CSS Grid Layout specification

La Norma de Diseño de Cuadrícula que próximamente estará disponible en los navegadores web nos permite diseñar una serie de elementos por medio del Diseño de Hojas en Cascada (**Cascade Style Sheets** o mejor conocido por su acrónimo de tres letras **CSS**) con una rapidez asombrosa y pocas líneas de código. Por cierto, la imagen de introducción a este artículo fue creada por medio de código, no utilizamos el ratón ni una sola vez.

Introducción.

En esta entrada describiremos muy brevemente lo que son **HTML** y **CSS** para así poder probar en avance -aún no es está implementado en los navegadores web- las nuevas Normas de Diseños de Cuadrícula. Vamos pues, en esta novedosa área del diseño de páginas web.

HTML.

El **Hyper Text Markup Language** o simplemente **HTML** es un lenguaje de marcado, tal como lo explicamos anteriormente por este vuestro humilde portal web. Una muy buena definición, en castellano, de dicho lenguaje de marcado la podeis leer acá en este enlace. Específicamente vamos por la recomendación 5 que se presentó en diciembre de 2012 y que progresivamente se ha ido incorporando a los modernos navegadores web, tanto de código fuente abierto como privativo. *Pero es el software libre quien lleva la delantera en esto* y es por ello que los programadores de Mozilla Firefox están siempre tratando de alcanzar a las recomendaciones (normas, de facto) planteadas: una labor de hormiga, un paso a la vez.

El dibujito de advertencia que os hemos puesto es, entonces, para advertiros que en vuestro navegador actual ***NO FUNCIONARÁN PARA NADA LOS EJEMPLOS CON QUE TRABAJAREMOS HOY*** *(a la fecha de hoy 14 marzo 2017 ya fue actualizado Mozilla Firefox)**.*** Tal vez ustedes pensarán que es una total y absoluta pérdida de tiempo si dichas normas cambian o, peor aún, nunca se llega a implementar en navegador web alguno (o son retiradas en su uso), pero oigamos a Eduardo Galeano: "*para eso son las utopías, para mantenernos en movimiento".* Más adelante os explicaremos como instalar la sempiterna versión Beta de Mozilla Firefox: **Nightly.**

CSS.

Cuando presentamos el tema lo dijimos: vamos a ir muy rápido en los conceptos básicos para poder presentar la nueva norma de diseño de cuadrícula. Si **HTML** es un lenguaje de marcado basado en etiquetas podemos decir entnonces que viene a ser como el chasís y motor de nuestro automóvil y he aquí que el **CSS** viene a ser la carrocería y pintura del mismo. Como ya podéis sospechar, tanto en páginas web y como en automóviles, los humanos estamos es pendientes de las formas y colores *la presentación de los elementos, y esto ya cae dentro del ámbito del gusto de cada quien.*

> Con HTML podremos crear nuestra propia página web.
>
> CSS es un lenguaje que describe el estilo de un documento HTML.
>
> CSS describe cómo los elementos HTML deberían ser mostrados.

Es por ello que siempre hemos considerado que **CSS** es y será la norma que nos ahorrará gran trabajo pues nos centraremos en imaginar y escribir las etiquetas **HTML** mínimas necesarias para nuestros programas [entradas, botones, listas, etcétera enlazadas o no con alguna(s) base de datos] y con **CSS** y **JavaScript** (eso sí, acompañado de base de datos) podremos ofrecer a nuestros usuarios unas plantillas básicas de presentación que ellos y ellas podrán escoger e incluso personalizar (por ello el uso de base de datos para luego poder recuperar esas preferencias). Pero no no extendamos más en este punto y sigamos adelante con nuestro objetivo principal, ***acompañadnos, por favor***.

Prerrequisitos.

Firefox Mozilla Nightly.

Se tiene estimado que el 7 de marzo de 2017 salga a la luz pública la versión 52 de Mozilla Firefox y por consiguiente el apoyo a al "**CSS Grid Layout**". Mientras tanto debemos utilizar la versión de avanzada, que siempre está y estará en versión BETA (una versión para un público reducido y previo a su lanzamiento) la cual es llamada **Nightly.** Acá os advertimos de nuevo: en este navegador no den nada por sentado, habrán cosas que se mejorarán, otras que se corregirán y algunas que no irán a la versión definitiva siguiente. **De más está deciros que cualquier error que veaís lo podréis colaborar en su corrección, COLABORAD, pero no salgáis a criticar por las redes sociales porque no es una versión definitiva.** *Además, si todos nos diéramos a la tarea de criticar por ganar fama o popularidad* ***¿Quién demonios se va a dedicar a programar? ¡Alguien tiene que trabajar, colaborad, por favor!***

Instalando Nightly en Ubuntu.

