4 Variables

Table of Contents

[To turn back on your Watch SASS 1](#_Toc161739560)

[What are Variables 2](#_Toc161739561)

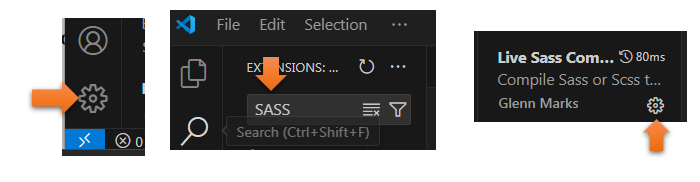
[Using SASS variables 4](#_Toc161739562)

[What are Maps? 7](#_Toc161739563)

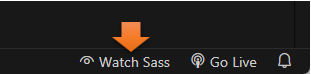
[To Add Lorem auto text filler 8](#_Toc161739564)

# To turn back on your Watch SASS

You might have to go back to settings, put SASS in the search bar, and go to the wheel here



At the bottom, choose WATCH SASS

****

# What are Variables

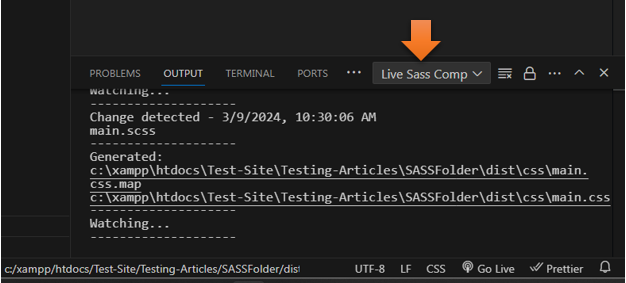
Yes, you might say that variables are actually used inside of regular CSS, but SASS has had variable usage a lot longer than CSS has been using them.

When you use SCSS variable, they do not compile into the language of CSS, they compile straight into the actual value that is stored in the variable.

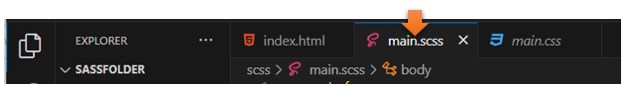
This is how you would use a regular CSS variable.

Watch it, we do everything inside of our SCSS style sheet, and nothing is done inside of the .css file. Also when you first start writing this code, it wants everything all done at once, and will show you squiggly lines until you fill some rules into the :root, so don’t freak out.

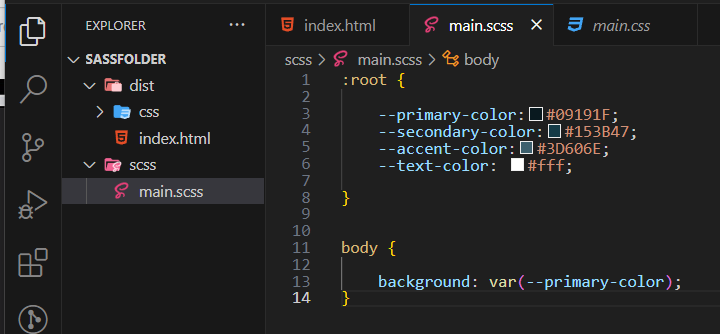
Watch it, Make sure you have Live SASS Compile in the terminal.



