

SCSS Variables (i)

We've walked through how to work with variables in Scss to make it possible to make a single change that populates a number of style definitions. To take this knowledge a step further, in this guide we'll examine how to set default variable values in Scss.

Why Use Variables in SCSS?

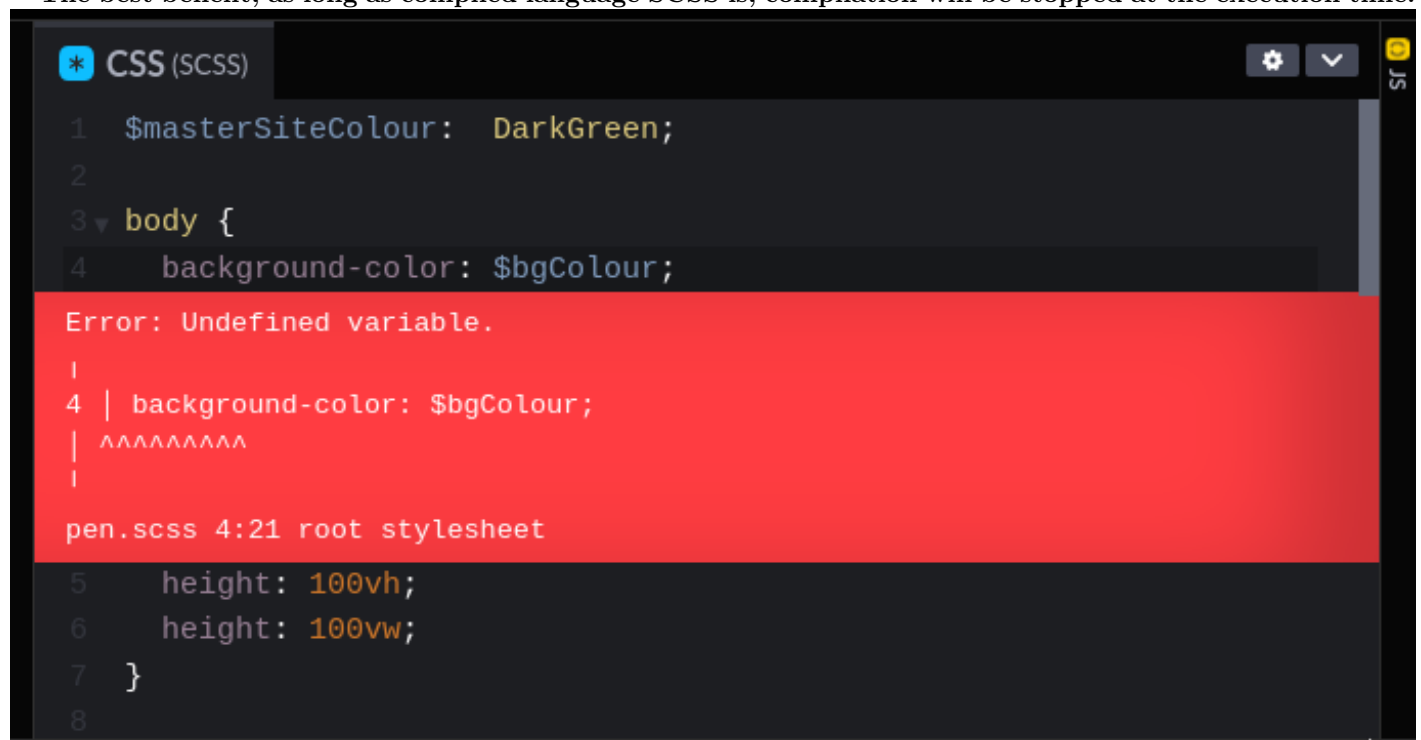
One of the most powerful aspects of using SCSS is the ability to make your code more **efficient**.

If you're familiar with programming languages like **Ruby** or **Python**, the concept of variables is already second nature. A **variable** is simply a **container** that stores a value and can be called anywhere in your application.