Para instalarlo debemos tener permiso de administrador o usuario **root**, abrimos una ventana terminal e introducimos los siguientes comandos:

```
  sudo  add-apt-repository  ppa:ubuntu-mozilla-daily/ppa      sudo  apt-get update   sudo apt-get install firefox-trunk
```

https://twitter.com/ks7000/status/838454136966361088

Una vez descargados aproximadamente 46 megabytes de datos comprimidos, se procederá a su instalación y ocupará 156 megabytes en disco duro. Una observación importante que acota al final de la instalación es que debemos cerrar todas las ventanas que tengamos abiertas con Mozilla Firefox *o de lo contrario tenderemos problemas, eso es un indicativo de que las librerías entre Mozilla Firefox 51 (a la fecha) y Nightly se comparten.*

https://twitter.com/ks7000/status/838457258069659648

Instalando Nightly en Debian.

Aún usamos Debian 7, en honor a la verdad nos hemos quedado rezagados en eso porque siempre nos decantamos más por Ubuntu. La mala noticia es que no hallamos, de buenas a primera, un repositorio para esa versión, sin embargo observamos que para la versión 9 abundan repositorios, intentad fortuna con el siguiente a ver si podéis vosotros solitos.

CSS Grid Layout specification.

¿Qué es la Norma de Diseño de Cuadrícula CSS?

El Diseño de Cuadrícula nos permite separar apropiadamente el orden de los elementos desde la fuente contra su presentación visual. Como diseñador esto significa que eres libre de cambiar la ubicación de los elementos de la página en función del dispositivo que sea presentado *sin necesidad de comprometer la sensible estructura de un documento HTML y siempre con un diseño* ***responsivo.***

Para el Real Diccionario de la Lengua la palabra **responsivo** significa "perteneciente o relativo a la respuesta" pero acá en programación de ordenadores lo enfocamos en que **la respuesta viene representada por un tamaño de pantalla.** Esto es así debido al la proliferación de dispositivos móviles (entiéndase, principalmente, teléfonos celulares o móviles) que tienen diversos tamaños

de pantalla, *pero no os llaméis a engaño, ya estos aparatitos están, de hecho, sustityendo a los ordenadores personales.* Si queréis ver una prueba de lo que hablamos, y si utilizáis Firefox, probad y haced click en el menú desplegable "Herramientas" -> "Desarrollador web" -> "Modo de diseño adaptable" (o por medio del teclado pulsad de manera simultánea CONTROL+INVERSO+M).

Presentado lo anterior esperamos os habréis dado cuenta del futuro que os presentamos en este blog, ¡pinta bien y promete!

Terminología de cuadrícula.

Rápidamente os presentamos los conceptos que debéis conocer y manejar bien antes de comenzar a editar código alguno, debemos saber exactamente los nombres de los elementos.

Líneas de cuadrícula.

Son las líneas que "maquillan" a la cuadrícula, pueden ser horizontales o verticales. Podremos referirnos a ellas con un número o nombre, la mejor analogía serían las líneas de una hoja de cálculo, elemento abstracto que es conocido pro la mayoría de los usuarios de ordenadores. Acá una figurita, por si las dudas.

[SVG: Lineas de rejilla]

Pistas de cuadrícula.

Una pista de cuadrícula es toda el área entre dos líneas de cuadrícula. Dibujo de nuevo:

[SVG: Pista de cuadrícula]

Celda de cuadrícula.

Una celda de caudrícula es la menor área posible a representar y se define como toda el área delimitada entre dos líneas verticales y dos líneas horizontales.

[SVG: Celda de cuadrícula]

Área de cuadrícula.

Un área de cuadrícula también delimitada por dos líneas horizontales y dos líneas verticales **pero contiene dos o más celdas de cuadrícula.**

[SVG: Área de caudrícula]

Ahora que tenemos nuestras definiciones básicas podremos comenzar nuestro tutorial en sí.

Archivos base.

