silocreativo.com

Ejemplo de Landing Page con CSS Grid: Apollo 8 (I) • Silo Creativo

Alfonso Serrano

8-11 minutos

Desde hace unos meses, desde Silo Creativo estamos siguiéndole la pista a CSS Grid Layout, el nuevo sistema en el que podremos controlar y posicionar los elementos de nuestra web en dos dimensiones mediante CSS, sin tener que modificar nuestro HTML. Y es que esta nueva herramienta viene para **intentar revolucionar el diseño web que se viene haciendo en los últimos años**. Algo que ya se hizo, aunque personalmente creo que a menor escala, con la llegada de Flexbox, del cual hicimos una comparación con CSS Grid hace unos meses.

Así que, en las dos partes en las que se dividirá este artículo, queremos realizar un ejemplo que podríamos llamar casi de producción, y que sirva como final a una serie de **artículos de introducción a CSS Grid que hemos estado escribiendo en Silo Creativo**. Por lo que vamos a realizar una landing page de una página ficticia, en CodePen, que sirva de homenaje al 30 aniversario del lanzamiento del Apollo 8, la primera misión tripulada en salir de órbita terrestre, llegar y orbitar a la Luna y finalmente regresar a la Tierra. Y es que como ya habéis podido ver en algunos de nuestros calendarios, en Silo Creativo somos amantes

de la exploración espacial.

En este primer texto, nos centraremos en el diseño previo de la landing page, veremos el concepto que hemos realizado en Affinity Designer, y mostraremos cuales serán las pautas a seguir para cada una de las secciones en que hemos dividido la web. Es decir, un boceto de la rejilla a usar en el diseño web, como va a cambiar según el dispositivo en donde la veamos y su anchura, etc. **Y en la siguiente parte, nos meteremos de lleno en la escritura del código de la web usando CodePen**, y veremos como solucionar todas las propuestas que hemos realizado en esta primera parte.

Antes de empezar, os recomendamos (si aun no lo habéis hecho), que **echéis un vistazo a nuestros artículos de introducción a CSS Grid**, en los que hemos estado repasando las diferentes propiedades.

- Introducción y primeros pasos con CSS Grid (I)
- Introducción y primeros pasos con CSS Grid (II)

También me gustaría mencionar que en Silo Creativo estuvimos trabajando en un side project en el que poner en marcha nuestros conocimientos sobre CSS Grid, y donde queríamos ponerlos a prueba en una web real. Y de ahí surgió GridExamples, un escaparate **showcase donde recopilamos ejemplos reales de webs que actualmente están usando CSS Grid**, y que nos pueden servir de guía e inspiración para que lo incorporemos a nuestro flujo de trabajo. Porque en estos momentos, esta es una de las preguntas que se hacen muchos desarrolladores, estudios y empresas. ¿Puede mi web dar el salto a CSS Grid hoy en día?

Pero, ¿puedo usar Grid en producción actualmente?

Si hace unos meses ya os mostramos la tabla de compatibilidades de CSS Grid con los distintos navegadores web, no está de más volverla a repasar. Y es que durante este tiempo, la compatibilidad es prácticamente total, con pequeñas excepciones.

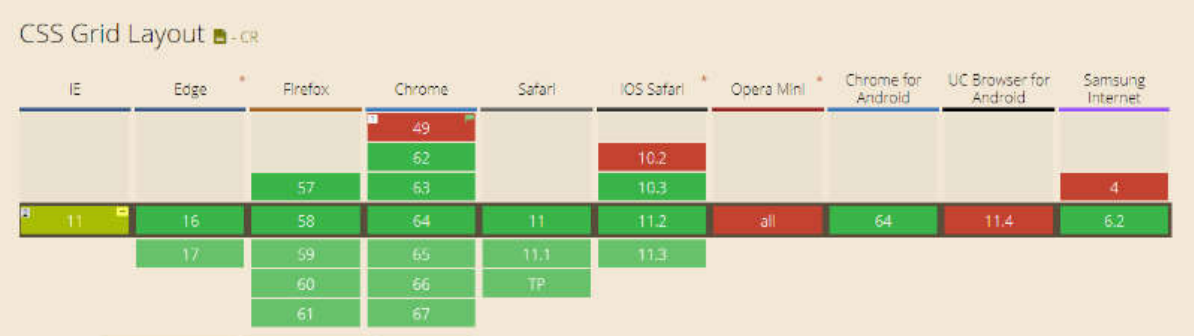


Tabla de compatibilidad actual de CSS Grid Layout con los navegadores más usados.