Benefits of Variables:

SCSS variables empower you to manage styles more effectively by: - **Centralizing changes**
- **Removing redundancy** - **Easier Maintenance** - **Cleaner Code** - **Scalability**

The best benefit, as long as compiled language SCSS is, compilation will be stopped at the execution time:



```
* CSS (SCSS)
1 $masterSiteColour: DarkGreen;
2
3 body {
4   background-color: $bgColour;
5   height: 100vh;
6   height: 100vw;
7 }
8
```

Error: Undefined variable.

4 | background-color: \$bgColour;
| ^^^^^^^^^^^

pen.scss 4:21 root stylesheet

This allows better debugging process.

The Problem: Duplication in CSS

Imagine a situation where you have a CSS file spanning **thousands of lines**. You might encounter repeated instances of the same styles—like a background color or font color.

For example, consider a scenario where a client requests you to: - Change all instances of **dark red** to **dark green**.

Without variables, you would need to manually search and replace every instance. This process is: - **Time-consuming**
- Prone to **errors**

SCSS Variables: A DRY Solution

In SCSS, you can follow the “**Do Not Repeat Yourself**” (DRY) principle by leveraging **variables**.

How Variables Help:

1. Define variables at the **top of your file**.
 2. Use those variables wherever you need the values.
 3. When updates are needed, change the variable's value, and it updates across the entire file.
-

Step-by-Step Process

1. **Set Up SCSS:**
Ensure your SCSS is properly configured to compile and process styles.
2. **Identify Duplication:**
Look for repeated values like colors or fonts in your CSS.
3. **Define Variables:**
Place your variables at the **top** of your SCSS file for easy management.
4. **Replace Values with Variables:**
Update your code by using variables instead of hardcoded values. —

Examples

CSS Before declaring variables

```
body {  
  background-image: radial-gradient(circle at top left, #353840, #141414);  
  display: flex;  
  flex-direction: column;  
  justify-content: flex-start;  
  align-items: stretch;  
  margin: -0.5em 0 -0.5em 0;  
  min-height: 100vh;  
  /* min-width: 100vw; */  
  box-sizing: border-box;  
}  
  
.mainBody {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: center;  
  margin: -0.5em 0 -0.5em 0;  
  padding: 20px;  
  flex-grow: 1;  
  max-width: 100%;  
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);  
}  
  
.bodyPart1, .bodyPart2 {  
  flex: 1;  
  margin: -0.5em 0 -0.5em 0;  
}  
  
.bodyPart1 p {  
  font-family: 'Lato';  
  text-align: left;  
  color: #cacaca;  
  font-size: 1.9rem;
```

```

    margin-bottom: 30px;
}

.bodyPart1 > p > strong {
    color: #cacaca;
}

.bodyPart1 > p::before {
    content: "Text ";
}

.bodyPart1B p {
    color: #cacaca;
    font-size: 1.8rem;
}

.bodyButtons {
    display: flex;
    gap: 30px;
}

.signUpBodyButton {
    font: bold 24px 'Lato';
    color: #cacaca;
    background-color: #4bce79;
    padding: 0.5em 1em;
    border-radius: 0.15em;
    cursor: pointer;
    transition: background-color 0.3s, transform 0.2s;
}

```

SCSS after variables declared

```

$textColorAsVariable:    #cacaca;
$marginAsVariable:      -0.5em 0 -0.5em 0;

body {
    background-image: radial-gradient(circle at top left, #353840, #141414);
    display: flex;
    flex-direction: column;
    justify-content: flex-start;
    align-items: stretch;
    margin: $marginAsVariable;
    min-height: 100vh;
    /* min-width: 100vw; */
    box-sizing: border-box;
}

.mainBody {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin: $marginAsVariable;
    padding: 20px;
    flex-grow: 1;
    max-width: 100%;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);
}

.bodyPart1, .bodyPart2 {

