silocreativo.com

Diseño Editorial y CSS Grid: Inspiración y ejemplos • Silo Creativo

Ricardo Prieto

7-8 minutos

Los maquetadores web estamos de enhorabuena. Nunca nos habíamos encontrado en un momento como el actual, donde poder elegir entre tres **diferentes grupos de sintaxis CSS** para diseñar el layout de un proyecto web. Me estoy refiriendo a floats, Flexbox y CSS Grid (o Grid Layout). Con los tres podemos conseguir casi el mismo resultado final, por lo que la elección dependerá de nuestras habilidades con uno u otro y con la idoneidad del código generado (por temas de optimización, escalabilidad futura, compatibilidad de navegadores, etc…)

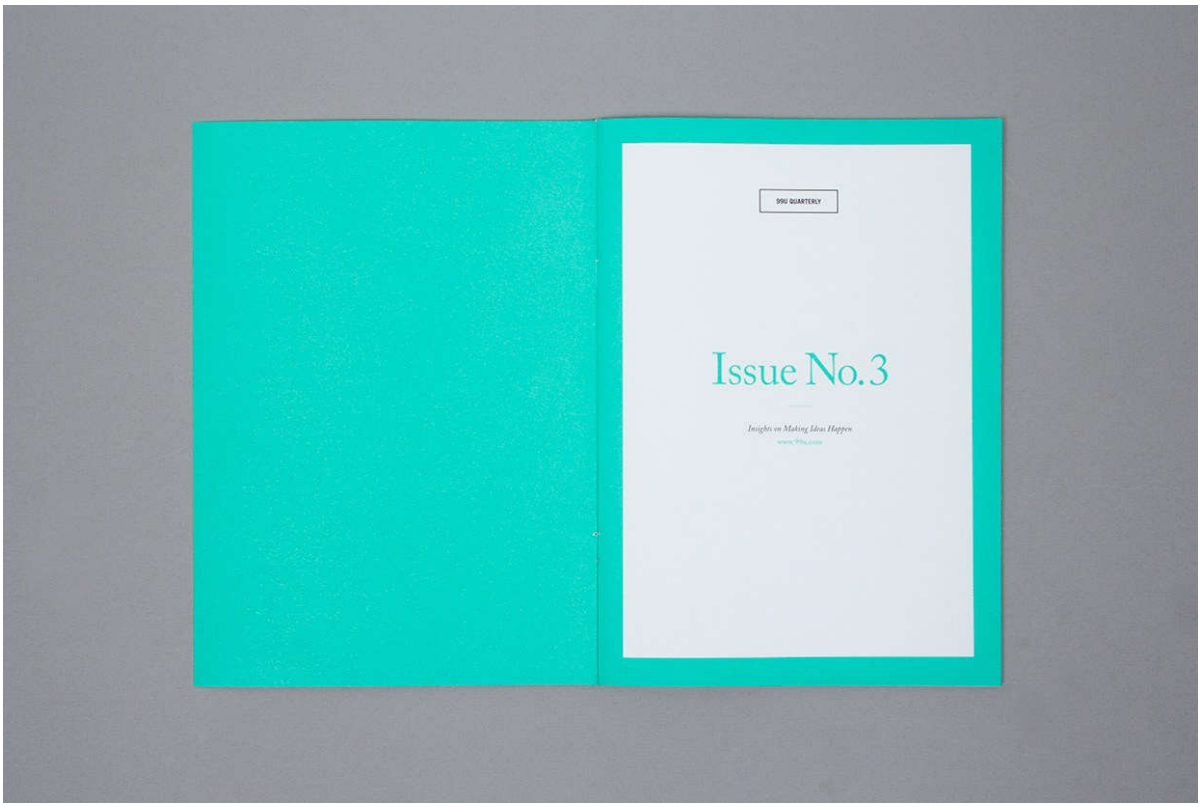
Sin embargo creo que CSS Grid incorpora un lógica de trabajo distinta al resto (Floats y Flexbox). Mientras que con los anteriores **trabajábamos apilando bloques** como aquellos juegos de construcción que usábamos cuando eramos niños (ya fueran en filas o columnas), con CSS Grid el método cambia, pues los bloques ya no tienen peso y tampoco se apilan. Es algo más parecido al corcho que teníamos colgado en la pared de nuestra habitación de adolescentes (por seguir con la misma referencia) y donde colocábamos con chinchetas notas, dibujos y demás logros.

Todo esto hace que aparezca en CSS Grid la posibilidad de **definir**

espacios vacíos con la misma intensidad que definimos los espacios llenos, y no de forma automática o por omisión como sucedía con los floats o en Flexbox. Esto afortunadamente nos acerca más a otras formas de composición, especialmente al diseño editorial, donde los espacios llenos y vacíos conviven con el mismo nivel de importancia. Hablábamos hace un tiempo de esto en nuestras tendencias sobre diseño web, con algunos ejemplos.