Watch it, Make sure you are in your **main.scss,** and not your main.css file

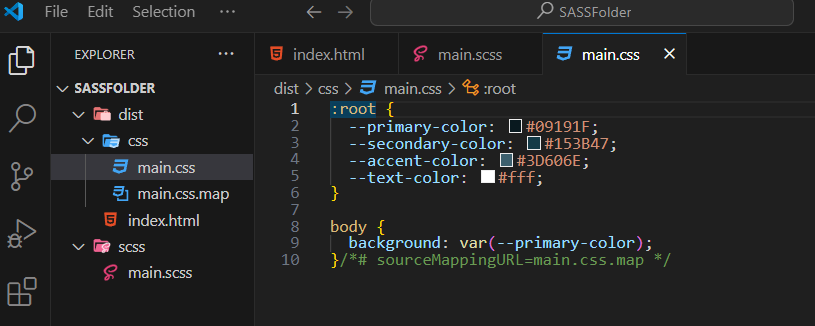


This is how you would write the code in CSS

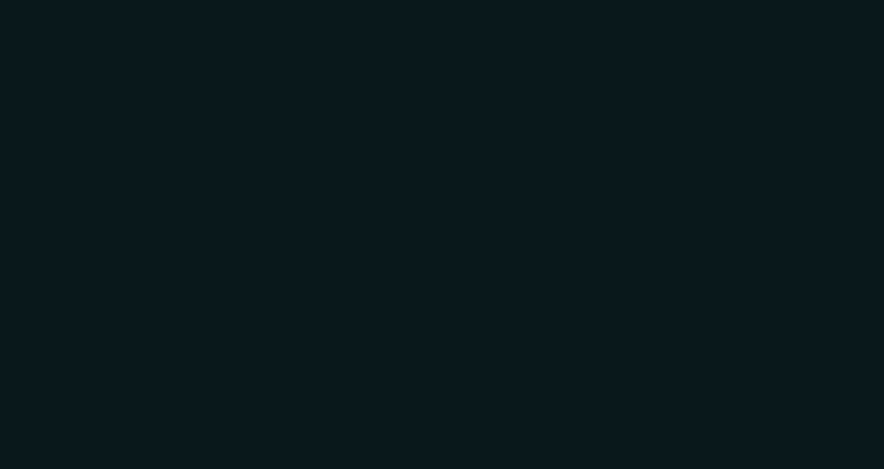


Watch it, YOU MUST SAVE IT!

You see, since this is how CSS is done, you will notice that even though we wrote it in the SCSS file, that the CSS will be identical if you look at it



Watch it, if you want to test this, you will need to go back to the **index.html**, if you just see a file structure, you are trying to look at your CSS file, and it will not be what you want.

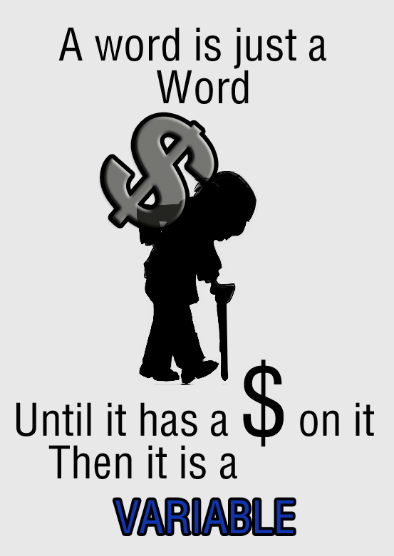


# Using SASS variables

When you use SCSS variables, you can get rid of the word - **root**, the **double hyphens**, and the **curly braces.**

You will add a **dollar sign** to show that this is a variable.

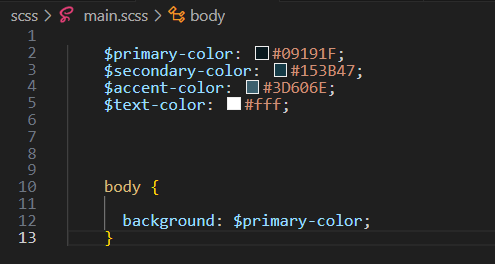
IN SASS a Word is just a Word



See, this way of writing it is a lot easier, and cleaner. You just have to get used to seeing that dollar sign in front of the name of your variables.

**Key: Value Pairs**

Watch it, these are key: value pairs. Use the colon between the key and the value, and then end the line with a semi-colon. Oh, and don’t forget that funny looking dollar sign to signify the variable name.



