In this episode we'll be diving into two more command line tools that can help you compile Sass. They both do a lot more than just compile Sass so they're definitely worth looking in to. But for the purposes of keeping this video short and to the point, we'll just focus on the Sass side of things.

What are Grunt and Gulp

Grunt and Gulp are two popular tools for helping automate your development workflow. They're command line tools that are written in JavaScript and to use them you'll need to install NodeJS and the Node Package Manager.

If you want to follow along with this video, head over to the Node JS website for installation instructions.

Grunt and Gulp can both perform a number of tasks for jobs like minifying or compressing your code, concatenating multiple files together, checking code for errors, compressing images or compiling Sass. They have a slightly different approach to these tasks and you'd only use on or other of them on an individual project.

Tools like this may sound overly complex, a bit abstract or even a bit scary if you're new to web development. You're not alone. I felt the same way in the past but the only way to get better at your craft is to break out of your comfort zone.

For a great introduction to tools like Grunt, Chris Coiyer wrote an excellent article titled Grunt for people who think things like Grunt are weird and hard
a couple of years ago for 24 Ways.

Let's dive in to a practical step by step guide to using Grunt or Gulp. You can then decide which one you prefer or chose to use neither and keep with your existing process.

Compiling Sass with Grunt

I've got a simple project here with an `scss` folder containing a `main.scss` file and a `css` folder where the styles will be compiled.

With Node and `npm` installed, next we need to install Grunt. This is done from the command line with the following command:

```
npm install grunt-cli -g
```

The `-g` stands for "global" and this will install the Grunt command line interface so it can be used on any of your projects. This is something that only needs to be done once; if you already have Grunt installed, you can skip this step.

Once installed, we need to create a file called `package.json` which contains information about a series of tools (called node modules) that Grunt needs to perform all its tasks - like watching files for changes and compiling Sass.

On the command line, ensure you're in the project folder and run

```
npm init
```

Answer the questions if you're setting up a real project or just hit enter to use the defaults.

Now we can install a local version of Grunt and two additional packages: one for watching files for changes and one for compiling Sass. Run the following command to install these packages and save the details to the `package.json`

```
npm install --save-dev grunt grunt-contrib-watch grunt-contrib-sass
```

If you look in your package.json file, there should be a list of the 3 packages that were just installed:

```
"devDependencies": {
    "grunt": "^0.4.5",
    "grunt-contrib-sass": "^0.9.2",
    "grunt-contrib-watch": "^0.6.1"
}
```

With everything installed, we now need to create a configuration file to tell Grunt what we want it to do.

Create a new file named `gruntfile.js` in the root of the project and add the following code (just head to atozsass.com/g to copy and paste!):

```
module.exports = function( grunt ) {
    grunt.initConfig({
        sass: {
            dist: {
```

```
                files: {
                    'css/main.css' : 'scss/main.scss'
                }
            }
        },
        watch: {
            css: {
                files: '**/*.scss',
                tasks: ['sass']
            }
        }
    });
    grunt.loadNpmTasks( 'grunt-contrib-sass' );
    grunt.loadNpmTasks( 'grunt-contrib-watch' );
    grunt.registerTask( 'default', ['watch'] );
}
```

This sets up two Grunt tasks, one called `watch` that watches our Sass files for changes and one called `sass` that compiles `main.scss` into `main.css`. If you use this setup in future projects, you may have to make adjustments to the file paths but this is a great starting point.

Grunt can now be run with the `grunt` command, optionally specifying the name of the task afterwards.

Running `grunt sass` will run the `sass` command and your styles will be compiled once.

Running `grunt watch` will run the `watch` task and Grunt will run and keep watching for changes until you quit by using the keyboard shortcut `Ctrl+C`.

Running `grunt` without a task name will run the `default` task which in this case just runs `watch` but could be configured to do all sorts of other things too.

To double check everything works as expected, write some Sass code in your `main.scss` file, hit save, check your terminal to see that Grunt detected a change and then look inside the `css` folder. You should see your compiled CSS in a file called `main.css`. Nice work!

Compiling Sass with Gulp

Gulp is a very similar tool to Grunt but uses a different style of configuration and a different approach to running the tasks behind the scenes.

The Gulp config syntax may look more familiar if you're used to writing JS or jQuery code and while I've used both Grunt and Gulp, I now

favour Gulp for all my projects.

For this exercise, I've got another bare-bones project with an `scss` folder with a `main.scss` file for my Sass and a `css` folder for the styles to be compiled into.

The initial Gulp setup follows a similar process to Grunt.

First we need to install the Gulp tool globally so it can be used on any project - this is a one-time installation so if you already have Gulp installed, you can skip this step.

To install Gulp, run the following command.

```
npm install -g gulp
```

We now need to create a `package.json` which will list all the packages needed for the Gulp tasks.

```
npm init
```

Now we need to install a local version of Gulp and the Gulp Sass package:

```
npm install --save-dev gulp gulp-sass
```

With the packages installed, we create a configuration file to set up our tasks. Create a `gulpfile.js` in the project root with the following code (just head to atozsass.com/g to copy and paste!):

```
var gulp = require( 'gulp' );
var sass = require( 'gulp-sass' );

gulp.task( 'sass', function() {
    gulp.src( 'scss/main.scss' )
        .pipe( sass().on( 'error', sass.logError ) )
        .pipe( gulp.dest( './css/' ) );
});

gulp.task( 'default', function() {
    gulp.watch( 'scss/**/*.scss', ['sass'] );
});
```

Now to run the tasks, we use the `gulp` command followed by the task name. To compile your styles once, run `gulp sass`. To watch for changes we can run the default task by just running `gulp` in the terminal from the project root.

If all is working properly, you should see a `main.css` file compiled when you add some Sass to `main.scss` and hit save.

So what's the difference between Grunt and Gulp and why are there two very similar ways of doing the same thing?

The Gulp syntax is shorter and I personally find chained functions easier to read than a big configuration object used with Grunt.

But the biggest difference lies in the compiler that's used behind the scenes. Grunt uses Ruby Sass whereas Gulp uses Node Sass. Node Sass is a wrapper for Libsass which is a much faster than the traditional Ruby Sass.

On a big project compiling with Gulp will be significantly faster than compiling with Grunt and that alone is the reason why I've switched from Grunt to Gulp for all my projects.

Next Steps with Automation

Having gone through the process of setting up these tools, we can now add additional tasks to help speed up our workflow.

Two tools that I couldn't live without are Live Reload and Autoprefixer.

Live Reload will automatically refresh the browser when changes to any files is detected. Even though this isn't a particularly difficult thing to do manually, it's something that I do hundreds of times per day so the time saving is hundreds of times multiplied by however long it takes me to `cmd-tab` between my editor and browser and hit `cmd-r` to refresh. It's a micro optimisation but totally worth it.

Autoprefixer is a postCSS
plugin that runs on your compiled CSS and adds in any vendor prefixes where needed as per the data on caniuse.com.
This means I never have to write prefixes in my Sass and means I can just focus on making things, rather than faffing around looking up or writing prefixes manually.

Whether you prefer Grunt or Gulp, I'd highly recommend automating your development process where possible as it's one less thing you'll have to think about.