Using compound grids

There are times when our traditional grids doon’t work as well as we’d want them to, particularly in art-directed content.

Taking the default 12-column grid I use as default we can code it like this:

```
body {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
  grid-column-gap: 2vw;
  grid-row-gap: 2vh;
  align-content: start;
}
```

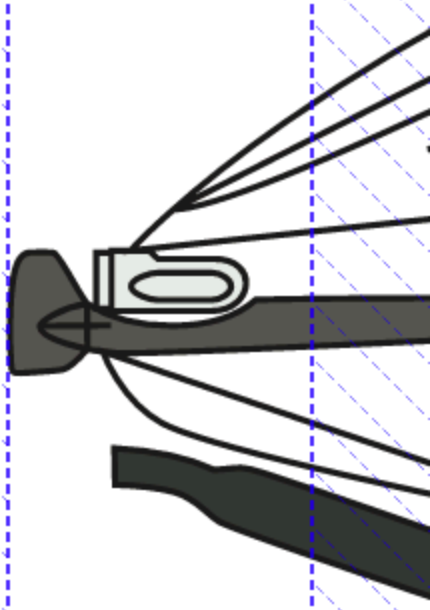
This code is good and it does a lot of the work of grids and layout. But there are times when we need a little more flexibility.

For example, we might want to have space between our main body and the aside portion of the page, something similar to this layout.

Figure 1: Compound Grid Image. Image taken from a page designed and built by Andy Clarke

Notice how there is space that is smaller than the text in the aside element to the left of the image?

If we look at the page using Firefox’s Grid Inspector, we can see that the layout of the page it’s not uniform.



Series 1. 1961–68

Figure 2: Compound grid image viewed with Firefox Grid Inspector. Image taken from a page designed and built by Andy Clarke

The image uses a 12-coolumn grid but it's arranged differently, as shown in the code below:

All the values of `grid-template-columns` should be in one line. I broke them down into rows for readability.

The compound grid breals the content into three areas of 2fr (2 equal portions of the screen width) separated by 2 columns of 1fr (1 equal portion of the screen width)

```
body {
  display: grid;
  grid-template-columns:  2fr
                          1fr 1fr
                          2fr 2fr
                          1fr 1fr
                          2fr;
  grid-column-gap: 2vw;
  grid-row-gap: 2vh;
  align-content: start;
}
```

This gives us flexibility in how we layout the content. without changing it. We can still place content in two 2fr increments or we can make the content narrower by using the 1fr columns as margins.

Using the grid above we can do interesting things. We can have asides with auxiliary information on either side of the main content just by changing the

placement in the grid.

For left-side aside content we could do something like this:

```
aside {
    grid-column: 1 / 3;
    align-self: end;
}
main {
    grid-column: 4 / 8;
}
```

but if we want to flip it to the right, the code changes slightly:

```
main {
    grid-column: 2 /6;
}

aside {
    grid-column: 7 /9;
    align-self: end;
}
```

We can also use this as a further refinement to the grids we discussed in Art Directed Layouts and CSS Grid. We still get 12 columns in our grid but breaking it the way we did gives us additional flexibility to play and experiment.