The Code

  $primary-color: #09191F;

    $secondary-color: #153B47;

    $accent-color: #3D606E;

    $text-color: #fff;

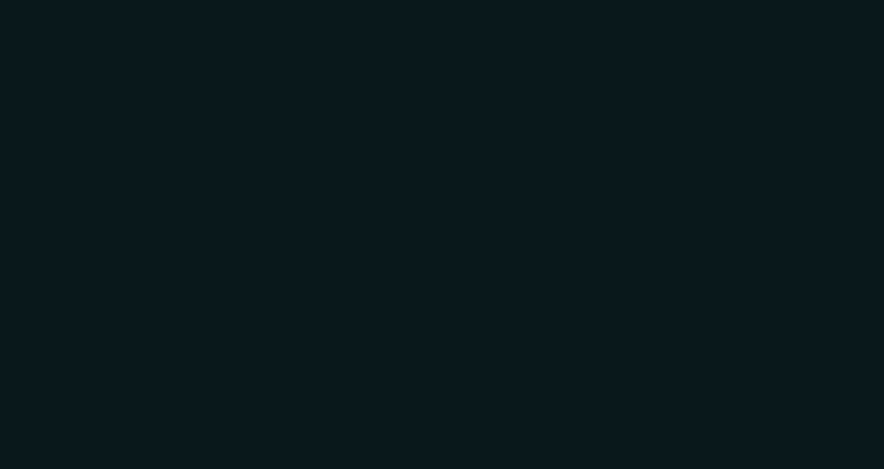
Then for the body, you can just put in the **variable name** with its little **dollar sign**. Remember the colon to separate, and the semi-colon at the end of the line.

body {

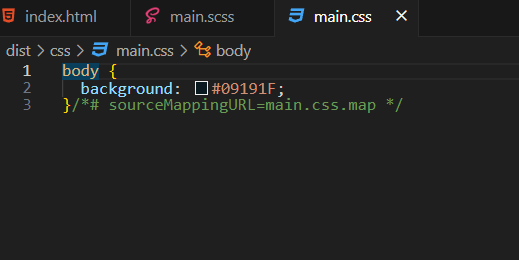
      background: $primary-color;

    }

Go to your **index.html** to check it, and it is still your background color



…but look at how your CSS interpreted it. This is much simpler code for the CSS file. As I said earlier, it puts the actual value of the variable in there.



# What are Maps?

Maps are like your SCSS variables. What I mean by this is that **maps** are **key: value** pairs too.



And this is what a map will look like when we add it to our SCSS file

Notice the dollar sign and colon in the selector, and then the colon in the rule, every rule property (except the last one) is separated with a comma. Maps are also sort of like containers for multiple variables. Wrap the variables in parenthesis to hold them together.

$font-weights:(

        "regular": 400,

        "medium": 500,

        "bold": 700

        );

    $font-colors: (

        "light": #f2e8d4,

        "middle": #9b8559,

        "dark": #1e1401

    );

**To See this in action**, we need to add some HTML Markup to our **index.html** file

# To Add Lorem auto text filler

in your paragraph tab, write out the word **Lorem 50**, hit **enter** and this will give you 50 words in Lorem ipsum. You want to add this div, with a paragraph, and some text, so we can test our new styles.

**INDEX.html**

<body>

<div class="main">

<p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Temporibus rem voluptatum accusantium itaque atque eum rerum velit veniam, ratione architecto, officiis ab impedit ut assumenda quisquam. Ipsum autem laboriosam illum nobis facere, unde ea, aut nesciunt eius, a voluptas non iste consequatur soluta molestiae doloribus quae esse iusto? Vel, id!</p>

</div>

</body>

Go back to the **SCSS** file

Now remember that the official CSS property for our font color is just plain **color**, so that is what we need to use.