Maquetación Web de un diseño editorial con CSS Grid

Para ilustrar (y celebrar) este acercamiento entre diseño editorial y diseño web, nos hemos propuesto maquetar con CSS Grid algunas páginas del número 3 de la revista 99U Quarterly Magazine, una publicación muy interesante para la comunidad creativa.



El primer ejemplo es la portada de la revista (no confundir con la cubierta, aquí las diferencias).

Mantenemos el diseño a dos hojas, pero en resposive ya no necesitamos la hoja de la izquierda, por que desaparece, mientras que los tamaños del título disminuyen:

Como punto interesante, entre las resoluciones de 800px y 1200px tenemos un tamaño de fuente flexible para el título, referenciado a las dimensiones del viewport con la unidad `vw` (viweport width) y modificado con `calc()`:

```
@media (min-width: 800px) and (max-width: 1200px)
{
  .right-page .title {
      font-size: calc(0.15 * (50vw - 120px));
  }
}
```

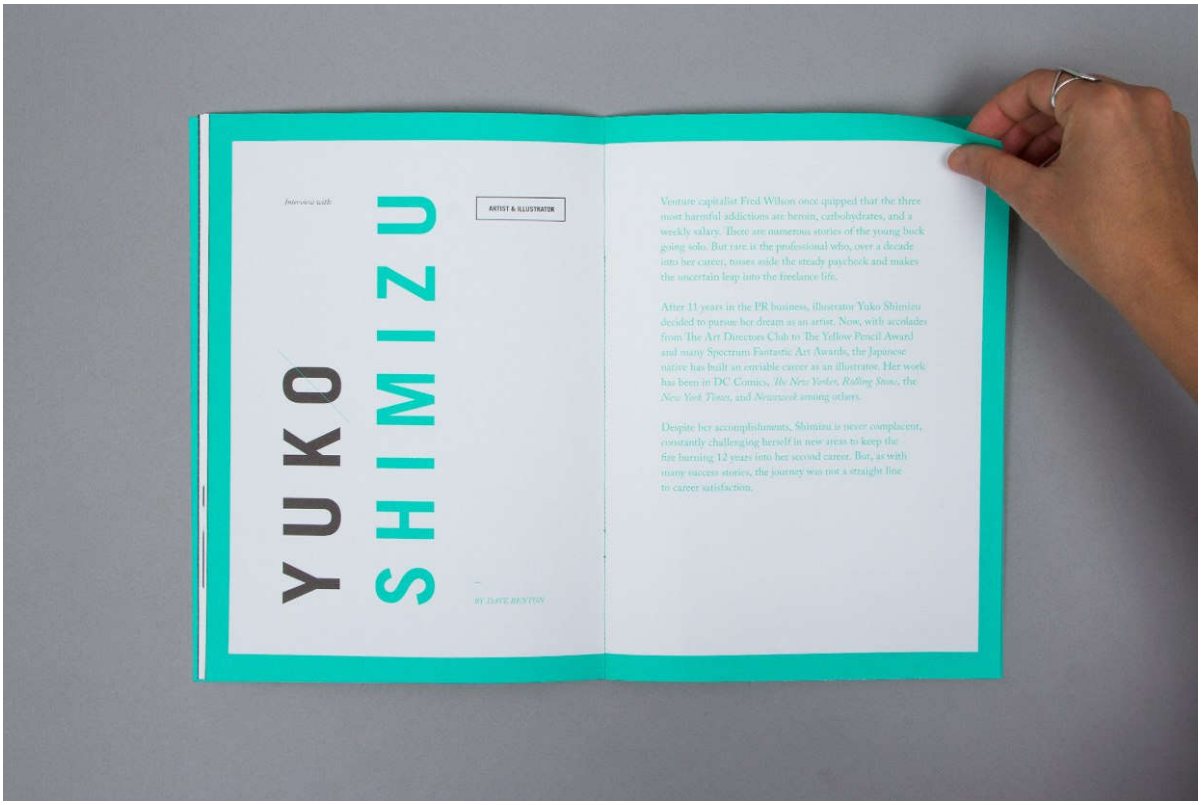
También ofrece muchas posibilidades la función `minmax()` que nos permite **definir en una declaración CSS Grid un valor mínimo y máximo**. En nuestro caso lo hemos usado para definir la dimensión de la fila en el template de nuestro grid, tomando como valor mínimo 100vh (el alto del viewport) y como máximo 1fr, para cuando se necesite más espacio, podamos hacer scroll vertical (pantallas con muy poco alto):

```
.container {
  background-color: #00D0C0;
  display: grid;
  grid-template: minmax(100vh, 1fr) / 50% 50%;
}
```

Te animo a que redimensiones el ejemplo para ver los distintos comportamientos.