```

```

    flex: 1;
    margin: $marginsAsVariable;
}

.bodyPart1 p {
    font-family: 'Lato';
    text-align: left;
    color: $textColorAsVariable;
    font-size: 1.9rem;
    margin-bottom: 30px;
}

.bodyPart1 > p > strong {
    color: $textColorAsVariable;
}

.bodyPart1 > p::before {
    content: "Text ";
}

.bodyPart1B p {
    color: $textColorAsVariable;
    font-size: 1.8rem;
}

.bodyButtons {
    display: flex;
    gap: 30px;
}

.signUpBodyButton {
    font: bold 24px 'Lato';
    color: $textColorAsVariable;
    background-color: #4bce79;
    padding: 0.5em 1em;
    border-radius: 0.15em;
    cursor: pointer;
    transition: background-color 0.3s, transform 0.2s;
}

```

Video Lesson Speech

[ENG] # Introduction to Variables in Scss This guide examines the syntax for declaring and calling variables in Scss files.

One of the most powerful parts of using SCSS in development is the ability to make your code more efficient. If you're coming from a programming language such as Ruby or Python then the concept of having variables is a pretty common thing that you would implement into a program.

A variable is simply a container that can store a value and then can be called from anywhere else in the application. So you don't have to put identical code all through the entire app. So let's take a look at this basic example.

So you should already have your CSS lined up and have the SCSS property set so that it compiles and it can actually process the SCSS as opposed to simply treating it like regular CSS. What we are going to be talking about here are variables.

What we want to talk about is how we can leverage variables. If we look we can see that we have some duplication here. We have the background color and whenever you see six F's in a row that's going to be the color white. We also have this dark red color. All of this is fine but imagine that you have a CSS file that is a few thousand lines long. An example of that is if you pulled in some kind of template or you're building a template and the client informs you that he would like to change all the instances of dark red to dark green. This means that you would have to search and make sure you change

