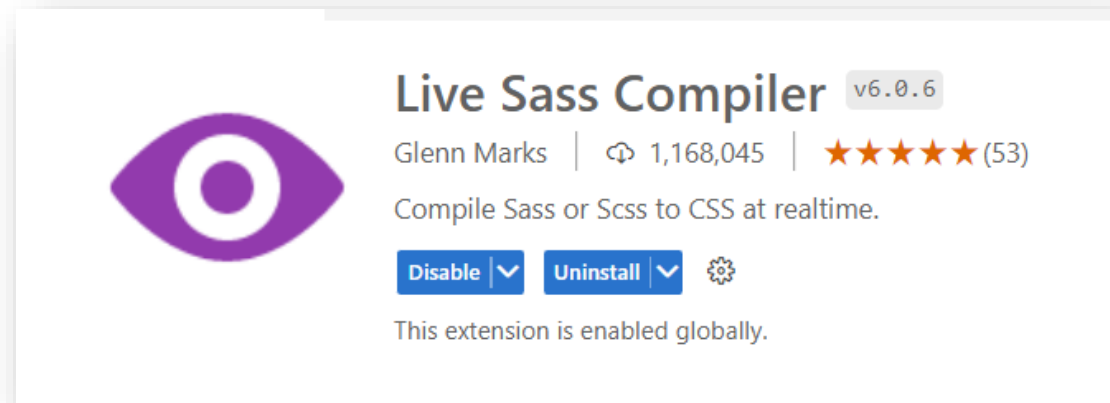


# Install SASS

- ✓ Install VS Code extension : **Live Sass Compiler**

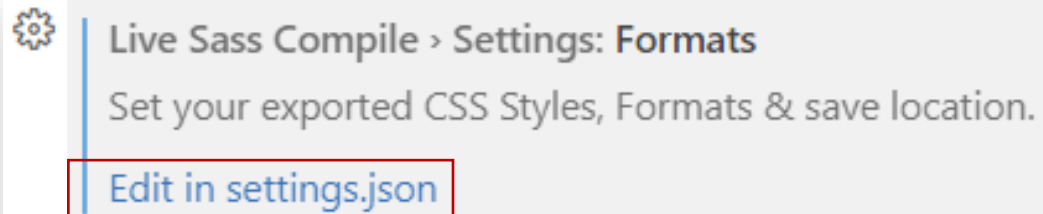


<https://www.youtube.com/watch?v=cpbNOYAW44g&t=183s>

# Setup SASS

- ✓ Configure The Live Sass Compile extension

Extension setting -> Live Sass Compile> Setting: Formats -> Edit in settings.json



```
"liveSassCompile.settings.formats": [  
  
  {  
    "format": "expanded",  
    "extensionName": ".css",  
    "savePath": "/css",  
  },  
  {  
    "format": "compressed",  
    "extensionName": ".min.css",  
    "savePath": "dist/css",  
  }  
],
```

## CHAPTER 8

*Sass*

# What will you learn today ?

## ✓ Understand

- ✓ What problems does **Sass** solve?
- ✓ What is a **CSS pre-processor**, what is **SASS** ?

## ✓ Manipulate

- Variable
- Nested rules
- @import
- @extend
- @Mixin



# What problems does **Sass** solve?

Stylesheets are getting larger, more complex, and harder to maintain.



CSS files

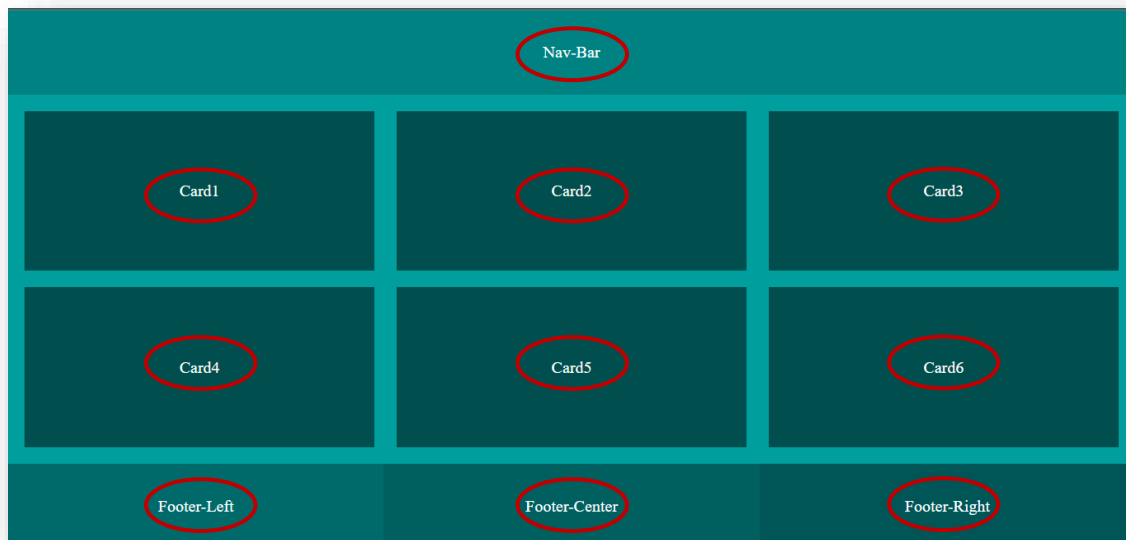


SCSS file

# Problem 1

If we want the **content** of each item centered. So, in CSS, we need to **combine every element** in **one style**.

