

Working with SASS

DIY - 11

- **Task 1: *Introduction to Sass:***
Mention at least two advantages of using Sass.

1. Variables:

Sass allows you to define variables, which can store values like colors, fonts, or any CSS value you need to reuse. This makes it easier to maintain and update your styles.

Advantages:

- **Consistency:** Ensures consistent styling across the entire project.
- **Maintainability:** Easy to update values in one place instead of searching through multiple CSS files.

2. Nesting:

Sass allows you to nest your CSS selectors in a way that follows the same visual hierarchy of your HTML. This makes the CSS more readable and maintainable.

Advantages:

- **Readability:** Makes the structure of the CSS easier to read and understand.
- **Modularity:** Encourages writing modular CSS, which is easier to manage and scale.

These features, among others, make Sass a powerful tool for front-end developers, enhancing both the productivity and maintainability of their stylesheets.

- **Task 2: *Installing Sass:***

Provide step-by-step instructions for installing Sass on your computer.

You can choose your preferred method (e.g., CLI, GUI tools).

To install Sass on a Windows computer, you can use various methods. The most common way is through Node.js and npm (Node Package Manager).

Here are the step-by-step instructions:

1. Install Node.js and npm

Step 1: Download Node.js

- Go to the [official Node.js website](https://nodejs.org/).
- Download the LTS (Long Term Support) version, as it is more stable.

Step 2: Install Node.js

- Run the downloaded installer.
- Follow the installation instructions. Ensure that the option to install npm (Node Package Manager) is selected.
- After the installation is complete, you can verify the installation by opening the Command Prompt (press Win + R, type cmd, and press Enter) and running:

```
cmd

node -v
npm -v
```

This should display the versions of Node.js and npm.

. Install Sass Using npm

Step 1: Open Command Prompt

- Press Win + R, type cmd, and press Enter.

Step 2: Install Sass

- In the Command Prompt, run the following command to install Sass globally

```
cmd

npm install -g sass
```

The -g flag installs Sass globally on your system, making it available from any directory.

Step 3: Verify Sass Installation

- After the installation is complete, verify that Sass is installed correctly by running

```
cmd

sass --version
```

This should display the version of Sass that was installed.

3. Using Sass

Step 1: Create a Project Directory

- Create a new directory for your project. You can do this through the Command Prompt or File Explorer.

cmd

```
mkdir my-sass-project  
cd my-sass-project
```

Step 2: Create Sass Files

- Create a .scss file within your project directory. For example, create a styles.scss file with some basic Sass code:

SCSS

```
// styles.scss  
$primary-color: #3498db;  
  
body {  
  font-family: Arial, sans-serif;  
  background-color: $primary-color;  
}
```

Step 3: Compile Sass to CSS

- In the Command Prompt, navigate to your project directory and run the following command to compile your .scss file to .css:

cmd

```
sass styles.scss styles.css
```

This will create a styles.css file with the compiled CSS.

Step 4: Watch for Changes (Optional)

- To automatically recompile your Sass files whenever you make changes, you can use the `--watch` flag:

```
cmd  
  
sass --watch styles.scss:styles.css
```

This will keep the Sass compiler running, and it will update the `styles.css` file whenever you save changes to `styles.scss`.

- **Task 3: Using Variables in Sass:**

Create a Sass file (`style.scss`) that defines variables for common colors used in web design. Use these variables to style a simple HTML page containing a header, a paragraph, and a button.

Step-by-Step Instructions

1. Create the HTML File

First, create an HTML file named `index.html` with the following content:

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Sass Variables Example</title>  
    <link rel="stylesheet" href="styles.css">  
  </head>  
  <body>  
    <header>  
      Welcome to My Website  
    </header>  
    <p>  
      This is a simple paragraph styled with Sass variables.  
    </p>  
    <button>  
      Click Me  
    </button>  
  </body>  
</html>
```

2. Create the Sass File

Next, create a Sass file named `styles.scss` with the following content:

Create a Sass file (`style.scss`) that defines variables for common colors used in web design. Use these variables to style a simple HTML page containing a header, a paragraph, and a button.

