Multicolumn Layouts

It’s been possible to work in multiple columns of text without having to resort to hacks to make columns work in multiple browsers and in multiple form factors.

According to caniuse.com developers need to pay attention to the following caveats:

- Firefox doessn’t support the break-before, break-after, break-inside properties but supports the page-break-* properties to accomplish the same result.
- WebKit-based based browsers support the non-standard -webkit-column-break-* properties to accomplish the same result (but only the auto and always values)
- Before Chrome 63, the browser supports the non-standard -webkit-column-break-* properties to accomplish the same result (but only the auto and always values)
- Samsum Internet version 4 does not support the column-fill property. This has been fixed for version 7

None of these issues are deal breakers, we just have to write more CSS to acomplish the same goal but it’s a good use case for feature queries or working with the cascade where browsers will ignore rules tht they don’t undertand.

> Visiting Vancouver, I was very impressed with the layout of @MONTECRISTO_Mag (https://t.co/iA23ilgJhw), but disappointed that their online presence didn't reflect that. So I made a prototype @CodePen to improve on that. https://t.co/nFNNmMcMIX pic.twitter.com/IUO9Yhpqbx
>
> — Dudley Storey (@dudleystorey) May 21, 2018

Looking at the source of the article I can see why the disappointment.

Let’s start with the basics.

We can specify columns either by indicating a number of columns, meaning that we will always have the same number of columns, regardless of how wide the display is:

```css
#column1 {
  border: 2px solid #000;
  column-count: 3;
}
```

We can also specify how wide we want the columns to be. The samller the width of the screen the fewer columns we'll get until it turns to a single column, responsive by default :)

```css
#column2 {
  border: 2px solid #000;
  column-width: 20em;
}
```

We can also specify the gap between the colummns using the `columns-gap` descriptor and can add a rule using `column-rule`.

In this example we make the columns 15em wide with a 2em gap between the columns and a solid green rules between them

```css
#column3 {
  column-width: 15em;
  column-gap: 2em;
  column-rule: 4px solid green;
  padding: 5px;
}
```

If we want columns of equal height we set up an explicit height and then use `column-fill` to make sure that they are all the same height (where possible).

```css
#column4 {
  column-width: 15em;
  column-gap: 2em;
  column-rule: 4px solid green;
  padding: 5px;
  column-fill: balance;
  height: 400px;
}
```

The final aspect of multicolumn layout that I want to cover is items spaning multiple columns. These may be subtitles or images that we want to cover the more than one column of text.

```
#column5 {
  column-count: 4;
  column-gap: 2em;
}

#span-content {
  column-span: all;
  border: 5px solid #0000ff;
}
```

“Sometimes when we are generous in small, barely detectable ways it can change someone else’s life forever.” — Margaret Cho

So what does it look like when put into a page? The full example shows the end result of the process.The relevant CSS code (shown as uncompiled SCSS) looks like this:

```
.container {
  margin: 0 auto;
  width: 80%;
}

.columns {
  columns: 2;
  column-gap: 2em;
}

.columns3 {
  columns: 3;
  column-gap: 2em;
}

figure {
  margin: 3em 0;

  img {
    width: 100%;
    height: auto;
  }
}
```

In this code the `container` class will center the element in the page.

The `columns` class will create 2 columns with a 2em gap between the columns.

`columns3` will create 3 smaller columns.

The `figure` element selector sets the top and bottom margin while the nested `img` selector makes the width 100% and the height automatic. This makes the image responsive to the width of the window.

Because we've used fixed column numbers we need to make sure that we adjust the columns and values using media queries.

If the screen is narrower than 640px then we want all text to be in a single column. Since the image is responsive we don't need to worry about changing it in the media queries.

```
@media screen and (max-width: 640px) {
  .columns,
  .columns3 {
    columns: 1;
  }
}
```

The next screen resolution is between 641 and 800 pixels wide. We set the text above the image to be a single column and the text below to two columns.

```css
@media screen and (min-width: 641px) and (max-width: 800px) {
  .columns {
    columns: 1;
  }

  .columns3 {
    columns: 2;
  }
}
```

This looks convoluted and confusing but we have to do it so it’s readable in tablets, phones and other form factors we may have never heard of.