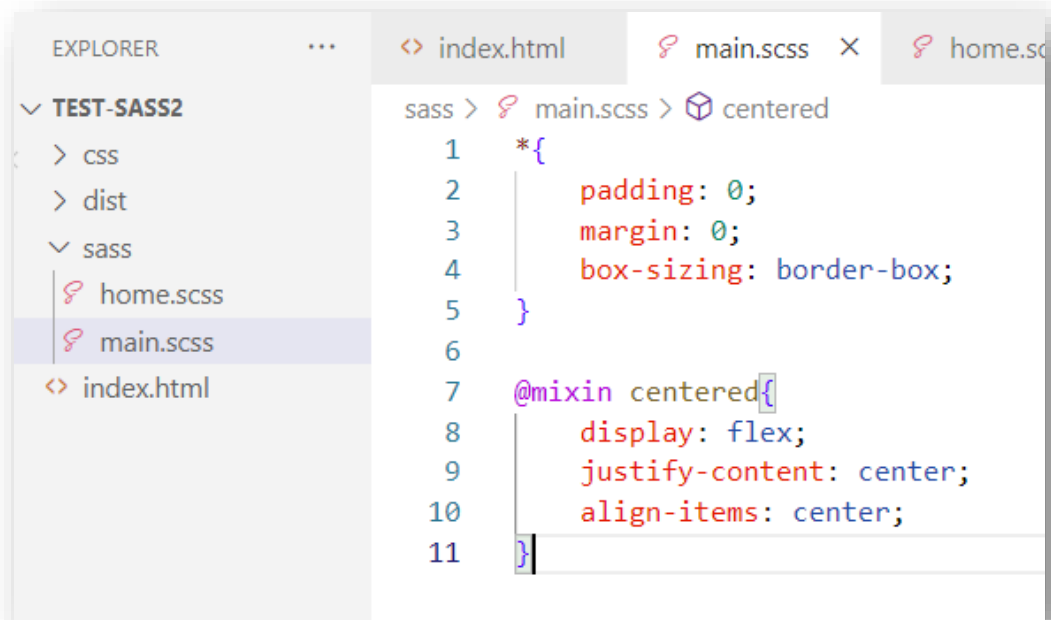
The **problem** is if we have **different sheets of files**, do we need to **set a style** like this **for every file**?



```
.card,  
.footer-right,  
.footer-center,  
.footer-left {  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}
```

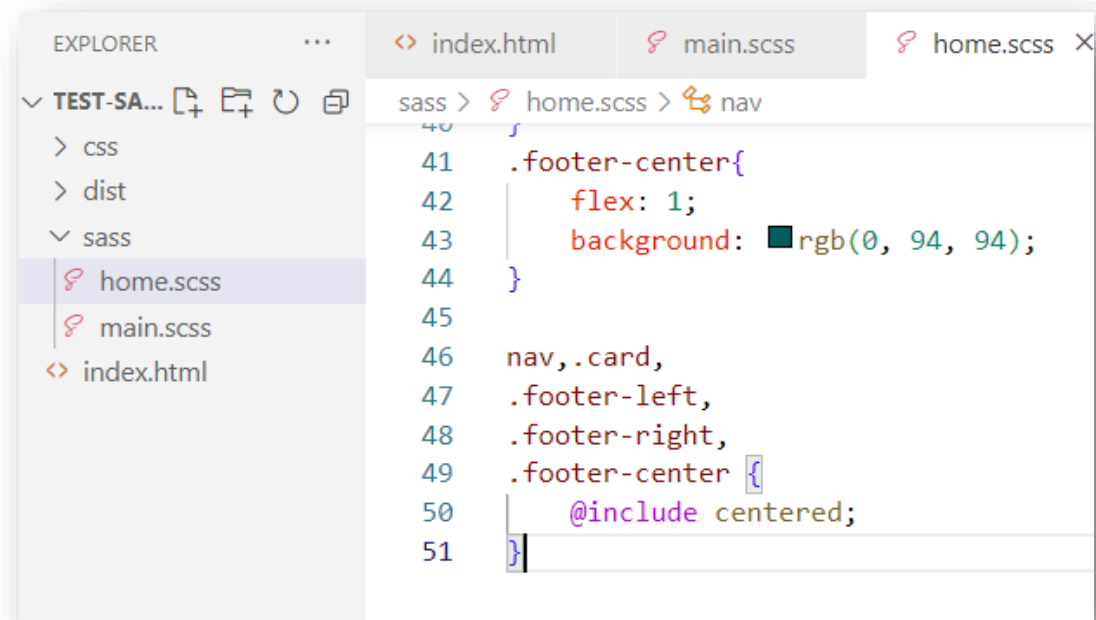
# Solution 1

In sass, we have **@mixin** and **@include** directive for solve this problem.



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer shows a project named 'TEST-SASS2' with folders 'css' and 'dist', and a 'sass' folder containing 'home.scss', 'main.scss', and 'index.html'. The 'main.scss' file is selected and open in the editor. The breadcrumb at the top of the editor reads 'sass > main.scss > centered'. The code in the editor is as follows:

```
1  *{
2      padding: 0;
3      margin: 0;
4      box-sizing: border-box;
5  }
6
7  @mixin centered{
8      display: flex;
9      justify-content: center;
10     align-items: center;
11 }
```



The screenshot shows the VS Code editor with the Explorer sidebar on the left. The Explorer shows the same project 'TEST-SASS2'. The 'home.scss' file is selected and open in the editor. The breadcrumb at the top of the editor reads 'sass > home.scss > nav'. The code in the editor is as follows:

```
40
41
42 .footer-center{
43     flex: 1;
44     background: #009494;
45 }
46
47 nav, .card,
48 .footer-left,
49 .footer-right,
50 .footer-center {
51     @include centered;
52 }
```

# Problem 2

Normally, on a **website**, we have a **few colors** that are **used**. If we want to **change** the **white color** of each content here *what should we do?*

Change one by one?

```
nav{
  flex: 1;
  background: teal;
  color: white;
}
main{
  flex:4;
  background: rgb(0, 156, 156);
  color: white;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  padding: 20px;
  gap: 20px;
}

footer{
  flex: 1;
  background: rgb(1, 176, 176);
  display: flex;
  color: white;
}
```