En el segundo ejemplo hemos trabajado una página con una breve

biografía de una artista:



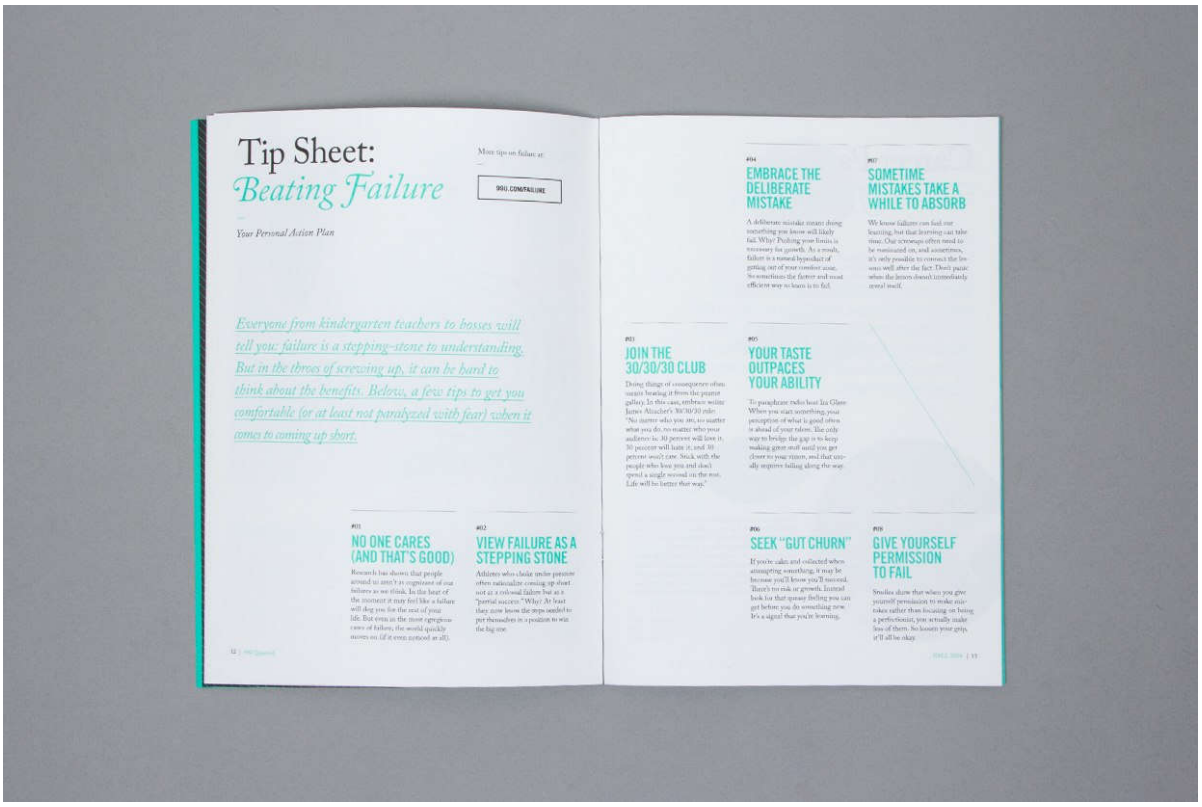
Quizás lo más complejo ha sido el título de la página izquierda, por la **dirección del texto** y también por cómo adaptar su interlineado a las diferentes resoluciones:

Para el tema del título y texto girado en vertical, hemos usado la propiedad `writing-mode` que tiene una compatibilidad correcta entre navegadores, sin embargo el valor que nos serviría para dicha propiedad, `sideways-lr` tiene una compatibilidad muy baja aún, y su uso es solo experimental. Por ello, optamos por girar el texto usando un `transform: rotate(180deg)`. Junto a esto, márgenes, tamaños de fuente e interlineados son d**efinidas con unidades relativas al viewport** para explorar hasta donde podemos llegar con CSS:

```
.left-page .title {
  grid-area: 1 / 1 / 4 / 3;
  text-align: left;
```

```
  writing-mode: vertical-rl;
  transform: rotate(180deg);
  color: #565052;
  text-transform: uppercase;
  align-self: end;
  font-size: calc(0.19 * (100vh - 140px));
  margin: 0;
  letter-spacing: calc(0.06 * (100vh - 140px));
  line-height: 17vw;
  margin-top: calc(-0.07 * (100vh - 140px));
  margin-left: -3vw;
}
```

La tercera sección que hemos maquetado ha sido la página de sumario:



A primera vista es fácil **identificar las líneas guías** que sirvieron al diseñador para distribuir los elementos en el layout. Serán esas

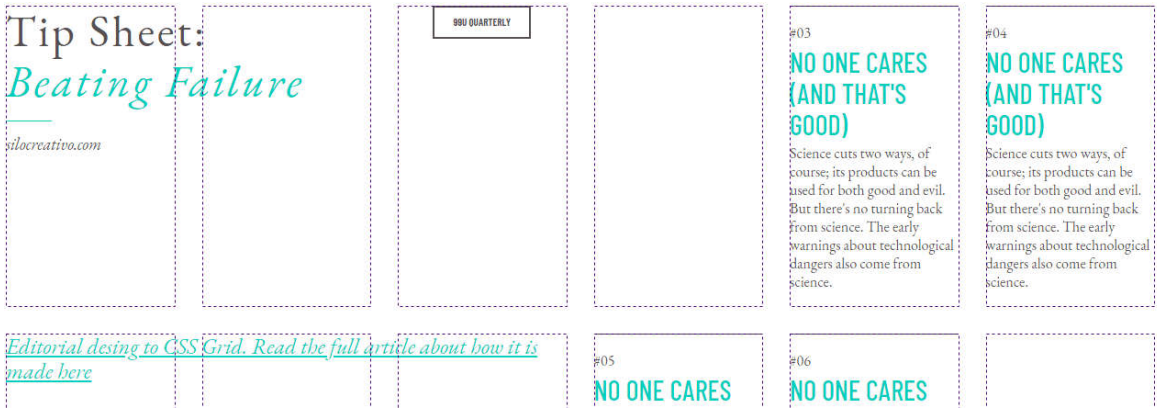
mismas líneas las que utilizaremos para definir nuestro grid:

Interesante cómo podemos ordenar cada uno de los resúmenes simplemente haciendo una selección usando el pseudo-elemento `:nth-of-type(n)`, y posicionándolos en el grid a través de la propiedad `grid-area`. En responsive, vamos a dos columnas, por lo tanto el pseudo-elemento cambia a `:nth-of-type(odd)` y `:nth-of-type(even)`.

```
.hentry:nth-of-type(1) {
  grid-area: 3 / 2 / 4 / 3;
}
.hentry:nth-of-type(2) {
  grid-area: 3 / 3 / 4 / 4;
}
.hentry:nth-of-type(3) {
  grid-area: 1 / 5 / 2 / 6;
}
.hentry:nth-of-type(4) {
  grid-area: 1 / 6 / 2 / 7;
}
.hentry:nth-of-type(5) {
  grid-area: 2 / 4 / 3 / 5;
}
.hentry:nth-of-type(6) {
  grid-area: 2 / 5 / 3 / 6;
}
.hentry:nth-of-type(7) {
  grid-area: 3 / 5 / 4 / 6;
}
.hentry:nth-of-type(8) {
```

```
  grid-area: 3 / 6 / 4 / 7;
}
@media (max-width: 450px) {
  .hentry:nth-of-type(odd),
  .hentry:nth-of-type(even) {
    grid-area: auto / 1 / auto / 5;
  }
}
```

Definido por tanto el grid inicial con seis columnas, es muy sencillo ubicar en entradas.



Declaración CSS Grid con seis columnas

Algunos comentarios sobre CSS Grid, diseño editorial y layouts

Como verás, pasar un diseño editorial a diseño web no es sencillo, aunque **con CSS Grid podemos conseguir un resultado bastante aceptable**. Hay dos problemas principales que separan diseño web del diseño editorial y a día de hoy son insalvables.

En primer lugar, mientras que en **diseño editorial conocemos el formato de salida** (dimensiones físicas de la revista o folleto), **en maquetación web no conocemos de antemanos la resolución de pantalla del usuario**, por lo que el uso de unidades relativas se

hace indispensable no solo para resoluciones menores como veníamos haciendo en diseño web responsive, sino para dispositivos cuyas dimensiones superan lo equivalente al tamaño físico de la revista.

En segundo lugar, **la proporción de la pantalla en diseño web no es la misma que en diseño editorial**, por lo que en muchos casos es necesario un scroll vertical. Además, pese a que tengamos la misma proporción de alto por ancho, necesitamos que la tipografía sea legible en todo momento, por que lo que no es válido reducir el tamaño de todo el diseño incluyendo tipografías por debajo de los niveles admisibles de legibilidad. Quizás los títulos si nos permiten cierta flexibilidad, pero los textos de párrafos deben mantenerse siempre al mismo tamaño.

Pese al trabajo necesario, **es una gran noticia que CSS Grid acerque el diseño editorial al diseño web**, seguro nos servirá a todos los diseñadores como fuente de inspiración para los futuros proyectos 🙂