```
// Define color variables
$primary-color: #3498db;
$secondary-color: #2ecc71;
$text-color: #333;
$background-color: #f5f5f5;
$button-color: #e74c3c;
$button-text-color: #fff;

body {
  font-family: Arial, sans-serif;
  background-color: $background-color;
  color: $text-color;
  padding: 20px;
}

header {
  background-color: $primary-color;
  color: $background-color;
  padding: 10px;
  text-align: center;
  font-size: 2em;
  margin-bottom: 20px;
}

p {
  background-color: $secondary-color;
  color: $background-color;
  padding: 15px;
  border-radius: 5px;
}

button {
  background-color: $button-color;
  color: $button-text-color;
  border: none;
  padding: 10px 20px;
  font-size: 1em;
  cursor: pointer;
  border-radius: 5px;
}

button:hover {
  background-color: darken($button-color, 10%);
}
```

- **Task 4: Exploring Operators in Sass:**

Explain how these operators can be used to perform calculations within Sass.

Arithmetic Operators

1. Addition (+)

- You can add numbers, colors, or lengths together.

```
$base-padding: 10px;
.container {
  padding: $base-padding + 5px; // Results in 15px padding
}


$base-color: #333;
.text {
  color: $base-color + #111; // Results in #444
}
```

2. Subtraction (-)

Subtract numbers, colors, or lengths from one another

```
$base-margin: 20px;
.box {
  margin: $base-margin - 5px; // Results in 15px margin
}

$base-color: #333;
```



3. Multiplication (*)

- Multiply numbers or lengths

```
$base-font-size: 16px;
.header {
  font-size: $base-font-size * 2; // Results in 32px font size
}

.container {
  width: 50px * 2; // Results in 100px width
}
```

Division (/)

- Divide numbers or lengths. Note that Sass treats division differently inside functions and mixins vs. plain CSS.

```
$total-width: 100%;  
.column {  
  width: $total-width / 3; // Results in 33.333333% width  
}  
  
$base-font-size: 16px;  
.small-text {  
  font-size: $base-font-size / 2; // Results in 8px font size  
}
```

Color Operations

1. Darken and Lighten

- You can manipulate colors by making them darker or lighter.

```
$base-color: #3498db;  
.dark-bg {  
  background-color: darken($base-color, 10%); // Darkens the color by 10%  
}  
  
.light-bg {  
  background-color: lighten($base-color, 10%); // Lightens the color by 10%  
}
```

Adjusting Hue, Saturation, and Lightness

- Adjust color properties individually.

```
$base-color: #3498db;  
.adjusted-color {  
  color: adjust-hue($base-color, 45deg); // Adjusts the hue by 45 degrees  
}  
  
.desaturated {  
  color: desaturate($base-color, 20%); // Reduces saturation by 20%  
}  
  
.saturated {  
  color: saturate($base-color, 20%); // Increases saturation by 20%  
}
```


Percentage and Length Calculations

1. Percentages

- Convert lengths to percentages and perform operations on percentages.

```
$total-width: 100%;  
.half-width {  
  width: $total-width * 0.5; // Results in 50% width  
}  
  
.quarter-width {  
  width: 25% + 10%; // Results in 35% width  
}
```

Unitless Numbers

- Perform calculations on unitless numbers, which can be useful for generating responsive layouts.

```
$columns: 12;  
$gutter: 2%;  
  
.grid-item {  
  width: (100% / $columns) - $gutter; // Calculate width for grid items  
}
```

Combining Operators

1. Combining Operators for Complex Calculations

- You can combine different operators to create complex calculations.

```
$base-padding: 10px;  
$multiplier: 1.5;  
  
.box {  
  padding: ($base-padding * $multiplier) + 5px; // Results in 20px padding  
}  
  
$base-color: #3498db;  
.custom-color {  
  color: lighten($base-color, 10%) + #222; // Lighten the base color and then add aot  
}
```

Example

Here's an example combining several operators in a real-world context:

```
$base-font-size: 16px;
$padding: 10px;
$margin: 20px;
$base-color: #3498db;
$columns: 12;

.container {
  font-size: $base-font-size * 1.2; // 19.2px
  padding: $padding + 5px; // 15px
  margin: $margin - 5px; // 15px
  color: darken($base-color, 10%); // Darkened color
}

.grid-item {
  width: (100% / $columns) - 2%; // Width calculation for grid items
}
```

- **Task 5: Nesting in Sass:**

Create a new Sass file (styles.scss) that mimics the structure of any website.