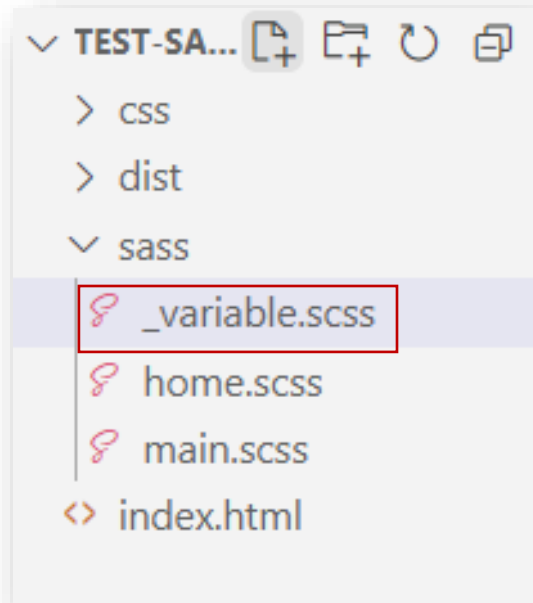
# Solution 1

## Step to solve this problem:

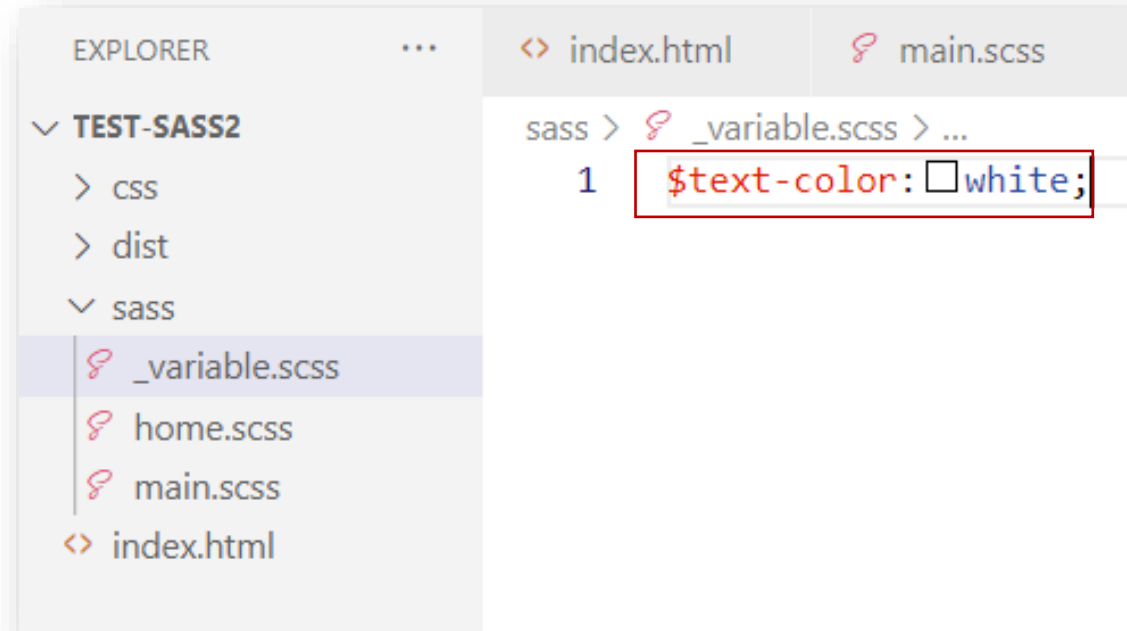
1. Create a component file called `_variable.scss`
2. Define the variable `$text-color`
3. `@import` as `global` in `main.scss`
4. Called it to use in `page` files
5. If we want to **change** the value color of **variable text-color** in `_variable.scss` it will **change** all **color properties** that use `$text-color`.

