Typography is not just about choosing some fancy fonts, good typography is all about having a system. Design is not decorating, it's about solving problems in a visually appealing way; and the best solutions are often founded in structure, order, rules and repetition.

Sass is an ideal partner for creating great typography because we can use it to create a repeatable system through variables, functions and mixins.

In this video we'll look at a couple of plugins that leverage custom functions and mixins to build a flexible typographic system. We'll cover

- Modular Scale
- and Sassline

Modular Scale

To bring some order and heirarchy to your heading sizes, you could consider using a Modular Scale. In this typographic system, you choose a base font size - such as `1em` and a ratio (or ratios). To calculate a series of font-sizes that work within this scale, you multiply each number by the same ratio.

Starting with the base of `1em` and a ratio of 1.5 the next size up would be `1 * 1.5 = 1.5em`. The next size up would be `1.5 * 1.5 = 2.25em`.

To create smaller numbers, you divide by the ratio. To create the first size down from the base you divide 1 by 1.5 to get 0.6667em. To get the next size down, divide this by 1.5 and so on.

This web-based tool at modularscale.com is a great way to visualise the scale and experiment with values which you can then copy and paste into your projects. There is also a Sass plugin you can install into your project for an even tighter integration but this requires some setup.

To use the Sass plugin, head to the Github page and choose your favourite method of installation.

I have a simple project set up here which is already using the Bower

package manager so I'll install with Bower. As long as you have Bower installed, you can run the following from the command line within your project:

```
bower install modular-scale --save-dev
```

This will install Modular Scale as a dependency. To use this in your Sass you will need to `@import` the modular scale library.

```
@import '../../bower_components/modularscale';
```

Now that the library has been imported, there is a small amount of setup required so we can start using the modular scale function.

I've got a `_type.scss` partial here which we'll use to set up all the typography settings.

The first thing we need is to create a `$modularscale` map variable which contains the base font size and the modular scale ratio.

```
$modularscale: (
    base: 1em,
    ratio: 1.5
);
```

For the ratio you can either manually add in a decimal value or use one of the pre-defined ratio variables. A list of these can be found on the modular scale github page.

```
$modularscale: (
    base: 1em,
    ratio: $golden
);
```

Having configured the modular scale variable we can now use the modular scale function to output values in our Sass.

I've created a series of selectors here for our heading styles and the default body styles.

```
body { }
small { }
h6 { }
h5 { }
h4 { }
h3 { }
h2 { }
h1 { }
```

We can set the font-size of each of these using the modular scale `ms()` function. This function takes a numeric parameter that determines how far up or down the modular scale we should go to calculate the font-size.

```
body { font-size: ms( 0 ) }
small { font-size: ms( -1 ) }
h6 { font-size: ms( 0 ) }
h5 { font-size: ms( 1 ) }
h4 { font-size: ms( 1 ) }
h3 { font-size: ms( 2 ) }
h2 { font-size: ms( 3 ) }
h1 { font-size: ms( 5 ) }
```

Having set these type sizes up we can now preview them in the browser to see what we have. You can now experiment with the base font-size or ratio to see what different combinations of settings yield, just by adjusting two values.

Sassline

One last Sass extension to mention is Sassline.
This is another tool that helps you handle setting type on a baseline grid. It also includes a responsive modular scale and contains a lot of typography best practices and turns on open type font features such as ligatures, old style numerals and kerning.

If you like the idea of using a baseline grid and having slick looking typography as well as a solid starting point for a new project, Sassline looks like a great option. It may not be the simplest thing to introduce to an existing project as it also comes with its own reset and grid but given the way the plugin has been structured, you could just `@import` the variables and mixins and go from there.

If you're starting a new project, this is definitely something worth looking into. Head to the Github page and download the project as a zip or clone it to your local machine.

In order to build and preview the demo we first need to install the project dependencies using `npm` and `bower`. You will need these tools installed first and details can be found on the Node.js website and Bower website.

To install dependencies run

```
npm install && bower install
```

After a while you should see a success message. The Sassline project uses the Gulp task runner to compile Sass and run a local development server. Ensure you have Gulp installed globally and then run `gulp serve` from the command line.

This will launch a local server which you can access via `http://localhost:1234`. If you browse to this URL, you should see the

Sassline demo page. This provides a great base for your new project and is a fully configured project using all the Sassline variables and mixins.

For details of how these all work, refer to this blog post by Jake, the plugin author.

Given the web is predominantly made up of text, it's hugely important to design legible text in a system that works - both for you and your users. The Modular Scale and Sassline projects can help you do that with ease!