Como veis, a día de hoy, incorporar CSS Grid en nuestro trabajo es completamente seguro, y no nos encontraremos ningún tipo de problema en los navegadores más usados. Chrome se actualizó hace algo menos de un año, al igual que Firefox y Safari. Y con respecto a navegadores móviles, hace poco más de un mes es compatible con Chrome para Android, y con Chrome y Safari para iOS desde otoño del 2017. **La única incompatibilidad entre los navegadores más usados la encontraremos en Opera Mini.** Por lo que como veis, el porcentaje de usuarios que no verán correctamente una web con CSS Grid será muy reducida.

Creación de la Landing Page de Apollo 8

En primer lugar, y como solemos hacer con nuestros proyectos, hemos **creado un prototipo de la web que queremos hacer de forma previa a escribir la primera línea de código**. Para esto,

hemos usado Affinity Designer, que en estos meses se ha convertido en una de mis herramientas favoritas para comenzar a diseñar una web. Si quieres saber más sobre Affinity Designer y como trabajar con él si te dedicas al diseño web, en Silo Creativo hablamos sobre él y vimos como crear artboards, usar guías y cuadrículas, insertar recursos y aplicar restricciones.

Uno de los motivos de crear un prototipo previo, aunque sea en papel, de nuestro diseño web, es **dejar definida desde un primer momento nuestra rejilla de trabajo**. Está la base donde empezaremos a mover y asignar los elementos de nuestro diseño.

Diseño previo de la web en Affinity Designer

Como se ve en la imagen, es un diseño muy sencillo, dividido en tres partes. La primera es un *hero*,donde situaremos algunos textos y una imagen. Que **según vayamos disminuyendo la**

anchura del dispositivo, se irán recolocando en diferentes posiciones, y en las que daremos uso de media queries. La segunda es una galería de fotografías, con cuatro tamaños de imágenes diferentes, que se posicionarán de forma automática, esta vez sin hacer uso de *media queries*. Para este diseño previo no hemos usado las fotografías finales que llevará la landing page, sino diferentes colores, que nos ayudarán a ver de un primer vistazo el tamaño de cada una de ellas. Y por último, un *footer* con un pequeño texto y algunos iconos sociales.

A diferencia de otros ejemplos que hemos realizado anteriormente, en esta ocasión, no tendremos las diferentes secciones de la web organizadas a partir de un grid. Esta vez, las tres secciones irán apiladas una tras otra, y cada una de ella tendrá una rejilla particular. De esta manera, veremos las diferentes opciones que nos ofrece este layout.

Sección *Hero*

Siguiendo un orden descendente, vamos a centrarnos en primer lugar en la sección superior de nuestra web. Por si en el diseño realizado en Designer no nos ha quedado muy claro de como se comporta la rejilla, vamos a ver mediante un pequeño esquema su organización.