# Solution 1

1

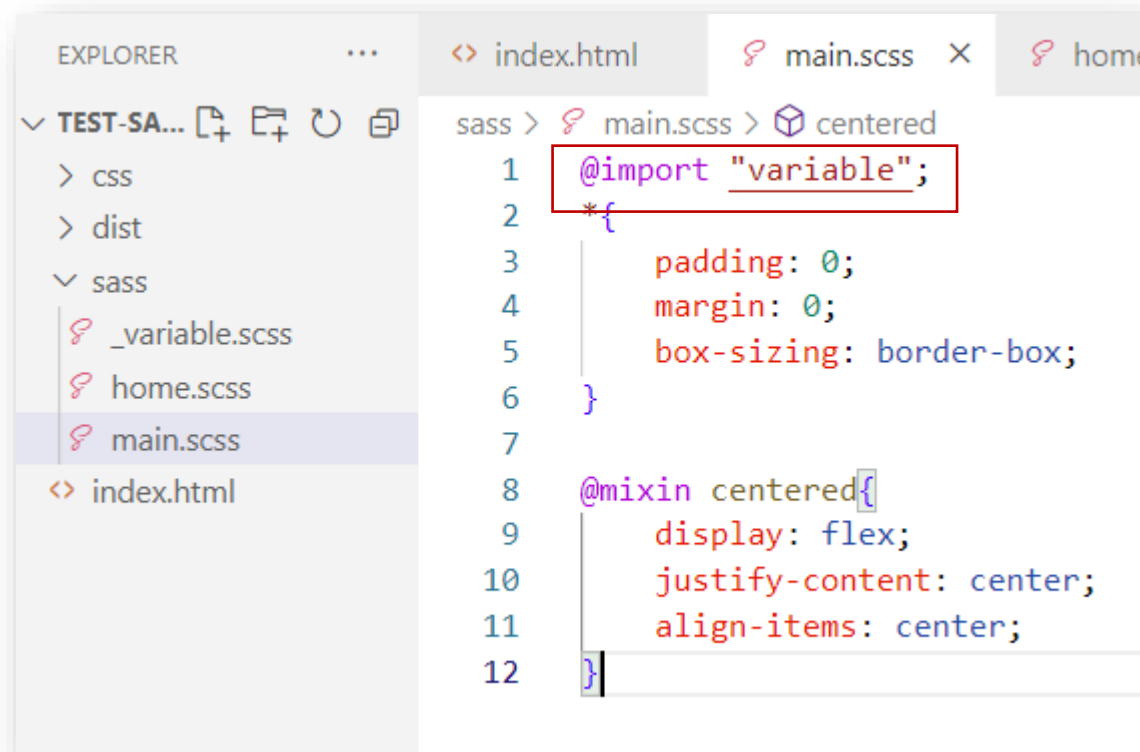


2



# Solution 1

3

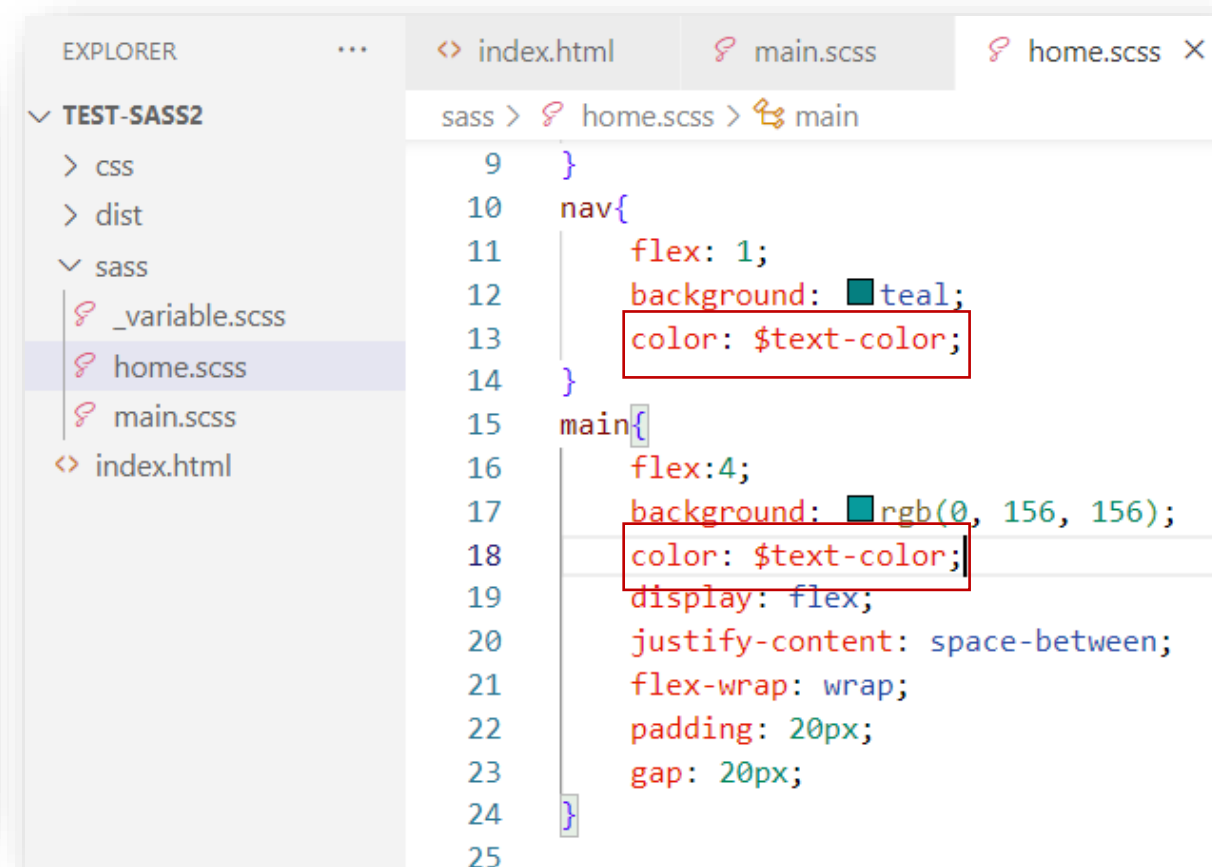


EXPLORER

- TEST-SA...
- > css
- > dist
- ▼ sass
  - \_variable.scss
  - home.scss
  - main.scss
- <> index.html

```
sass > main.scss > centered
1  @import "variable";
2  *{
3      padding: 0;
4      margin: 0;
5      box-sizing: border-box;
6  }
7
8  @mixin centered{
9      display: flex;
10     justify-content: center;
11     align-items: center;
12 }
```