Bien podemos escribir un solo documento **HTML** que incluya **CSS** o bien podemos escribirlos en archivos aparte, escoged lo que os más os guste para estos fines didácticos. *En todo caso presentamos el código básico para cada uno de ellos.*

Un solo documento HTML.

A

B

C

D

E

F

Con un archivo CSS externo.

Para colocar nuestro código en un archivo externo tomaremos el código anterior ***pero en vez de colocar*** simplemente colocaremos una sola línea con el siguiente código:

Insertando estilo en cada línea.

Si la opción anterior -un archivo CSS externo- es la mejor opción ya que por medio del lenguaje **PHP** podemos generar un "**link rel"** apuntando a diferentes archivos .ccs para cada usuario, la

tercera opción nos parece la menos adecuada pues se debe generar en cada línea de código **HTML** las instrucciones **CSS**. Para ello haríamos algo como esto:

```
# Encabezado 1 azul con margen izquierdo a 30 píxeles
```

Lo único bueno es que esta opción es la que veremos siempre, si la aplicamos, es decir: a medida que el navegador va aplicando los estilos el que prevalece es el último especificado, en este caso la línea en sí en código **HTML**. Por supuesto, si aplicamos código **PHP** perfectamente podemos programar para que, por ejemplo, cada vez que coloque un

inserte un estilo para cada usuario *pero li impractico de esto es que para cada línea repetiremos instrucciones, aumentando la cantidad de bytes a transferir: algo ineficiente.* Si se hace esta declaración de alguna de las dos primera maneras solo tenemos que establecer *una sola vez para todo el documento,* a menos que en algún punto muy específico querramos presentar un encabezado 1 con un estilo diferente. Como véis, las posibilidades y combinaciones son infinitas.

Sintaxis de los comandos CSS.

Una regla de CSS está compuesta de un selector y un bloque de declaración que se aplica, por supuesto, al selector (o selectores) en el inicio. El bloque de declaración debe comenzar con un corchete de apertura "**{**" y uno de cierre "**}**" pudiendo ocupar varias líneas con propósitos de claridad en nuestro código.

```
Selector                                                    De
claración            1                                  Declaració
n 2      Declaración "n"   .h1      {    color:blue   ; font-size:12px   ;
                }          propiedad:valor   propiedad:valor
```

Si os preocupáis por agregar cantidades innecesarias de caracteres os recomendamos que a vuestro código fuente le apliquéis una "precompilación" antes de subirlas a vuestro **servidor en producción**, el proceso se basa en eliminar los espacios innecesarios a fin de obtiner un código compacto; no obstante si vosotros optáis por comprimir los datos antes de enviar de vuestro servidor web el tema de los espacios en blanco y retornos de carro no tendrá mayor incidencia

porque como se repiten tanto son candidatos ideales para comprimirlos.

Definiendo nuestro primera cuadrícula.

A nuestro código base le vamos a hacer las siguientes modificaciones en el código HTML:

```
A

B

C

D

E

F
```

Y en el código CSS le colocaremos los siguientes valores:

```
body {    margin: 40px;  }    .envoltorio {    display: grid;    grid-tem
plate-columns: 100px 100px 100px;    grid-gap: 10px;    background-color:
 #fff;    color: #444;  }    .caja {    background-color: black;    color
: green;    border-radius: 7px;    padding: 15px;    font-size: 150%;  }
```

Fijémonos en la clase *envoltorio* y sus valores entre corchetes:

- "**display: grid**": indica al navegador que deseamos una "dibujar" una rejilla.
- "**grid-template-columns**": indicamos el ancho de cada columna (e implícitamente el número de ellas).
- "**grid-gap**": acá indicamos el espacio que queremos dejar entre cada una de las pistas de cuadrícula.

Resultado ejemplo N° 1.

El resultado es el siguiente (probadlo en vuestro **Mozilla Firefox Nightly**):

Actualizado el jueves 9 de marzo de 2017.

Ya tenemos instalado el Mozilla Firefox 52:

https://twitter.com/ks7000/status/839996433726095360

El código de este ejemplo debería ser mostrado correctamente, si tenéis la versión 52 instalada en vuestro ordenador (esto lo logramos en **Wordpress** colocando estilo **CSS** directamente dentro de la línea de cada instrucción **HTML**):

A

B

C

D

E

F

Especificando un orden exacto de los elementos.

Para este ejemplo aprovecharemos que hemos "enumerado" las celdas de nuestra cuadrícula con las letras del abecedario, vamos a establecer nuestro siguiente tinglado: ABC/DEF --> DAF/EBC.

