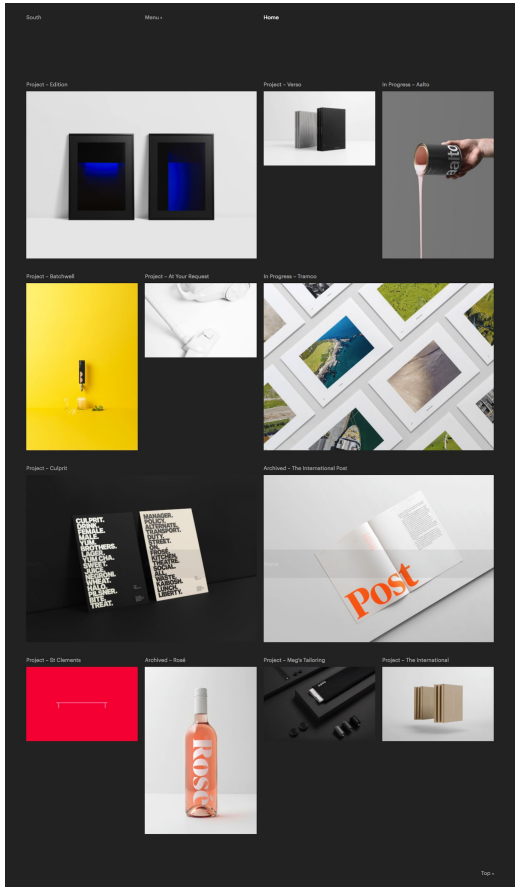
1. Items con posición **definida.**
2. Items con posición en el eje ppal **DISTINTA DE AUTO.**
3. Resto de items**.**

EJ.11 AUTO-PLACEMENT + GRID-ROW/COLUMN

EJ.APLICACIÓN. 4 - TWO FULL COLUMNS



EJ.APLICACIÓN. 5 - studiosouth.co.nz



CSS GRID LAYOUT + ANIMACIONES

En teoría se pueden animar:

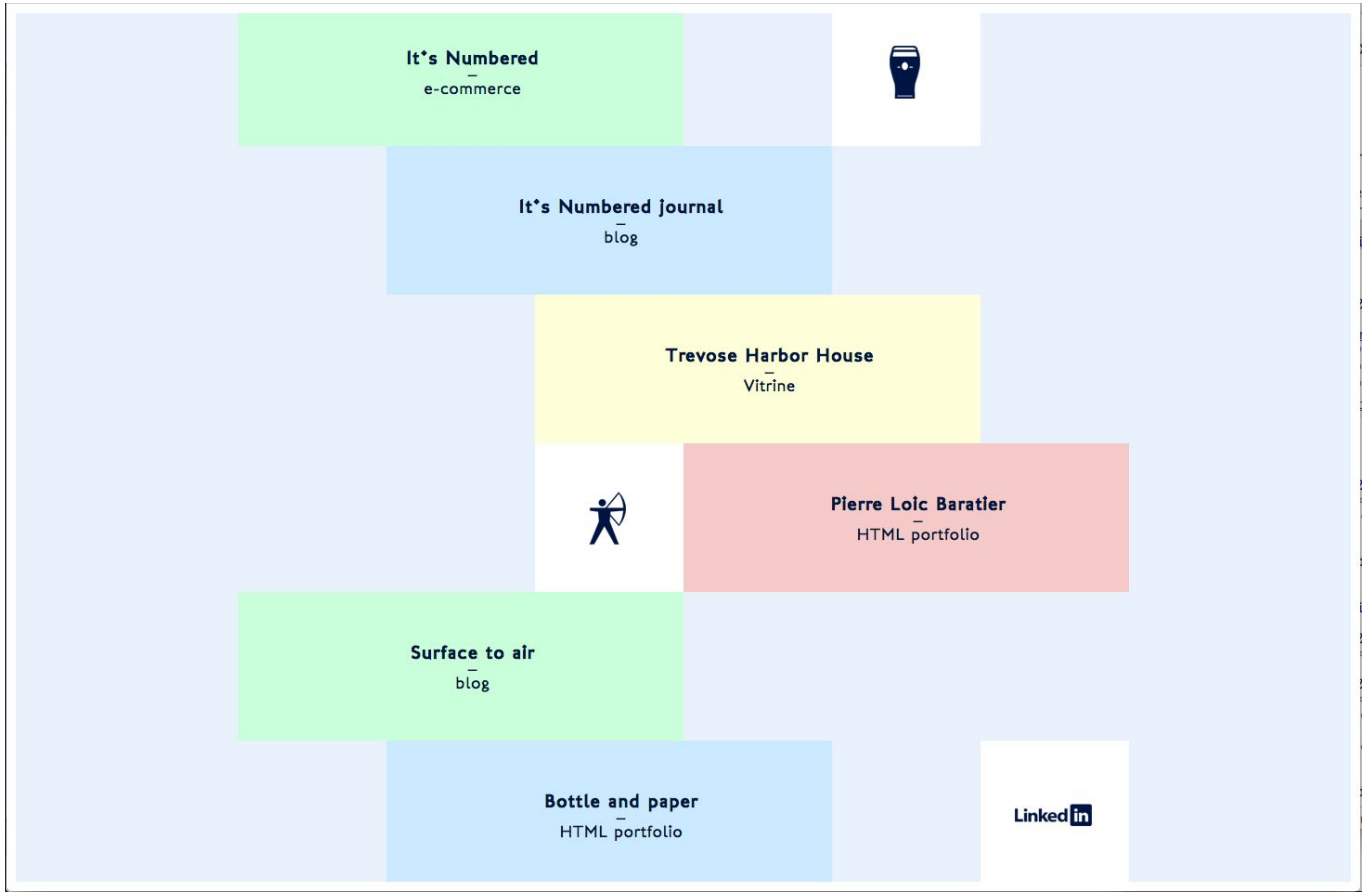
- **grid-template-columns/rows**: solo si son una lista de valores de longitud, porcentajes o calc, y solo cambian los VALORES. Es decir, no cambia el número de tracks p.e.
- **grid-column/row-gap, grid-gap**

CSS GRID LAYOUT + ANIMACIONES

En realidad (07/2017):

Browser	grid-gap, grid-row-gap, grid-column-gap	grid-template-columns	grid-template-rows
Firefox 55, Firefox 53 Mobile	✅	❌	❌
Safari 10.1	❌	❌	❌
Chrome 61	❌	❌	❌
Chrome for Android 58, Opera Mini 25	❌	❌	❌

EJ.APLICACIÓN. 6 - ANIMANDO thomasrobin



RECURSOS Y REFERENCIAS

- Especificación W3C CSS Grid Layout
- CSS Grid Layout - Mozilla Developer Network
- Grid by example - Rachel Andrew
- A complete guide to Grid - CSS Tricks
- Comprendiendo auto-placement - EnvatoTuts
- Post de Manuel Rego (@regocas) para entender AUTO-PLACEMENT
- Things I've learned about CSS Grid Layout - CSS Tricks
- Interesting facts about CSS Grid Layout - CSS Tricks
- Grid fallbacks and overrides - Rachel Andrew
- Grid Bugs - Rachel Andrew
- Usando Grid Layout con @supports - (@jorgeATGU) Jorge Aznar

GRACIAS!

Twitter: **@diana_aceves_**