4



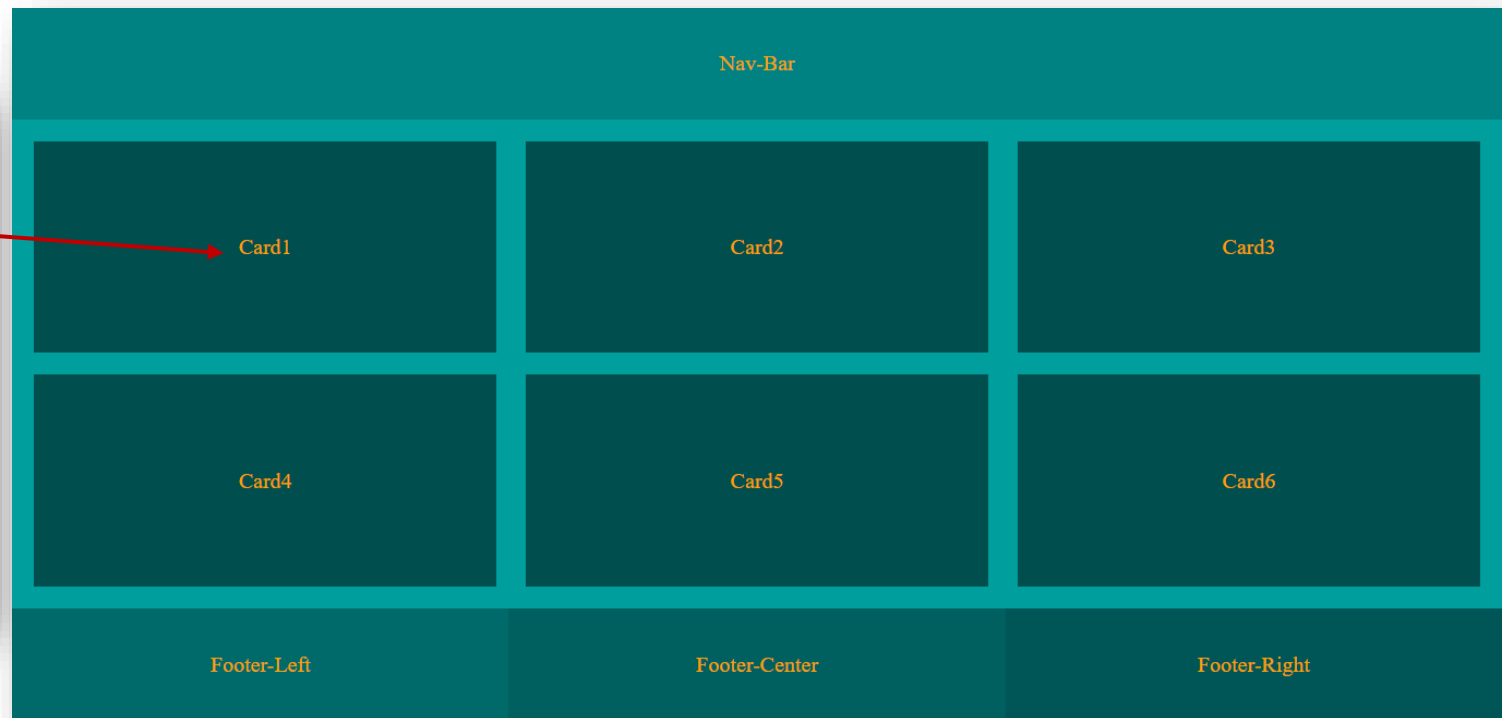
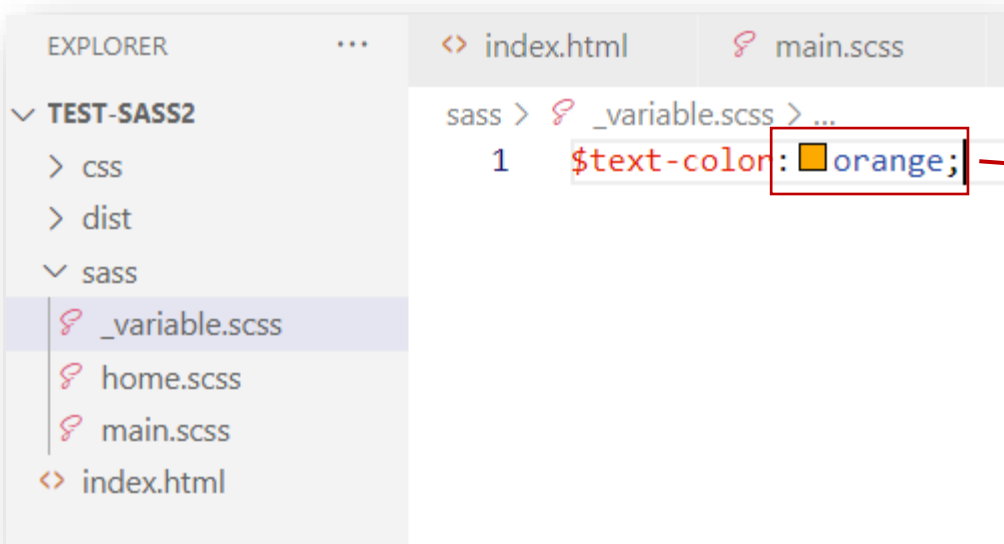
EXPLORER

- TEST-SASS2
- > css
- > dist
- ▼ sass
  - \_variable.scss
  - home.scss
  - main.scss
- <> index.html

```
sass > home.scss > main
9  }
10 nav{
11     flex: 1;
12     background: teal;
13     color: $text-color;
14 }
15 main{
16     flex:4;
17     background: rgb(0, 156, 156);
18     color: $text-color;
19     display: flex;
20     justify-content: space-between;
21     flex-wrap: wrap;
22     padding: 20px;
23     gap: 20px;
24 }
25 }
```

# Solution 1

5



# What is **SASS** ?



<https://www.youtube.com/watch?v=akDIJa0AP5c>



10 MIN



1

# What is **SASS** ?



- ✓ What does **SASS** stand for?
- ✓ Why **SASS**?
- ✓ How it can be **used**?



10 MIN



1

# What is **SASS** ?



- ✓ stands for Syntactically Awesome Stylesheet
- ✓ is an extension of CSS
- ✓ is a CSS pre-processor
- ✓ is completely compatible with all versions of CSS
- ✓ reduces repetition of CSS and therefore saves time



10 MIN

# How does Sass work?

- ✓ A **browser** does not **understand Sass** code.
- ✓ We need a **pre-processor** to **convert Sass** code into **CSS**.

```
● ● ●  
$myFont: Helvetica, sans-serif;  
$myColor: red;  
$myFontSize: 18px;  
$myWidth: 680px;  
  
body {  
  font-family: $myFont;  
  font-size: $myFontSize;  
  color: $myColor;  
}  
  
.container {  
  width: $myWidth;  
}
```

*SASS file*

SASS PRE PROCESSOR



```
● ● ●  
  
body {  
  font-family: Helvetica, sans-serif;  
  font-size: 18px;  
  color: red;  
}  
  
.container {  
  width: 680px;  
}
```