Código HTML.

Para este ejemplo reutilizaremos el código mostrado en el archivo base, no redundaremos, subid y mirad por favor.

Código CSS.

Para este ejemplo reutilizaremos el ejemplo anterior *pero agregaremos al final el siguiente código que especifica el orden en que queremos mostrar los elementos.*

```
.a {          grid-column-start: 2;          grid-column-end: 3;
 grid-row-start: 1;          grid-row-end: 2;      }      .b {          g
rid-column-start: 2;          grid-column-end: 3;          grid-row-start
: 2;          grid-row-end: 3;      }      .c {          grid-column-star
t: 3;          grid-column-end: 4;          grid-row-start: 2;          g
rid-row-end: 3;      }      .d {          grid-column-start: 1;
grid-column-end: 2;          grid-row-start: 1;          grid-row-end: 2;
      }      .e {          grid-column-start: 1;          grid-column-end
: 2;          grid-row-start: 2;          grid-row-end: 3;      }      .f
  {                grid-column-start:  3;                  grid-column-
end: 4;          grid-row-start: 1;          grid-row-end: 2;      }
```

Fijaos bien en la numeración de los siguientes elementos CSS:

- grid-column-start
- grid-column-end
- grid-row-start
- grid-row-end

Resultado ejemplo N° 2.

Usando código abreviado para especificar un orden exacto.

En este ejemplo obtendremos el mismo resultado del ejemplo N° 2 **pero utilizando un código abreviado: en vez de utilizar un comando de inicio y otro de cierre** usaremos un solo comando acompañado de los valores numéricos de inicio y fin separados por una barra invertida, veamos:

```
  .a {      grid-column: 2 / 3;      grid-row: 1 / 2;    }    .b {      g
rid-column: 2 / 3;      grid-row: 2 / 3;    }    .c {      grid-column: 3
 / 4;      grid-row: 2 / 3;    }    .d {      grid-column: 1 / 2;      gr
id-row: 1 / 2;    }    .e {      grid-column: 1 / 2;      grid-row: 2 / 3
;    }    .f {      grid-column: 3 / 4;      grid-row: 1 / 2;    }
```

Es de hacer notar que en el mismo orden que los escribamos en el código CSS, así mismo será progresivamente dibujado en pantalla por el navegador web. *Si cometemos algún error (por ejemplo, asignando o superponiendo celdas) será mostrada el último elemento que haya ocupado la celda(s) en cuestión.* También veremos que un elemento podrá ocupar una o más celdas contiguas.

Utilizando un solo comando para especificar orden exacto.

De nuevo obtendremos el mismo resultado del ejemplo N° 1 y 2 *pero con un solo comando:* **grid-area**. Para ello le pasaremos los parámetros de la siguiente manera (separados por una barra invertida "/"):

- Fila de inicio
- Columna de inicio.
- Fila de finalización.
- Columna de finalización.

Código CSS.

Como es un solo comando, por elegancia usaremos una sola línea para cada uno de ellos:

```
    .a { grid-area: 1 / 2 / 2 / 3; }      .b { grid-area: 2 / 2 / 3 / 3; }      .c { grid-area: 2 / 3 / 3 / 4; }      .d { grid-area: 1 / 1 / 2 / 2; }         .e { grid-area: 2 / 1 / 3 / 2; }         .f { grid-area: 1 / 3 / 2 / 4; }
```

Colocando elementos que ocupen más de una celda.

Para este ejemplo N° 4 agregaremos dos elementos más: **G** y **H** para hacer notar que sucede si un elemento ocupa más de una celda. Haremos que el elemento **A** ocupe dos celdas horizontales contiguas (celda 1-1 y 1-2) y el elemento **B** dos celdas verticales contiguas (celda 1-3 y 2-3).

Código HTML.

A

B

C

D

E

F

G

H

Código CSS.