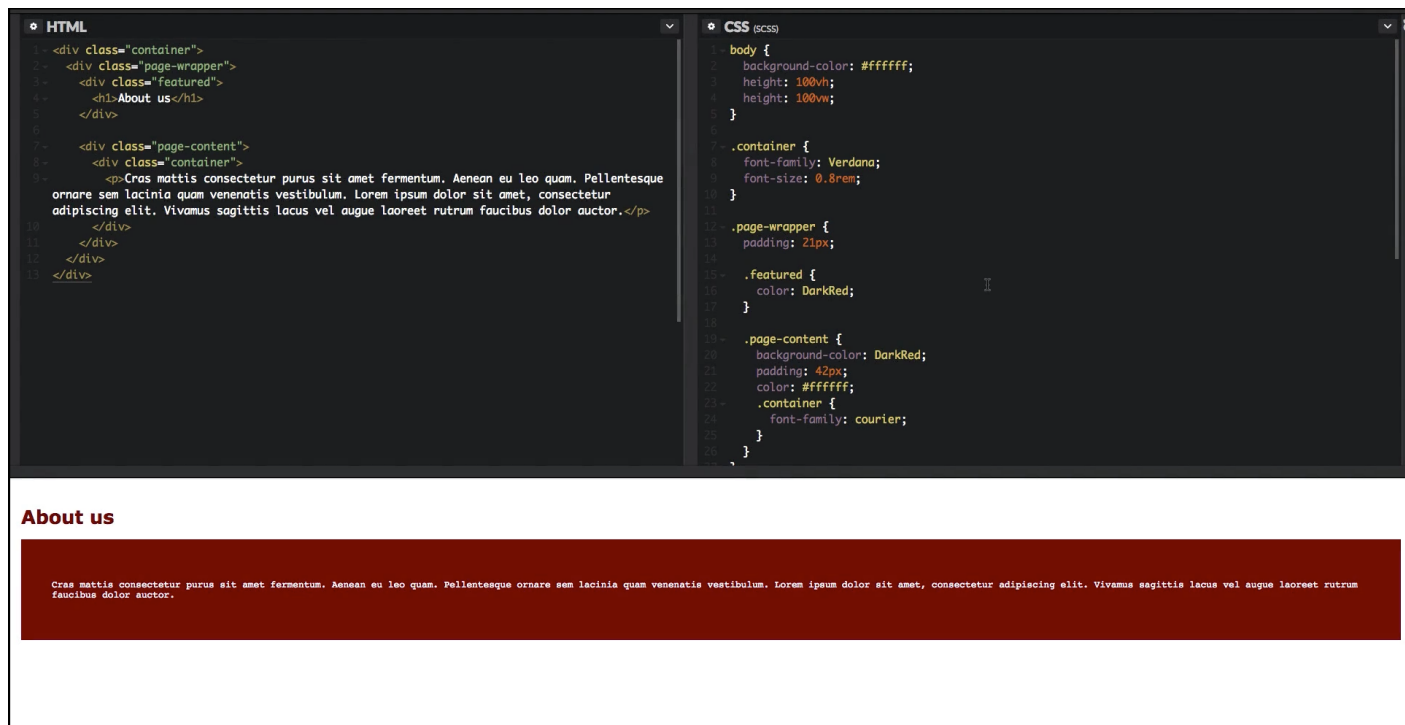


Figure 1: IMG1

every instance. Part of the logic behind using SCSS is the fact that you can implement the same type of mindset that Rails has with “Do Not Repeat Yourself”.

That’s where variables come in. We would want to set our variables at the very top. Here we can see how we set variables in SCSS.

Now, all we have to do is use that variable to change your styles across the CSS file. Imagine our client from earlier, now if he came to us with a change to make to our color scheme. We only have to change the value of that variable and it will reflect everywhere we used that variable.

We will go ahead and change the places that we had duplicate calls for #ffffff without a new variable.

Now everything is working and we’ve cleaned up and remove the duplication by leveraging variables in SCSS.

Here is a demo where I use variables to manage the background color and font color for a page.

Starter Code

```
<div class="container">
  <div class="page-wrapper">
    <div class="featured">
      <h1>About us</h1>
    </div>

    <div class="page-content">
      <div class="container">
        <p>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
        </p>
      </div>
    </div>
  </div>
</div>

body {
```

```

    background-color: #ffffff;
    height: 100vh;
    height: 100vw;
}

.container {
    font-family: Verdana;
    font-size: 0.8rem;
}

.page-wrapper {
    padding: 21px;

    .featured {
        color: DarkRed;
    }

    .page-content {
        background-color: DarkRed;
        padding: 42px;
        color: #ffffff;
        .container {
            font-family: courier;
        }
    }
}

```

Finished Code

```

$white: #ffffff;
$master-site-color: DarkGreen;

body {
    background-color: $white;
    height: 100vh;
    height: 100vw;
}

.container {
    font-family: Verdana;
    font-size: 0.8rem;
}

.page-wrapper {
    padding: 21px;

    .featured {
        color: $master-site-color;
    }

    .page-content {
        background-color: $master-site-color;
        padding: 42px;
        color: $white;
        .container {
            font-family: courier;
        }
    }
}

```

}

[SPA]

Introducción a las Variables en SCSS

Esta guía examina la sintaxis para declarar y llamar variables en archivos SCSS.

Una de las partes más poderosas de usar SCSS en el desarrollo es la capacidad de hacer que tu código sea más

Una variable es simplemente un contenedor que puede almacenar un valor y luego puede ser llamado desde cualquier

Debes tener tu CSS alineado y la propiedad SCSS configurada para que compile y pueda procesar el SCSS en lugar

Lo que queremos discutir es cómo podemos aprovechar las variables. Si observamos, podemos ver que tenemos cie

Aquí es donde entran en juego las variables. Querríamos establecer nuestras variables en la parte superior. Aqu

Ahora, todo lo que tenemos que hacer es usar esa variable para cambiar tus estilos en todo el archivo CSS. Im

Procederemos a cambiar los lugares donde teníamos llamadas duplicadas para #ffffff sin una nueva variable.

Ahora todo está funcionando y hemos limpiado y eliminado la duplicación al aprovechar las variables en SCSS.

Aquí hay una demostración donde uso variables para administrar el color de fondo y el color de fuente de una