*CSS file*

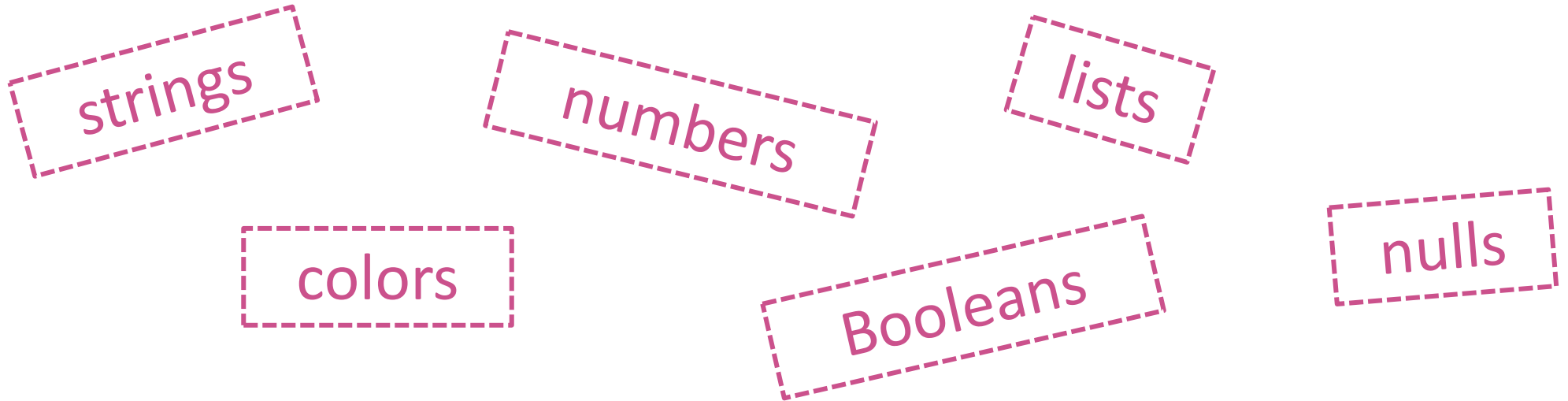




10 MIN

# Sass variable

- ✓ Sass Variables **store information** that you can **re-use** later.



- ✓ Sass uses the **\$symbol**, followed by a **name** to **declare** a new **variables**

*Example: **\$variable-name: value;***



5 MIN

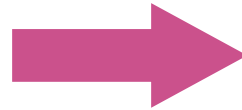


1

# What will be the generated CSS?

```
● ● ●  
$myColor: red;  
  
h1 {  
  $myColor: green;  
  color: $myColor;  
}  
  
p {  
  color: $myColor;  
}
```

*SASS file*



```
● ● ●  
h1 {  
  color: green;  
}  
  
p {  
  color: red;  
}
```

*CSS file*



5 MIN



1

# Activity 1

Use **variables** in difference block in **main.css**





5 MIN

# SASS Nested rule


TO CHANGE  
- Nested rule not properties

Many CSS properties have the same **prefix**, like **font**-family, **font**-size and **font**-weight or **text**-align, **text**-transform and **text**-overflow.


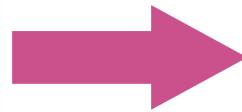


5 MIN

# SASS nested properties



```
font:{  
  family: Helvetica, sans-serif;  
  size: 18px;  
  weight: bold;  
}  
  
text:{  
  align: center;  
  transform: lowercase;  
  overflow: hidden;  
}
```

*SASS file*

```
font-family: Helvetica,sans-serif;  
font-size: 18px;  
font-weight: bold;  
text-align: center;  
text-transform: lowercase;  
text-overflow: hidden;
```

*CSS file*



10 MIN



1

## Activity 2

- ✓ Open README.txt file and write **nested properties** and **elements** In **main.scss**

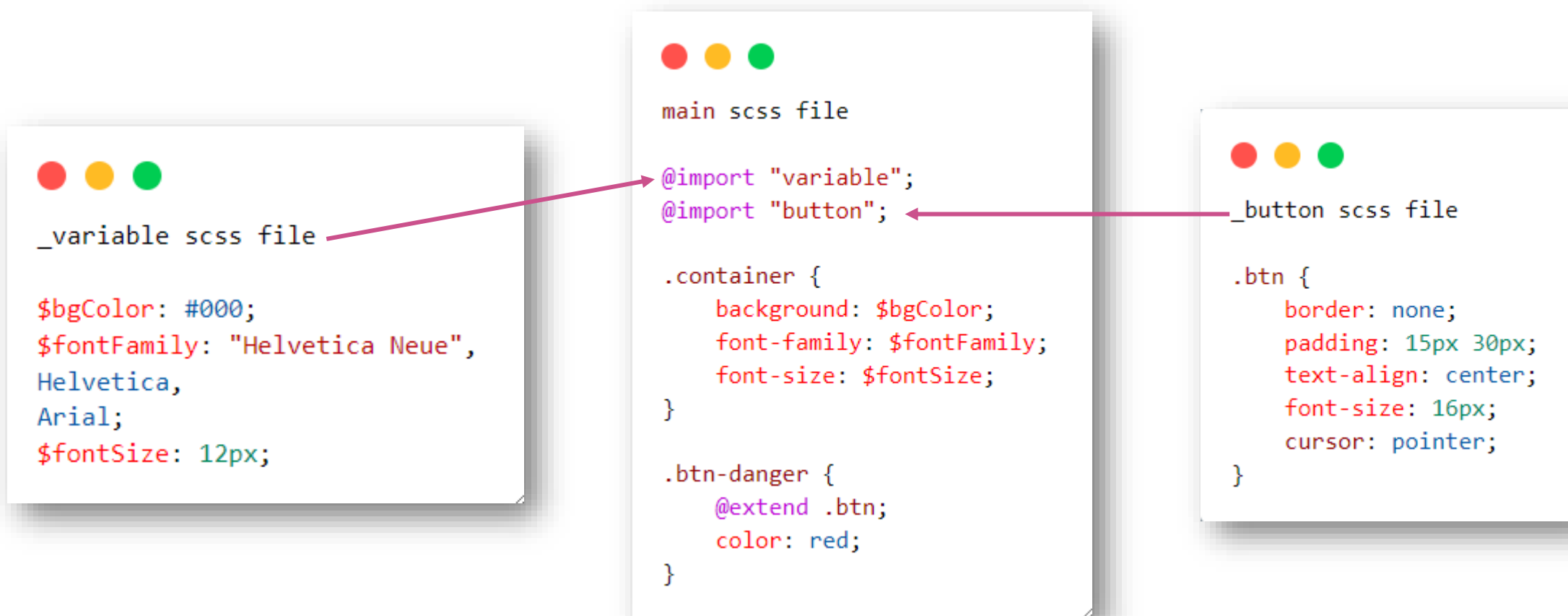




5 MIN

# Sass imports

The **@import** directive allows you to **include** the content of **one file to another**.





