When compiling Sass you might have noticed two files are generated. One is your compiled CSS file that ends in a `.css` file extension. The other is a file that ends in a `.css.map` file extension called a source map.

This map file provides a way to associate lines of code in the compiled CSS back to their original source in your Sass files.

This can be very useful when debugging your code in the browser and when correctly configured in your developer tools open up some really exciting features that allow you to edit and save your code direct from the browser.

In this video we'll cover:

- Enabling and working with source maps
- Creating workspaces to auto save your dev tools work to your code

Creating Source Maps

In Sass 3.4 and above, source maps are generated by default when you run the `sass` command.

I've got a simple Sass project set up here with an index page and an assets folder containing `scss` and `css` folders.

If I run the standard Ruby Sass command and compile by `style.scss` into `style.css` with the `--watch` flag and then save one of my Sass files, you'll see two files generated.

```
sass assets/scss/style.scss:assets/css/style.css --watch
```

Sass writes two files: `style.css` and `style.css.map`.

If we open the CSS file, you should recognise some familiar looking code. The Map file doesn't contain any styles but does contain a JSON object that contains a series of data points that maps line numbers in the css source Sass files to lines in the compiled CSS file.

So the browser knows where to find the map file, there is a line at the end of the CSS file that links the two together.

This is what source maps consist of, but what does that mean for us developers?

Well to understand that, let's jump into the browser.

Working With Source Maps

Here's the index page for our Simple Sass Project.

If I inspect element on the Sass logo in the title, it will show me the elements view in the Chrome dev tools.

In the styles tab, we see all the CSS declarations that are applying to this element along with the files where those declarations are made:

- _box.scss on line 15
- _reset.scss on line 3
- _box.scss on line 10 etc.

If I click on one of these file names, the dev tools will open the sources tab and jump to the line in the specified file. Here you'll see your authored Sass code rather than the compiled CSS - even though the only file linked to this HTML document is `style.css`.

Due to the presence of the source map, the `scss` folder is shown as a source in the file tree to the left. This can be an invaluable tool when working on - and debugging - a project.

Creating Workspaces

It's common to use the browser developer tools to track down bugs or to experiment and "design in the browser". You can inspect any element and play around with CSS values directly in the browser which is a great way to speed up your workflow.

The only downside is that these changes disappear when you refresh the page so to incorporate the various settings into your code you have to make a mental note of them and recreate them one by one. This is slow and prone to human error. But some developer tools allow you to save your work back to your original source files when working locally. In the Chrome browser this is done by creating a "Workspace".

The first step is to start a local webserver from within our project folder.

I'm on a Mac so will use the Python SimpleHTTPServer which should be available without the need to install anything. Do make sure your Sass compiler is still running and watching for changes too.

```
python -m SimpleHTTPServer 7000
```

Running this command will start a server running at `http://localhost:7000` which you can visit in the browser to check everything is working as before.

To create a link between the files being served on this local host and my file system, you first need to open the developer tools and click on the icon with three vertical dots to open the sub-menu.

From here click on “Settings” and then select “Workspace”.

A bit like the source maps forming a link between your Sass and your compiled CSS, a Workspace creates a mapping between files being served on localhost and your project files and folders.

Click on the “Add Folder” button and add `http://localhost:7000/assets` in the URL prefix field and `/` in the Folder path field. Then click the Add button to create your mapping. When done, you can close the settings window.

Back in the browser we can inspect elements as usual but instead of adjusting values in the styles tab, we can `cmd+click` on the property name or value which will take us to the sources tab and jump down to the corresponding line in the file.

From here, we can change colours, font-sizes, spacing and anything else right within the dev tools. Saving the file will auto refresh the page and your new Sass code will be saved. You can even edit variables and other Sass features without leaving your browser. Magic.