desktop : 1400px tablet : 768px mobile : 320px

Como habéis podido comprobar, vamos a usar en esta parte la

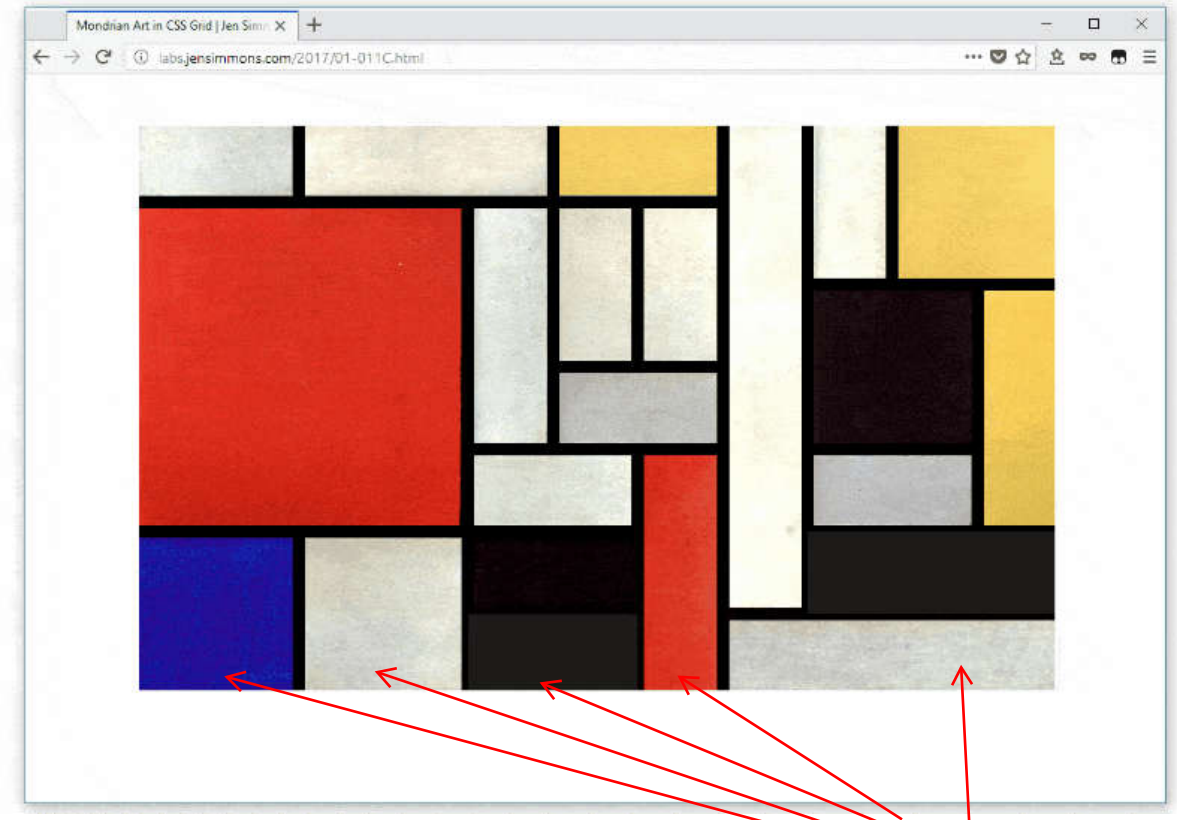
propiedad *grid-area*, que es una de mis favoritas, ya que trabajamos con un código muy visual. Una vez creada la rejilla, con sus correspondientes filas y columnas (*rows* y *columns*), asignaremos a cada elemento de nuestro *grid container* un nombre, mediante la propiedad *grid-area*. **Los nombres que vamos a usar son los mismos que vemos en el esquema, para que cuando veamos el código, sepamos exactamente a que elemento nos referimos de un simple vistazo.** Y para el responsive, haremos uso de los *media queries* como hemos mencionado anteriormente. Por lo que en cada *media query* vamos a crear una rejilla diferente, cada una con su *grid-area* específica.

Aunque a primera vista pueda parecer mucha tarea, veremos que crear una rejilla y asignar elementos a su *grid-area* correspondiente es una acción que realizaremos de forma muy sencilla. Esto apenas supondrá escribir un par de líneas de código.

Sección *Galería*

En el diseño de la sección de galería solo hemos especificado el tamaño que tendrá cada imagen, y no donde irá cada una de ellas por una sencilla razón. Y es que vamos a usar **otra de mis propiedades favoritas de CSS Grid: *grid-auto-flow: dense***. Esta propiedad lo que hace es que, una vez que asignemos a cada imagen es el espacio o *track* que debe ocupar dentro de la rejilla: 1×1, 2×1, 1×2 o 2×2. CSS Grid automáticamente los redistribuirá para que no quede ningún espacio en blanco, y todos estén ocupados por alguna imagen. Por lo que no vamos a hacer uso de media queries en esta sección, ya que conforme vayamos modificando la anchura de nuestro contenedor, **las imágenes irán ocupando de forma automática los espacios que vayan**

quedando libres según sea el tamaño que les hayamos definido. Y para marcar a que tamaño exacto debemos empezar a reducir columnas según el ancho de la web, haremos uso de la propiedad *minmax*.



Comportamiento de la propiedad ***grid-auto-flow: dense***

Un gran ejemplo de este uso lo tenemos en la web de Jen Simmons, a la que os recomiendo seguir de cerca. En el apartado de Labs, tiene un ejemplo de un cuadro de Mondrian responsive en el que hace uso de estas propiedades, y que intentaremos replicar en nuestra galería de fotografías espaciales

Sección *Footer*

Quizás sea la sección más sencilla, pero no quería olvidarme de ella, ya que **además del uso de CSS Grid, usaremos algunas propiedades de Flexbox**. A pesar de que podríamos perfectamente no usarlo, en algunas problemáticas que afectan a

diseños en 2 dimensiones, como las alineaciones de textos. Aquí, simplemente, tendremos dos elementos alineados a cada margen del contenedor. Que llegados a cierta anchura mínima, se apilarán uno sobre otro.

Conclusiones

Una vez hemos realizado todo este proceso de diseño previo, y **sabiendo exactamente como queremos que se comporte nuestra web en cualquier situación**. Solo nos queda ponernos manos a la obra y empezar a escribir código. Algo que, con CSS Grid, no nos va a suponer ningún quebradero de cabeza, ya que en pocas líneas vamos a resolver el diseño de nuestra *landing page*.

Os esperamos para nuestro próximo artículo, y mientras tanto, os animamos a que realicéis en papel o de forma digital vuestros propios diseño. Y así tenerlos listos en el momento que entremos en profundidad a trabajar con CSS Grid!