5 MIN



1

# Activity 3

Import all the component files to main file







5 MIN

# Sass @extends

- ✓ The `@extend` directive lets you share a set of CSS properties from one selector to another.
- ✓ The `@extend` directive is useful if you have almost identically styled elements that only differ in some small details.

```
.btn {  
  border: none;  
  padding: 15px;  
  text-align: center;  
  font-size: 16px;  
  cursor: pointer;  
}  
  
.btn-report {  
  @extend .btn;  
  background: teal;  
}  
  
.btn-submit {  
  @extend .btn;  
  background: orange;  
  color: white;  
}
```



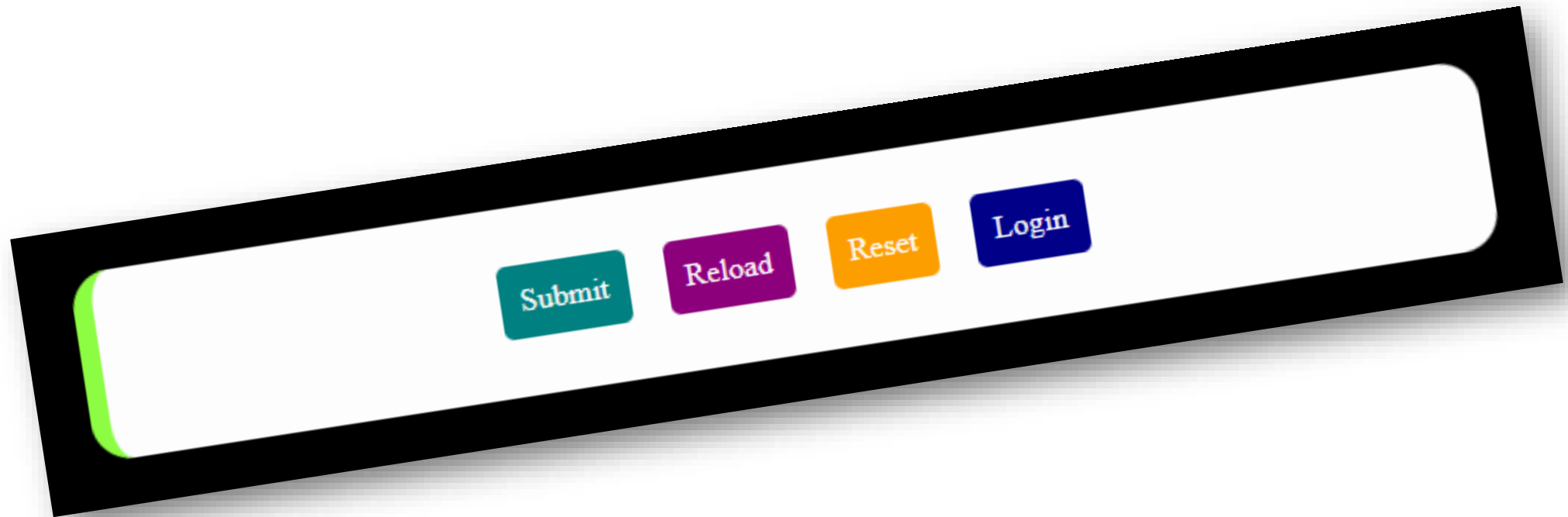
10 MIN



1

# Activity 4

Let's create four buttons follow the picture below by using **Sass**....!





5 MIN

So with SASS you can create style component button in a file **\_button.scss**

The image shows two side-by-side screenshots of a code editor, likely VS Code, illustrating the structure of SASS files for button styles.

**Left Screenshot:** The Explorer panel on the left shows a project structure with folders 'css' and 'dist', and a 'sass' folder containing '\_button.scss', 'style.scss', and 'index.html'. The 'style.scss' file is selected in the editor. The code in 'style.scss' includes an '@import' for '\_button.scss', a wildcard selector '\*' with padding and margin, a 'body' background color, and a '.group-btns' class with various flex and padding properties.

```
sass > style.scss > .group-btns
1  @import 'button';
2
3  * {
4    padding: 0;
5    margin: 0;
6  }
7
8  body {
9    background: #000;
10 }
11
12 .group-btns {
13   width: 50%;
14   background: white;
15   display: flex;
16   justify-content: center;
17   padding: 20px;
18   margin: auto;
19   margin-top: 100px;
```

**Right Screenshot:** The Explorer panel shows the same project structure, but the '\_button.scss' file is selected in the editor. The code in '\_button.scss' defines a base '.btn' class with color, padding, margin, border, border-radius, and cursor. It also defines two specific button classes: '.btn-submit' (teal background) and '.btn-reset' (orange background), both extending the base '.btn' class.

```
sass > _button.scss > ...
1  .btn {
2    color: white;
3    padding: 10px;
4    margin: 10px;
5    border: none;
6    border-radius: 5px;
7    cursor: pointer;
8  }
9
10 .btn-submit {
11   @extend .btn;
12   background: teal;
13 }
14
15 .btn-reset {
16   @extend .btn;
17   background: orange;
18 }
19
```

Submit

Reload

Reset

Login



5 MIN

# Sass @mixin

- ✓ The @mixin directive lets you create CSS code that is to be reused throughout the website.
- ✓ The @include directive is created to let you use (include) the mixin.

Define

```
@mixin name {  
  property: value;  
  property: value;  
  ...  
}
```

Use

```
selector {  
  @include mixin-name;  
}
```



5 MIN

# Sass @mixin



```
@mixin important-text {  
  color: red;  
  font-size: 25px;  
  font-weight: bold;  
  border: 1px solid blue;  
}
```



```
.danger {  
  @include important-text;  
  background-color: green;  
}
```