```
$primary-color: #3498db;
$secondary-color: #2ecc71;
$text-color: #333;
$background-color: #f5f5f5;
$header-bg: #34495e;
$footer-bg: #2c3e50;
$font-stack: Arial, sans-serif;
$padding: 20px;
$margin: 20px;

// Mixins
@mixin container {
  max-width: 1200px;
  margin: 0 auto;
  padding: $padding;
}

@mixin flex-center {
  display: flex;
  justify-content: center;
  align-items: center;
}

// Global Styles
body {
  font-family: $font-stack;
  color: $text-color;
  background-color: $background-color;
  margin: 0;
  padding: 0;
}
```

```
a {
  color: $primary-color;
  text-decoration: none;
  &:hover {
    text-decoration: underline;
  }
}

// Header Styles
header {
  background-color: $header-bg;
  color: #fff;
  padding: $padding;
  text-align: center;
  @include container;
  h1 {
    margin: 0;
  }
}

// Navigation Styles
nav {
  background-color: $secondary-color;
  @include container;
  ul {
    list-style: none;
    display: flex;
    justify-content: center;
    padding: 0;
    li {
      margin: 0 $padding / 2;
      a {
        display: block;
        padding: $padding / 2;
        color: #fff;
      }
    }
  }
}
```

```
    &:hover {
      background-color: darken($secondary-color, 10%);
    }
  }
}
}
}

// Main Content Area
.main {
  @include container;
  display: flex;
  flex-wrap: wrap;
  gap: $padding;

  .content {
    flex: 2;
    background-color: #fff;
    padding: $padding;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    border-radius: 5px;

    h2 {
      margin-top: 0;
    }

    p {
      line-height: 1.6;
    }
  }
}
```

```
.sidebar {
  flex: 1;
  background-color: #ecf0f1;
  padding: $padding;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  border-radius: 5px;

  h3 {
    margin-top: 0;
  }
}

// Footer Styles
footer {
  background-color: $footer-bg;
  color: #fff;
  text-align: center;
  padding: $padding;
  position: relative;
  bottom: 0;
  width: 100%;
}
```

Example HTML to Link the CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Website Layout</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <h1>My Website</h1>
  </header>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <li><a href="#">Services</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>
  <div class="main">
    <div class="content">
      <h2>Welcome to My Website</h2>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla vel eros eu
    </div>
    <div class="sidebar">
      <h3>Sidebar</h3>
      <p>Links, ads, or any additional content can go here.</p>
    </div>
  </div>
  <footer>
    <p>&copy; 2024 My Website</p>
  </footer>
</body>
</html>
```

- **Task 6: Mixins in Sass :**

Write a Sass mixin that generates a box-shadow for a given element.

```
@mixin box-shadow($x-offset, $y-offset, $blur-radius, $spread-radius, $color) {  
  box-shadow: $x-offset $y-offset $blur-radius $spread-radius $color;  
}
```

- **Task 7: Parameters in Sass:**

Apply the updated mixin to style another set of cards with varying parameters, including some without specified values to test the default behaviour.

Updated Box-shadow Mixin with Default Values

```
@mixin box-shadow($x-offset: 0px, $y-offset: 2px,  
$blur-radius: 10px, $spread-radius: 0px, $color:  
  rgba(0, 0, 0, 0.1)) {  
  box-shadow: $x-offset $y-offset $blur-radius  
    $spread-radius $color;  
}
```

