Having embraced the power of Sass and started to use variables to configure various settings in your project - such as colours, font sizes and media query breakpoint - you might start to wonder how you can share these values between other front-end languages such as JavaScript.

In programming we want to minimise duplication and declaring variables in both your Sass files and your JavaScript files would be a maintenance nightmare.

There are a couple of ways to share data between Sass and JS using JSON. In this episode we'll cover the theory rather than the practice to keep things brief and to not end up going too deep down a JS rabit hole.

Sass Maps and JSON

In this series so far, we've not spent too much time looking at Sass maps as we'll have a whole video coming up very soon.

However, Sass maps and JSON serve a very similar purpose and have quite a similar syntax.

They are both stores of key and value pairs that can hold all sorts of different types of values, each with a unique key or label.

Here is a comparrison of the two, with Sass on the left and JSON on the right:

```
$colors: (
    pink: '#cc3f85',
    blue: '#94d6ed',
    purple: '#a485fb',
    yellow: '#e3db6f'
);
```

```
{
    ”colors”: {
        ”pink”: ”#cc3f85”,
        ”blue”: ”#94d6ed”,
        ”purple”: ”#a485fb”,
        ”yellow”: ”#e3db6f”
    }
```

```
}
```

As the syntax of both these data types is quite similar, it's possible to convert a Sass map into JSON or JSON into a Sass map.

Before looking into how this can be done, let's first think about why we might want to do it. Every project is different but there are some common use cases for sharing data:

- Sharing colours for switching between different visual themes
- Sharing breakpoint values for detecting screensizes in JS
- Sharing breakpoint values for working with responsive images
- Sharing version information or other meta data across languages

Let's start by looking at how Sass can be converted to JSON and made availabe to our JS files.

Sass -> JSON

Styling information should be kept in Sass/CSS as much as possible so starting with a Sass variable and converting it into JSON seems like a logical first approach.

The idea is to convert the Sass map into a JSON string and then apply it to the `content` property of a psuedo element so it can then be read by JS. A good choice is the `body:before` element as this is rarely used for any visual purpose.

After processing our Sass into JSON (more on that later) our compiled CSS might look like this:

```
body:before {
    display:none!important;
    content: '{ ”colors”: { ”pink”: ”#cc3f85”, ”blue”: ”#94d6ed”, ”purple”: ”#a485fb”, ”yellow”: ”
}
```

This value can then be read by JavaScript using the `getComputedStyle` method, santized and parsed into JSON.

```
var santizeContent = function( string ) {
    return string.replace(/^['”]+|\s+|\\|(;\s?})+|['”]$/g, '');
}
var getSassVars = function() {
    var style = null;

    if ( window.getComputedStyle && window.getComputedStyle(document.body, '::before') ) {
        style = window.getComputedStyle(document.body, '::before');
        style = style.content;
    }
```

```
    return JSON.parse( santizeContent( style ) );
}
var sassVars = getSassVars();
```

Having gone through this somewhat long-winded process, your Sass variables are now available in JavaScript and can be accessed through the `sassVars` JS variable:

```
var blue = sassVars.colours.blue;
console.log( blue ); // #94d6ed;
```

JSON -> Sass

An alternative and slightly simpler approach is to start with a JSON file that contains all the shared variable information.

Using a third-party tool like Sass JSON vars,
the JSON can be imported into your Sass files direct.

```
gem install sass-json-vars
```

```
@import 'shared-variables.json';
```

```
sass scss/main.scss:css/main.css -r sass-json-vars --watch
```

This technique leverages the computing power of a more powerful language to do the conversion rather than leaning on Sass to parse its native map variables into a JSON string.

And because JSON is easily consumed by JS, there's little work to be done on the JavaScript side other than to load the file - no extra parsing or sanitization required!

If I ever need to share data between the two languages, this is the approach I'd use for sure.

So, let's take a look at some other conversion tool that can help make this happen.

Tools

When researching this topic, I came across a number of blog posts and tools to help convert Sass to JSON. This is a relatively advanced topic but it seems the desire to share data between languages is strong.

To aid in converting Sass maps to JSON, check out

SassyJSON is
a pure Sass solution that converts Sass maps to JSON by Hugo Giraudel.

Sass to JS is
another Sass tool that converts Sass to JSON but also bundles JS helpers to aid in reading them from CSS into JS

To go the other way round and start with JSON and convert it to Sass there's Sass JSON
vars
which we've already discussed. This is a ruby gem that allows you to directly import a JSON file with the Sass `@import` command. There is an alternative version if you're using Node Sass with something like Gulp for compiling your styles.

I also found another Gulp plugin called
gulp-json-sass which
converts a JSON file into a Sass file with a series of global variables.

So whether or not you have the need to share variables between your Sass and your JavaScript, I hope this trip down the Sass rabit hole has demonstrated just how much power we have at our fingertips when using this fantastic tool.