

01-Sass and Less

01-Sass Fundamentals

What is SaSS?

Syntactically Awesome Style Sheets

Sass is preprocessor of CSS.

Sass Setup

1. Install nodejs
2. Verify node version
 - (base) ashujauhari@Ashus-MacBook-Air sass % **node -v.**
3. Verify the npm version > **npm --version**
4. Create project folder : 01-sass-variable-inheritance-mixin
5. Switch to project folder
6. > **npm init -y** // This will create a package.json
7. Create the project structure

```
project/
├─ src/
│   └─ styles/
│       └─ main.scss
│       └─ _partial.scss
├─ dist/
│   └─ styles/
│       └─ main.css
└─ package.json
```

8. 01-sass-variable-inheritance-mixin % **npm install sass --save**
9. Create a main.scss file in src folder

```
$color: blue ;
body{
  background : $color;
}
```

10. Update package.json – Change test to sass

```
"scripts": {
  "sass": "sass src/styles/main.scss dist/styles/main.css"
},
```

Now automatic compilation will occur on change detection

11. This way we can run> **npm run sass**
Otherwise give command: **npx sass src/styles/main.scss dist/styles/main.css**

12. Verify **main.css** the compiled version of **main.scss** file

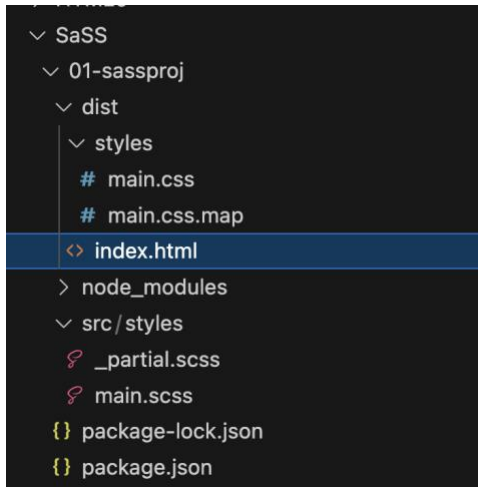
13. Create index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link rel="stylesheet" href="styles/main.css" />
    <title>Sass Sandbox</title>
  </head>
  <body>
    <header>
      <h1>Sass Demo</h1>
    </header>
  </body>
</html>
```

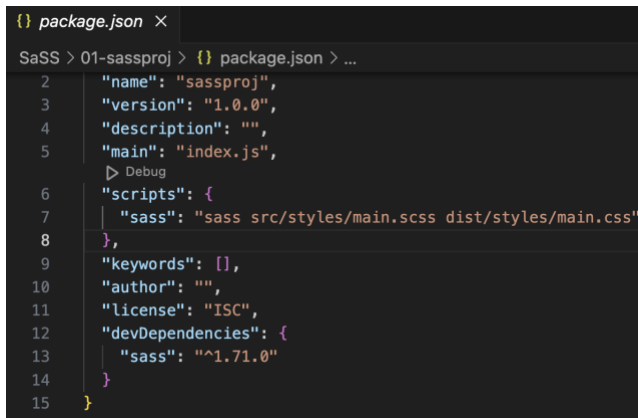
14. Write main.scss for testing

```
$color: blue ;
body{
  background : $color;
}
```

➤ npm run sass - It will compile the scss into css and it will run.



15.



16.

Variables for css properties

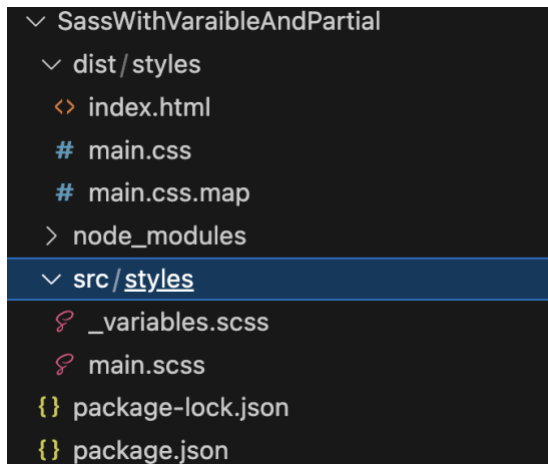
Lab: Create variables in file and use them SCSS file to implement in html file.

1. Create a project
2. **npm init -y**
3. **npm install sass --save**
4. Update package.json – Change test to sass