With the map set up above now you can use it in the body rule in the scss file. the map.get pulls the stuff you created in the map, and throws it into the official CSS property.

 body {

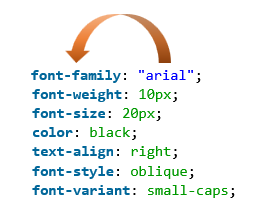
      background: $primary-color;

      font-weight: map-get($font-weights, bold);

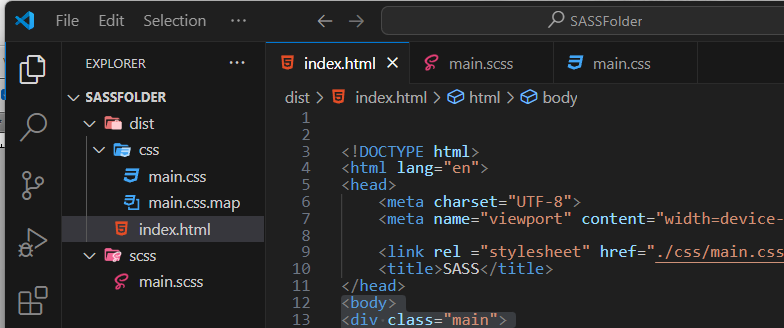
      color: map-get($font-colors, middle);

    }

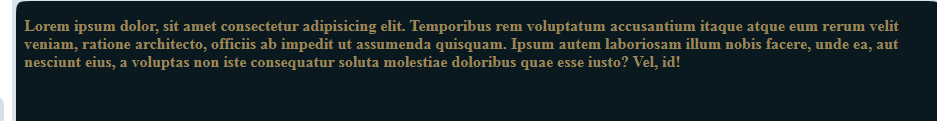
You can get the official CSS properties for your fonts [here](https://www.geeksforgeeks.org/css-cheat-sheet-a-basic-guide-to-css/#Font%20Properties), remember the syntax is that we are throwing everything on the right side, into the official CSS property on the left.



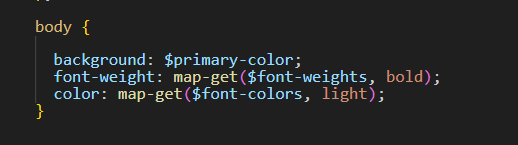
Go back to your **Index.html** to view the file



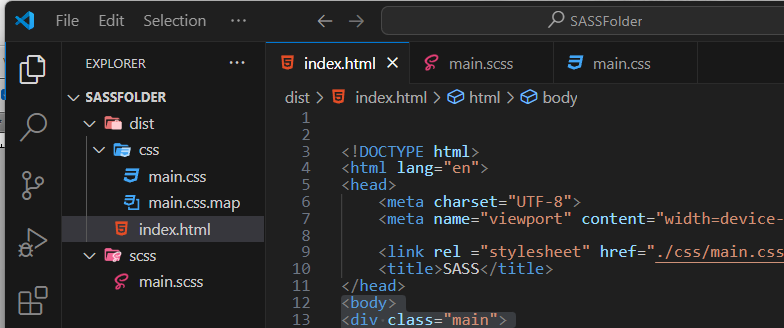
Now use the live server at the bottom to view it



Change the SCSS to light to test it

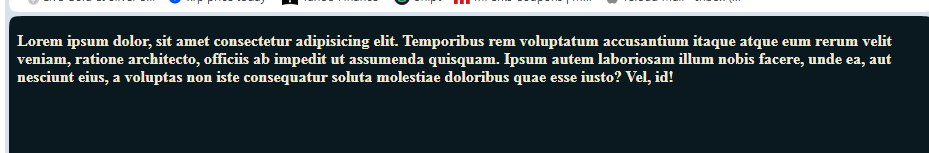


Back to the **Index.html**



Hit the **live Server** to view it in the site

and it is listening…

****

If you look below, in the whole site if you need to change the size of your medium font, for example, you can do that here in one place.

$font-weights:(

        "regular": 400,

        "medium": 500,

        "bold": 700

        );

    $font-colors: (

        "light": #f2e8d4,

        "middle": #9b8559,

        "dark": #1e1401

    );

In the CSS, the entire thing that we wrote in the SCSS in minified for our CSS file, and the regular CSS file is what the browser is reading.