Solo especificaremos una ubicación específica a los elementos **a**, **b**, **c** y **d** pero notad como son desplazados los elementos restantes **e** hasta la **h**:

```
    .a { grid-column: 1 / 3; grid-row: 1    ;}      .b { grid-column: 3
   ;  grid-row:  1  /  3;}          .c  {  grid-column:  1       ;  grid-
row: 2    ;}      .d { grid-column: 2    ; grid-row: 2    ;}
```

Hemos formateado el código de tal manera que notéis rápidamente la diferencia: hemos omitido el valor final.

```
      .a  {  grid-column:  1  /  3;  grid-row:  1  /  2;}         .b  {  grid-
column: 3 / 4; grid-row: 1 / 3;}
```

Resultado ejemplo N° 4:

Percatad que es indiferente si al código CSS le eliminamos los elementos **c** y **d**: automáticamente el navegador web los coloca en la celda inmediatamente disponible ¡y el resto de los elementos también!

Colocando elementos que ocupen más de una celda con la palabra clave "span".

El verbo "to span" en inglés significa "extender" en castellano, pues bien, podemos especificar el inicio (columna o fila) *pero especificaremos cuantas celdas queremos extender.* Esto facilita , mentalmente, las ubicaciones: solo especificamos el inicio y cuantas celdas se extiende.

También haremos un cambio en como se define la cuadrícula con un comando nuevo:

- **grid-template-rows**: especifica el ancho de cada fila (anotaremos tantos valores como filas queremos tener).
- *Si los elementos exceden en cantidad a los valores anteriores, el navegador utilizará los valores por defecto que a bien tenga establecer.*

Código CSS.

```
.envoltorio {   display: grid;   grid-gap: 10px;   grid-template-columns:
 100px 100px 100px;   background-color: #fff;   color: #444;   }  .caja {
   background-color: black;   color: green;   border-radius: 5px;   paddi
ng: 20px;   font-size: 150%;   }     .a { grid-column: 1 / span 2; }   .b
  {  grid-column:  3  ;  grid-row:  1  /  span  2;}      .e  {  grid-
column: 1 / span 3; grid-row: 3 ;}
```

Resultado ejemplo N° 5.

Rejillas y líneas con nombres.

Código en HTML.

Simplificaremos este ejemplo con solamente cuatro elementos y os explicaremos en la sección CSS qué es lo que queremos realizar.

```
A

B

C

D
```

Código en CSS.

Vamos a abstraernos aún más en este punto, ¿recordáis la introducción y que numeramos las líneas de las filas y las columnas? Pues podemos asignarles un nombre a cada una de ellas, p**ero necesitamos un comando o comandos para declarar dichas variables.** De nuevo os presentamos dos instrucciones anteriores:

- **grid-template-columns**
- **grid-template-rows**

Ambas nos permiten almacenar valores en variables pero con una sintaxis especial: los paréntesis rectos o corchetes **"[]"** que permiten asignar el nombre y luego, como hemos visto, podemos especificar el ancho en diferentes unidades (píxeles, porcentajes, ect.); mirad el código, observad bien como os hemos estructurado e indentado las líneas:

```
body {   margin: 40px;  }    .envoltorio {    display: grid;    grid-gap:
 10px;    grid-template-columns:      [col1-ini] 100px      [col2-ini] 10
0px      [col3-ini] 100px      [col3-fin];    grid-template-rows:      [f
il1-ini] auto       [fil2-ini] auto       [fil2-fin];    background-color
: #fff;    color: #444;   }    .caja {    background-color: #444;    colo
r: #fff;    border-radius: 5px;    padding: 20px;    font-size: 150%;  }
   .a {    grid-column: col1-ini / col3-ini;    grid-row: fil1-ini ;  }
.b {    grid-column: col3-ini ;    grid-row: fil1-ini / fil2-fin;  }  .c
{     grid-column: col1-ini;     grid-row: fil2-ini ;  }  .d {     grid-
column: col2-ini ;    grid-row: fil2-ini ;  }
```

Visualización del código, ejemplo N° 6.

A estas alturas imaginamos que ya tenéis la nueva versión de Mozilla Firefox, así que os presentamos **no por imagen sino por código con estilo en cada línea** el resultado de este ejemplo:

A

B

C

D

Rejillas y lineas con nombres extendiendo con palabra clave "span".

Retomamos el ejemplo anterior (el código HTML es el mismo) pero simplificaremos las filas, veamos.