```

      "scripts": {
        "sass": "sass src/styles/main.scss dist/styles/main.css"
      },

```
5. **npm run sass**
6. Make sure about below project structure



7. Create `_variables.scss` file to store variable for css properties

```
$color: rgb(255, 183, 0);  
$secondary-color: rgb(135, 235, 188);  
$light-color: #f4f4f4;  
$dark-color: #333;  
$primary-color: steelblue;  
$font-stack: Arial, Helvetica, sans-serif;
```

8. Use the above `_variables.scss` file in `main.scss`

```
@import 'variables';  
  
* {  
  margin: 0 ;  
  padding: 0;  
}  
  
body{  
  background : $secondary-color;  
  color: $dark-color;  
  font-family: $font-stack;  
  line-height: 1.5;  
}
```

9. Create `index.html` in `dist` folder

```
10. <!DOCTYPE html>  
  
<html lang="en">  
  
  <head>  
  
    <meta charset="UTF-8" />  
  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```

<meta http-equiv="X-UA-Compatible" content="ie=edge" />
<link rel="stylesheet" href="main.css" />
<title>Sass Sandbox</title>
</head>
<body>
  <h1>Sass Sandbox</h1>
</header>
<section class="section section-a">
  <h3>Section A</h3>
  <p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ad ea tempore
    optio molestiae! Fuga suscipit asperiores, provident odio distinctio
    dicta aspernatur inventore illo, rerum odit nobis eligendi doloribus
    iste architecto?
  </p>
  <a href="#" class="btn-light">Read More</a>
</section>
<section class="section section-b">
  <h3>Section B</h3>
  <p>
    Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ad ea tempore
    optio molestiae! Fuga suscipit asperiores, provident odio distinctio
    dicta aspernatur inventore illo, rerum odit nobis eligendi doloribus
    iste architecto?
  </p>
  <a href="#" class="btn-dark">Read More</a>
</section>
</body>
</html>

```

11. Link the main.css in index.html -><link rel="stylesheet" href="main.css" />

12. **npm run sass**

13. Verify the page

Multiple SCSS and wants to compiled in one main.css

File1.scss

Ashu Jauhari

```
$color: blue ;

body{
  background : $color;
}
```

File2.css

```
$tcolor: rgb(255, 0, 195) ;

body{
  color : $tcolor;
}
```

File3.scss

```
$fsize: 16px ;

body{
  font-size: $fsize;
}
```

Main.scss

```
@import 'file1';
@import 'file2';
@import 'file3';
```

Directly execute main.scss> `npx sass src/styles/main.scss dist/styles/main.css`

```
01-SassFundamental.docx  main.css  main.css.map  file2.scss  file3.scss  main.scss

HTMLCSSJS > SaSS > 01-sassproj > src > styles > main.scss
1  @import 'file1';
2  @import 'file2';
3  @import 'file3';

PROBLEMS  OUTPUT  PORTS  DEBUG CONSOLE  TERMINAL
zsh - 01-sassproj +
(base) ashujauhari@Ashus-MacBook-Air 01-sassproj % npx sass src/styles/main.scss dist/styles/main.css
```

Using script in package.json

```
"scripts": {
  "sass": "sass src/styles/main.scss dist/styles/main.css"
}
```

01-sassproj % **npm run-script sass**

Compiles style.scss to style.css.

\$ sass style.scss:style.css

Compiles light.scss and dark.scss to light.css and dark.css.

\$ sass light.scss:light.css dark.scss:dark.css

```
# Compiles all Sass files in themes/ to CSS files in public/css/.
$ sass themes:public/css
```

<https://sass-lang.com/documentation/cli/dart-sass/#:~:text=If%20the%20%2D%2Dno%2Dsource,with%20other%20source%20map%20options.>

Nesting and Structuring

Lab: Apply the css for section and subsection

1. Update main.scss.

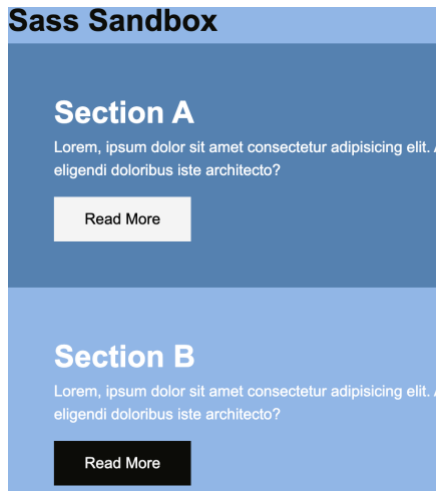
```
//-----Nesting and Structuring-----
```

