

EJEMPLO DE DISEÑO RESPONSIVE CON GRID

La caja de herramientas que llevamos vistas para maquetación no está nada mal. Pero si te fijas, el posicionamiento de elementos con CSS se convierte en la mayoría de ocasiones en un sopesar los efectos colaterales que cada técnica empleada genera hasta ver la que mejor nos encaja. De ahí que la demanda del mercado para disponer de más y mejores tecnologías no haya cesado. Al igual que se investiga con técnicas quirúrgicas buscando ser más efectivas y con menos efectos secundarios para el paciente.

Algo parecido les va a pasar a los maquetadores cuando empiezan a estar hasta el gorro de la propiedad ya conocida de CSS como es **float**, al tener que enfrentarse con ciertas distribuciones de elementos mucho más complicadas: desde el control preciso de las dimensiones verticales de la rejilla y el modo de distribuirse automáticamente al cambiar las resoluciones, hasta poder cambiar de distribución por completo una página sólo con un pequeño cambio de CSS, sin tocar el contenido.

Para dar cobertura a estos casos y muchos otros de maquetación compleja, la W3C diseñó unas tecnologías como **Flexbox** y **Grid**. Ambas tecnologías no siendo excluyentes entre sí, más bien se pueden complementar porque, de hecho, cuando así ocurre se les puede sacar más potencia al combinarse para lograr los mejores resultados. En el siguiente ejemplo práctico puedes comprobarlo [<https://www.silocreativo.com/flexbox-vs-css-grid-ejemplo-practico/>].

Existen por ahí cantos de sirena al estilo de esas publicidades que te venden aprender algo sin esfuerzo como puede ser Bulma [<https://bulma.io/>] donde proclaman eso de "hágaselo *responsive* sin tener ni idea de CSS". Lo dejo ahí para atrevidos.

Ahora con Grid

Partimos de recordar que con Flexbox sólo es posible controlar cómo se alinearán y colocarán elementos en una fila o en una columna, pero no lograr que cada elemento se expanda a través de fila y columna a la vez, si fuese necesario. CSS Grid sí permite este extremo al presentarse como una rejilla bidimensional.

Cabe insistir que los sistemas de Flexbox y CSS Grid no compiten entre sí. Más bien se utilizan para dar solución a problemas distintos (y a veces complementarios), por lo que ninguno viene para sustituir al otro, y es conveniente conocer ambos para sacar el mayor provecho de cada uno de cara a aumentar la calidad del resultado para el proyecto que se tenga entre manos.

Al igual que ocurría con Flexbox, la propiedad fundamental que convierte un contenedor en gestor bidimensional de elementos hijo con Grid es [**display: grid;**]. Un par de conceptos muy importantes en los que se basa este sistema son las líneas y las áreas que dan carta de naturaleza a la rejilla y dan forma al contenedor. Aunque dicha rejilla es virtual, cada línea limitadora de fila y columna tiene por defecto un número empezando por el 1 que indica en qué posición está la línea, sin embargo, puedes especificar al gusto los nombres de estas líneas.

Y claro está, para trabajar con esas líneas y áreas, el sistema Grid tiene sus propiedades para disponer el contenedor. Para empezar, nos centramos la manera de configurar la altura de las filas de la rejilla se hace con **grid-template-rows** y la anchura de las columnas con **grid-template-columns** pudiendo ser sus medidas fijas (px) o proporcionales (fr). En la dirección [https://codepen.io/origamid/pen/ZyRYQp] tienes una muestra visual de lo que pasa al definir las columnas tanto con medidas fijas como proporcionales. Lo mismo para las filas en [https://codepen.io/origamid/pen/gRKpow].

Hay que agradecer a Sarah Edo este simulador para crear rejillas Grid (te da el código hecho)
<https://cssgrid-generator.netlify.app/>

Recordando una vez más que Grid es un sistema bidimensional, existe la posibilidad de trabajar por áreas para lo cual se incluye la propiedad **grid-template-areas** que permite etiquetar por nombre la zona o zonas destinadas a ubicar los elementos con información, lo que experimentarse [https://codepen.io/origamid/pen/owyjxz]. La siguiente propiedad atajo, **grid-template**, resume en una las propiedades de estas tres últimas y puede practicarse en la dirección [https://codepen.io/origamid/pen/NgzGOJ].

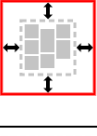
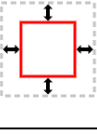
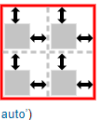
También existen unas propiedades para indicar la disposición de los elementos hijo dentro de las rejillas. Ello se define con **grid-column** cuya visualización y funcionamiento se muestran en [https://codepen.io/origamid/pen/JJZVNr] y **grid-row** quedando igualmente reflejado en la dirección [https://codepen.io/origamid/pen/dRKBKv]. De las dos anteriores derivan **grid-column-start** / **grid-column-end** y **grid-row-start** / **grid-row-end** indicando donde empieza/termina la columna/fila. La siguiente propiedad atajo, **grid-area**, resume en una las propiedades de estas cuatro últimas y puede practicarse en la dirección [https://codepen.io/origamid/pen/eRKwaN].

Los amigos de Codepip crearon este juego para aprender estas propiedades
<https://cssgridgarden.com/#es>

Una vez comprendido esto, se puede utilizar como documento de consulta rápida y visual el existente en la dirección [https://css-tricks.com/snippets/css/complete-guide-grid/#grid-properties].