Código en CSS.

```
body {    margin: 40px;  }    .envoltorio {    display: grid;    grid-gap
: 10px;    grid-template-columns: [col1] 100px [col2] 100px [col3] 100px
[col4] 100px ;    background-color: #fff;    color: #444;  }    .caja {
  background-color: #444;    color: #fff;    border-radius: 5px;    paddi
ng: 20px;    font-size: 150%;  }    .a { grid-column: col1 / span 2;}  .b
 { grid-column: col3 / span 2;}  .c { grid-column: col1         ;}  .d {
grid-column: col2 / span 3;}  .e { grid-column: col1 / span 4;}
```

Observemos los nombres que asignamos a cada columna y el cómo podemos referirnos a cada una de ellas. Este ejemplo colocamos nombres que a decir verdad no ahorran trabajo alguno ni ofrecen claridad: **col1** es más largo que usar un simple número 1, *pero acá el concepto es nombrar las columnas para nosotros , mentalmente, ubicarnos mejor.* Es por ello que proponemos cambiar los nombres de la siguiente manera (eliminamos atributos para simplificar y realzar nuestro punto):

```
.envoltorio {   grid-template-columns: [izq] 100px [centro] 100px [centro
] 100px [der] 100px ;  }    .a { grid-column: izq      / span 2;}  .b { g
rid-column: centro 3 / span 2;}  .c { grid-column: izq             ;}  .
d   {   grid-column:   centro   2   /   span   3;}      .e   {   grid-
column: izq      / span 4;}
```

Ya diferenciamos que la primera columna es **izq**, las dos del centro las nombramos iguales y la derecha pues **der**. He aquí que con **izq** y **der** no tenemos ninguna duda de su uso *pero las dos del centro debemos numerarlas debido a que le colocamos el mismo nombre a ambas (celdas **b** y **d**).* En el próximo ejemplo N° 8 ahondaremos más en estos detalles.

Visualizando el resultado del ejemplo, ejemplo N° 7.

A

B

C

D

E

Ejemplo N° 8: comando "repeat".

Ya aprendimos que podemos nombrar las lineas de cuadrícula y de paso les podemos asignar el mismo nombre porque le agregamos un espacio y luego el número de línea de cuadrícula para referirnos a ellas y evitar confusión. El comando que permite hacer esto es el comando **grid-template-columns**, ahora bien, hemos trabajado con pocos elementos de filas y columnas pero imaginemos que son 10, 15 o más filas y/o columnas... ¿cómo podemos definir la malla?

El "chiste" de la programación es ahorrarnos trabajo y nuestra rejilla debe estar definida, así que debemos *escribir todas y cada una de las filas/columnas al navegador.* ***A fin de ahorrarnos trabajo "inventaron" el comando repeat()*** *para colocar entre paréntesis los valores que queremos repetir, y así escribimos menos.*

Finalizando el tema.

Así nos despedimos por el día de hoy, no sin antes presentaros el código que utilizamos para presentar este humidle tutorial que espero os haya gustado y os sea útil a futuro, ¡gracias por vuestra atención!

Código en HTML.

A

B

CSS

D

E

F

G

HTML

Código en CSS.

```
body {   margin: 40px;  }    .envoltorio {    display: grid;    grid-temp
late-columns: 100px 100px 100px;    grid-gap: 10px;    background-color:
#fff;    color: #444;  }    .caja {    background-color: #444;    color:
#fff;    border-radius: 10px;    padding: 20px;    font-size: 150%;  }  .
d { border-radius: 47px; }  .c { background-color: blue;}  .f { border-ra
dius: 20px;}  .h {   grid-column: 2/3;   background-color: orange;  }
```

Actualizado el 14 de marzo de 2017: HTML con estilo CSS en cada línea.

Es más complejo escribirlo así pero es la manera más fácil para que **WordPress** muestre las mallas en Mozila Firefox 52 o superior:

A

B

CSS

D

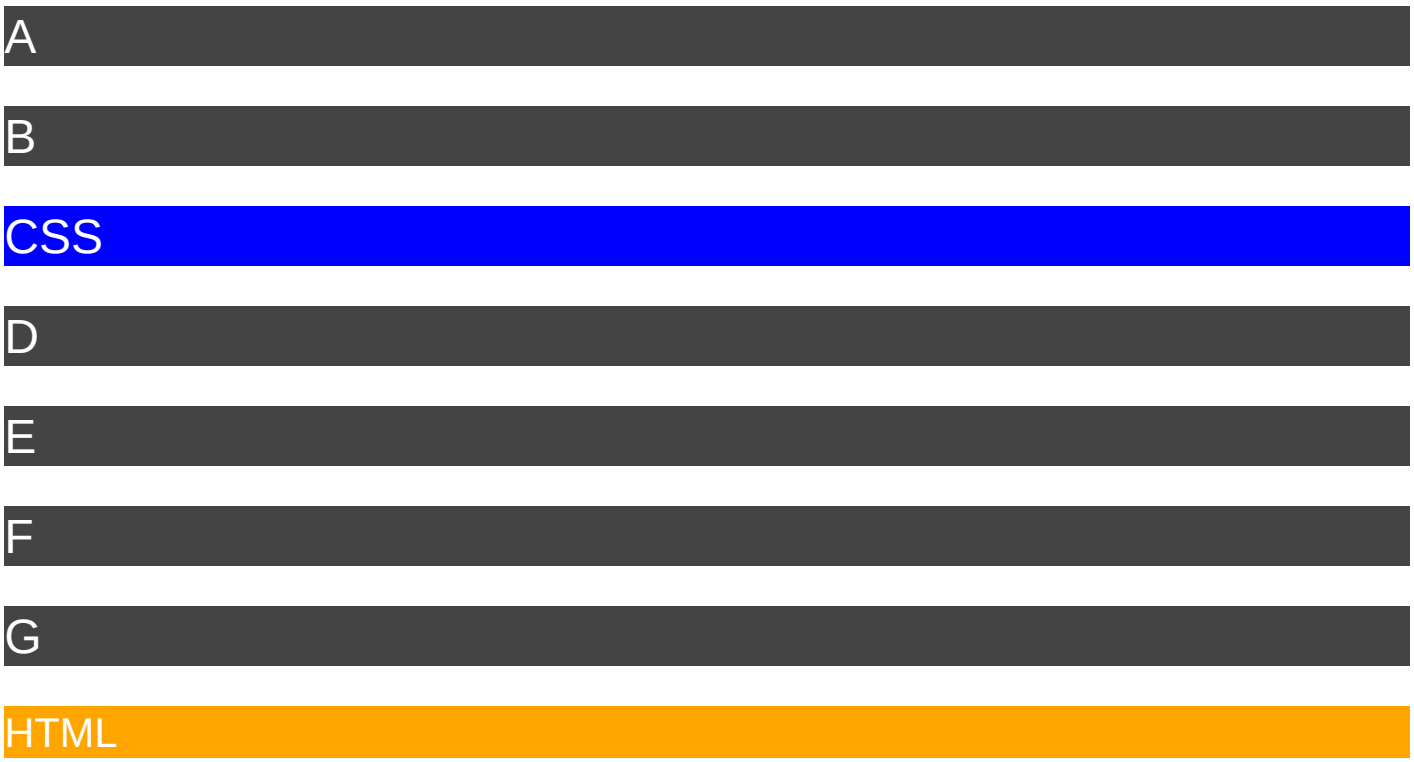
E

F

G

HTML

Visualización si tenéis instalado Mozilla Firefox 52 o superior.



CSS GRID AVANZADO

De la mano de la programadora Diana Aceves, española, os dejo un vídeo de la prorgamación de CSS Grid que consideramos avanzado:

https://www.youtube.com/watch?v=p7oXrr9yjXY

Fuentes consultadas.

En idioma castellano.

- "Definición de HTML".
- "" en "Mozilla Developer Network MDN".

En idioma inglés.

- "Cascading Style Sheets" at Wikipedia.
- "HTML5 Definition Complete, W3C Moves to Interoperability Testing and Performance" by W3.org
- "CSS Grid Layout Module Level 1"
- "HTML Tutorial" as W3Schools.com
- "CSS Tutorial" at W3Schools.com
- "Install Firefox Nightly on Ubuntu" by Matt_G.
- "LLVM Debian/Ubuntu nightly packages" at The LLVM Foundation.
- "CSS Syntax" at w3schools
- "Grid by Example" by Rachel Andrew.
- "CSS Grid demo" by Moz://a.

Youtube's tutorials:

CSSconf EU 2014 | Rachel Andrew: CSS Grid Layout

https://www.youtube.com/watch?v=GRexIOtGhBU

En idioma francés:

- "Les concepts de base des grilles CSS" at MDN.

Agradecimiento público por difundir el conocimiento:

https://twitter.com/Real_CSS_Tricks/status/838147042031644673

https://twitter.com/mozilla_hispano/status/841278925527384065

https://twitter.com/mozillaVe/status/841278927645560832

https://twitter.com/david_luna/status/841498666418470914