```
header{
  background: $color;
  color: $dark-color;
  padding: 1rem;

  h1 {
    text-align: center;
  }
}

.section {
  padding: 3rem;
  h3 {
    font-size: 2rem;
  }
  &-a { //& denote parent section
    background: $primary-color;
    color: $color
  }
  &-b {
    background: $secondary-color;
    color: $color;
  }
}
```

2. Compile it again -> npm run sass
3. Validate the output



Inheritance

Mixins can help you avoid repetition, create dynamic styles, and modularize your code. However, they also have some drawbacks that you should be aware of before using them. In this article, we will explore the benefits and drawbacks of using mixins in Sass and how to use them effectively.

Lab: Write scss for buttons where common button properties and specific would be in other section in main.scss.

1. Update main.scss

```
/*-----Inheritance-----*/
%btn-shared {
  display: inline-block;
  padding: 0.7rem 2rem;
  border: none;
  cursor: pointer;
  text-decoration: none;
  margin-top: 1rem;
}

.btn {
  &-light {
    @extend %btn-shared;
    background-color: $light-color;
    color: #fff // it is UDF
  }
}
```


Function and Mixin

Lab: rotate the button and call set-text-color function to rotate the button.

1. Create _function.scss

```
//Set Text color

@function set-text-color ($color) {
  @if ( lightness($color) > 50){
    @return #000;
  } @else
  {
    @return #fff;
  }
}

//Transform
@mixin transform ($property)
{
  --webkit-transform : $property;
  --ms-transform : $property;
  transform : $property;
}
```

2. Update main.scss

```
3. @import 'function';
4. .btn {
  &-light {
    @extend %btn-shared;
    background-color: $light-color;
    color: set-text-color($light-color); // it is UDF

    &-hover {
      @include transform(rotate(20deg));
      background-color: darken($light-color, 10%);
    }
  }

  &-dark {
    @extend %btn-shared;
```

```
background-color: $dark-color;
color: #fff;

&:hover {
  @include transform(rotate(-20deg));
  background-color: lighten($dark-color, 10%);
}
```

2. >npm run sass

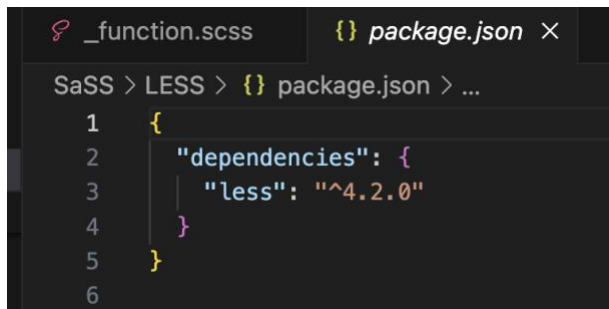


Less

/Users/ashujauhari/Data/NestJS-NodeJS-React-Mongo/HTML5/HTMLCSSJS/SaSS/LESS

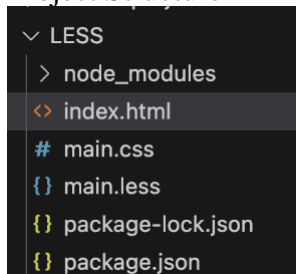
Less Setup

1. Create Project in VScode.
2. > npm install --no-bin-link less .



```
1 {
2   "dependencies": {
3     "less": "^4.2.0"
4   }
5 }
6
```

3. Project Structure



```

  LESS
  > node_modules
  <> index.html
  # main.css
  {} main.less
  {} package-lock.json
  {} package.json
```

4. Write main.less



```
@primarycolor: #FF7F50;
@color: #800080;

h2{
  color: @primarycolor;
}

h3{
  color: @color;
}
```

(base) ashujauhari@Ashus-MacBook-Air less % **./node_modules/less/bin/lessc main.less**

main.css



```
<!DOCTYPE html>
<head>
  <link rel="stylesheet" href="main.css" type="text/css" />
</head>
<body>
  <h2>A simple example of Less</h2>
  <h3>Hello Less!</h3>
</body>
</html>
```

SaSS Vs Less

Base	SaSS	Less
Language	It is based on Ruby	Initially based in ruby but ported to JavaScript now.
Error Reporting	Reports errors made in syntax.	More precise error reporting with an indication of the location of the error.
Syntax	Use “\$” for variables and “@” for mixins	Use “@” for variables.