Usage Example with Default and Custom Parameters

Here is how you can use this updated mixin in your Sass file to apply a box-shadow to a new set of cards:


```

// Define some variables for reuse
$card-bg: #fff;
$card-padding: 20px;
$card-margin: 20px;
$card-radius: 5px;

// New set of cards with varying parameters
.card-default {
  background-color: $card-bg;
  padding: $card-padding;
  margin: $card-margin;
  border-radius: $card-radius;
  @include box-shadow(); // Uses all default values
}

.card-custom1 {
  background-color: $card-bg;
  padding: $card-padding;
  margin: $card-margin;
  border-radius: $card-radius;
  @include box-shadow(5px, 5px, 15px, 1px, rgba(0, 0, 0, 0.2)); // Custom values
}

.card-custom2 {
  background-color: $card-bg;
  padding: $card-padding;
  margin: $card-margin;
  border-radius: $card-radius;
  @include box-shadow(-5px, 10px, 20px, 3px, rgba(0, 0, 0, 0.3)); // Custom values
}

.card-partial {
  background-color: $card-bg;
  padding: $card-padding;
  margin: $card-margin;
  border-radius: $card-radius;
  @include box-shadow($blur-radius: 25px); // Only overriding blur radius
}

```

Example HTML to Demonstrate the Cards

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Card Shadows Example</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="cards">
    <div class="card-default">
      <h3>Default Card</h3>
      <p>This card uses the default box-shadow values.</p>
    </div>
    <div class="card-custom1">
      <h3>Custom Card 1</h3>
      <p>This card uses custom box-shadow values.</p>
    </div>
    <div class="card-custom2">
      <h3>Custom Card 2</h3>
      <p>This card uses custom box-shadow values.</p>
    </div>
    <div class="card-partial">
      <h3>Partial Override</h3>
      <p>This card overrides only the blur radius.</p>
    </div>
  </div>
</body>
</html>

```

Explanation:

1. **Mixin with Default Values:** The @mixin box-shadow now includes default values for all its parameters, allowing it to be used without any arguments if desired.
2. **Default Card:** This card uses the mixin without any arguments, applying the default box-shadow values.
3. **Custom Cards:** These cards override the default values with custom ones, demonstrating different shadow effects.
4. **Partial Override:** This card overrides only the `blur-radius` parameter, using default values for the others.

• Task 8: Functions in Sass:

Create a Sass function that calculates the contrast ratio between two colors based on the WCAG (Web Content Accessibility Guidelines).

Step-by-Step Explanation

1. **Calculate Relative Luminance:** Relative luminance is calculated using the sRGB values of the color. The formula for each color channel (R, G, B) is:

$$\text{Luminance} = 0.2126 \cdot R + 0.7152 \cdot G + 0.0722 \cdot B$$

The values for R, G, and B need to be transformed using a specific formula based on their sRGB values.

2. **Contrast Ratio:** The contrast ratio is calculated using the relative luminance values of the two colors:

$$\text{Contrast Ratio} = \frac{L_1 + 0.05}{L_2 + 0.05}$$

where L_1 is the relative luminance of the lighter color and L_2 is the relative luminance of the darker color.

Sass Function

Here is the Sass function that implements these calculations:

```

@function luminance($color) {
  $r: red($color) / 255;
  $g: green($color) / 255;
  $b: blue($color) / 255;

  @function transform($c) {
    @return if($c <= 0.03928, $c / 12.92, pow((( $c + 0.055) / 1.055), 2.4));
  }

  $r: transform($r);
  $g: transform($g);
  $b: transform($b);

  @return 0.2126 * $r + 0.7152 * $g + 0.0722 * $b;
}

@function contrast-ratio($color1, $color2) {
  $l1: luminance($color1);
  $l2: luminance($color2);

  @if $l1 > $l2 {
    @return ($l1 + 0.05) / ($l2 + 0.05);
  } @else {
    @return ($l2 + 0.05) / ($l1 + 0.05);
  }
}

```

Example Usage

Here is how you can use this function in your Sass code to calculate the contrast ratio between two colors:

```

$color1: #ffffff; // white
$color2: #000000; // black
$ratio: contrast-ratio($color1, $color2);

body {
  background-color: $color1;
  color: $color2;
  &:after {
    content: "Contrast Ratio: #{$ratio}";
    display: block;
    margin-top: 20
  }
}

```