Existe otro grupo de propiedades para gestionar contenedores Grid que dotan de mayor automatismo la incorporación de nuevos elementos hijo al contenedor. A esta técnica se le conoce como rejilla implícita dado que no estamos aplicando propiedades CSS a la vista de los elementos hijo que hay sino que lo preparamos para que, vengan los que vengan, se dispongan según el patrón definido. Para hacer esto, se han dispuesto las propiedades **grid-auto-rows** que se recrea en la dirección [https://codepen.io/origamid/pen/mwKPYR], **grid-auto-columns** igualmente trabajable en [https://codepen.io/origamid/pen/eRgJrLv], así como **grid-auto-flow** que define el flujo de los elementos hijo en [https://codepen.io/origamid/pen/LLrZpN]. Proporciona también una propiedad atajo que sirve para reunir las *grid-templates* y *grid-auto* en una sola como es **grid** a secas. Código de ejemplo al canto para divertirse en la dirección [https://codepen.io/origamid/pen/RgJRLl].

Por último, al igual que ocurría en Flexbox, existen las propiedades que aplicadas al contenedor que permiten ubicar los elementos hijo **justify-content**,

Common	Axis	Aligns	Applies to
'justify-content'	main/inline	content within element (effectively adjusts padding)	 block containers, flex containers, and grid containers
'align-content'	cross/block		
'justify-self'	inline	element within parent (effectively adjusts margins)	 block-level boxes, absolutely-positioned boxes, and grid items
'align-self'	cross/block		
'justify-items'	inline	items inside box (controls child items 'justify-self: auto')	 block containers and grid containers
'align-items'	cross/block		

align-content y **align-items** con similar funcionalidad pero ahora incluye una más denominada **justify-items**. Si ello le unimos las de aplicación a los elementos hijo tenemos las llamadas **justify-self**, **align-self** y **place-self**.

La secuencia de códigos de ejemplo que acompaña a esta batería tiene por objeto que pruebes cada uno de ellos e identifiques en su código las propiedades de Grid que se han ido exponiendo hasta aquí para asimilar su funcionamiento. Una buena manera de hacerlo es utilizando las **Herramientas de Desarrollador** del navegador (Firefox y Chrome) para ver cómo afecta cada propiedad al diseño confeccionado.

Ejemplos interactivos reunidos (en lengua portuguesa)
<https://origamid.com/projetos/css-grid-layout--guia-completo/>

Además tienes a tu disposición un ejemplo básico desarrollado paso a paso en la dirección [<https://javadesde0.com/css-grid-layout/>] además de un primer minitutorial básico en [<http://developinginspanish.com/2018/04/29/aprende-grid-de-css-en-5-minutos/>] y un segundo minitutorial de refuerzo con ejemplos en [<https://www.ionos.mx/digitalguide/paginas-web/creacion-de-paginas-web/css-grid-layout/>].

Para los muy cafeteros:
<https://gridbyexample.com/examples/example3/>

Te propongo a continuación un ejercicio curioso desarrollado con tecnología Grid en el que puedes aprender bastante:

Tablero de Formación Certificada



Acabó en Diocesianas-Arriaga (Vitoria) sus estudios de **Formación Profesional en Informática** en 1985 y posteriormente en la Universidad de Sevilla los de **Ingeniero Técnico en Informática de Gestión**

1985 / 1990



Especialista en Internet y sus Aplicaciones en el Mundo de la Educación y de la Empresa por la Universidad Nacional de Educación a Distancia (UNED)

1996



Coordinador del Proyecto Europeo Comenius "Intercambio de Experiencias Educativas en Informática" SAFA Ntra Sra de los Reyes (Sevilla), La Providence (Amiens-FR) y St. Ignatius College (Enfield-ING)

1996-1999



Curso sobre el **Lenguaje de Marcas XML** a través del portal de Internet ALMAGESTO perteneciente al Grupo Eidos

2001




Experto Profesional en **E-Learning**: Educación y Formación por Internet por la Universidad Nacional de Educación a Distancia (UNED)

2007



Master en Desarrollo de Aplicaciones Empresariales en J2EE por Sun Microsystems y Formación Digital

2008



Curso **Metodologías y Herramientas Orientadas al Aprendizaje Activo** del Alumnado de FP por impartido por Daniel Irazola (Maristas-Durango) enmarcado en la XXVI Escuela de Verano SAFA

2008



Programador Certificado en Java SCJP5 por la empresa Sun Microsystems con una puntuación de 86%

2009



Gestión del **E-Learning** en Moodle: Instalación, Administración y Uso Avanzado impartido por Tadel Formación, impulsado y patrocinado por el FSE y el MICyT

2009




Curso **MY OXFORD ENGLISH** cuyos 10 niveles son compatibles con la certificación B2 diseñada por Oxford University Press

2013




Experto en **Análisis Forense Informático y Pericial** por la Asociación de Peritos Judiciales Tecnológicos de Andalucía

2019



Programa Avanzado en **Python** por el Instituto Tecnológico Telefónica para el desarrollo de APIs JSON en DJANGO y FLASK

2020




Cursos sobre **Java, Básico de Android, Kotlin, Javascript, Typescript y C# (.NET)** por la plataforma CODECADEMY

2019-2021



Cursos sobre **Laravel5, Colecciones Java, Java y XML, Clean Code y Javascript** por la plataforma OPENWEBINARS

2019-2021



Especialista en **Maquetación Responsive** con HTML5, Flexbox, CSS Grid y Bootstrap por la plataforma campusMVP

2021