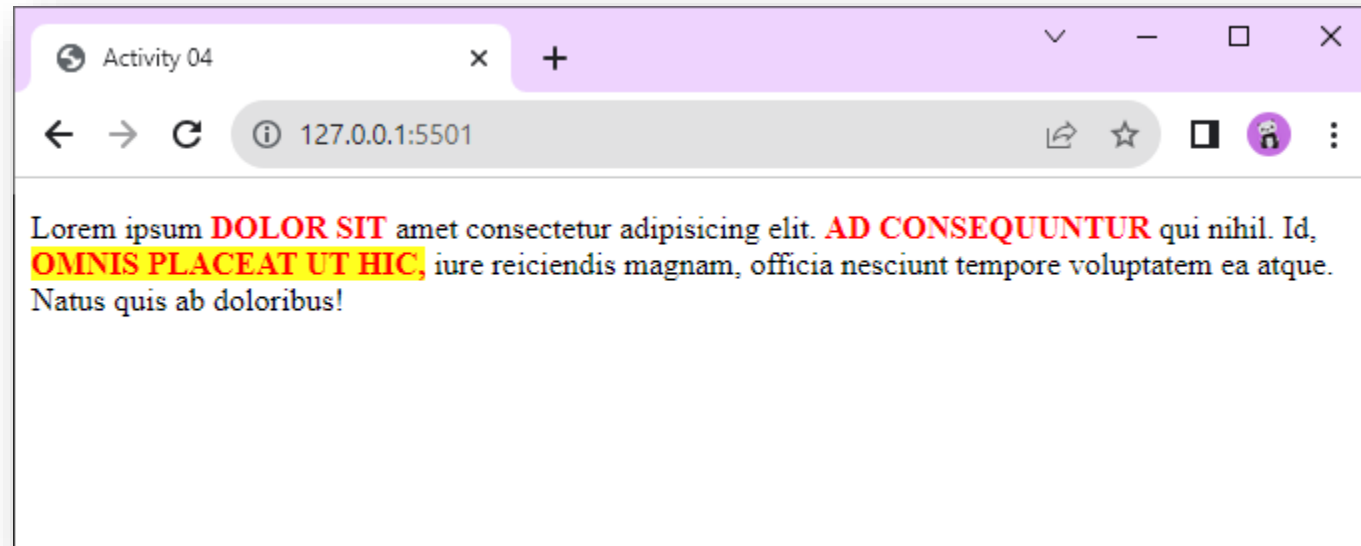
10 MIN



1

# Activity 5

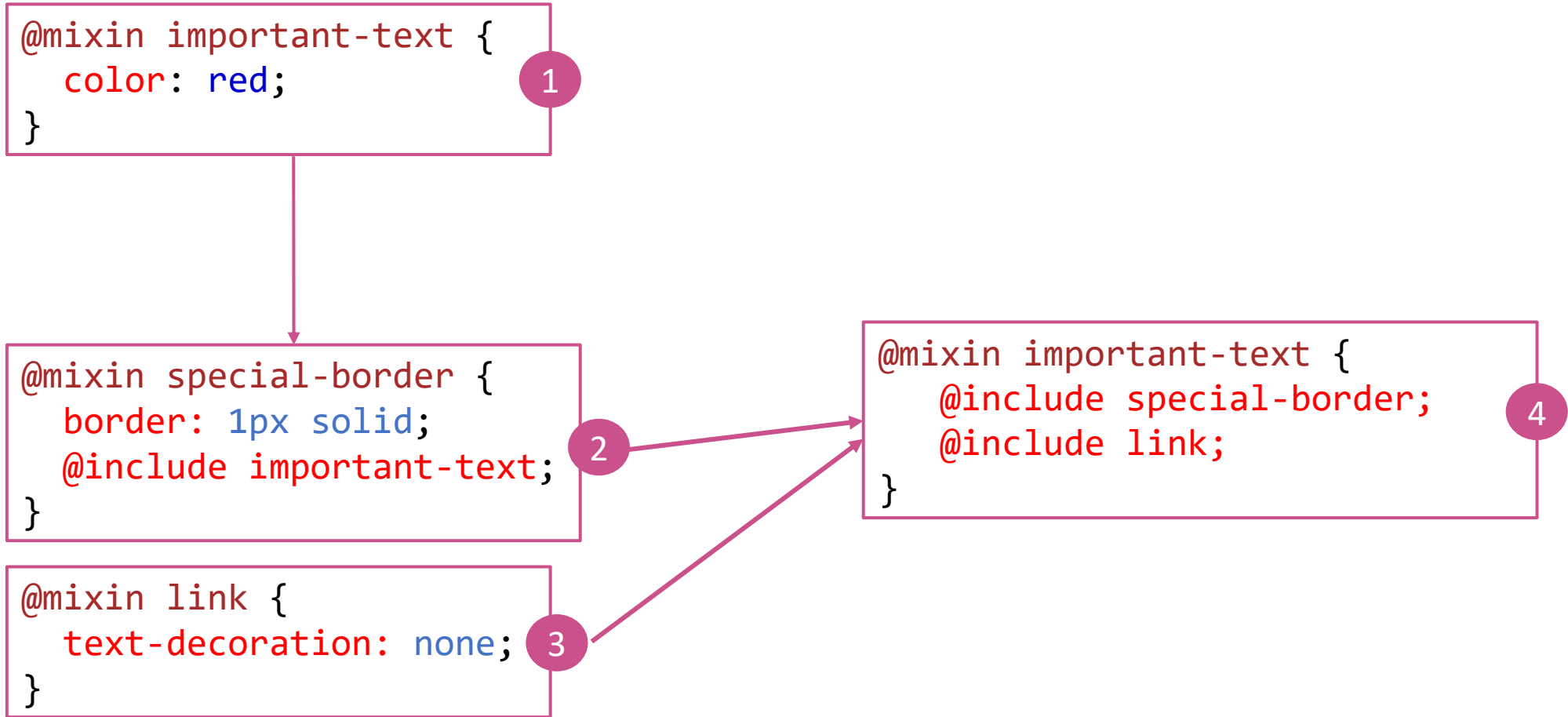
- ✓ Write SASS to change text in span tag to **BOLD** and color **RED**.
- ✓ If span has id="remark" change background-color to yellow.





5 MIN

# Sass @mixin





5 MIN

# @mixin parameters

## @mixin parameters

```
/* Define mixin with two arguments */
@mixin bordered($color, $width) {
  border: $width solid $color;
}

.myArticle {
  @include bordered(blue, 1px); // Call mixin with two values
}

.myNotes {
  @include bordered(red, 2px); // Call mixin with two values
}
```

## Default @mixin parameters

```
@mixin bordered($color: blue, $width: 1px) {
  border: $width solid $color;
}
```