FRAMEWORK

SASS / SCSS

What is a CSS preprocessor?

- A **CSS preprocessor** is a program that lets you generate **CSS** from the **preprocessor's** own unique syntax.
- Each **CSS preprocessor** has its own syntax that they compile into regular CSS so that browsers can render it on the client side.

SASS/SCSS

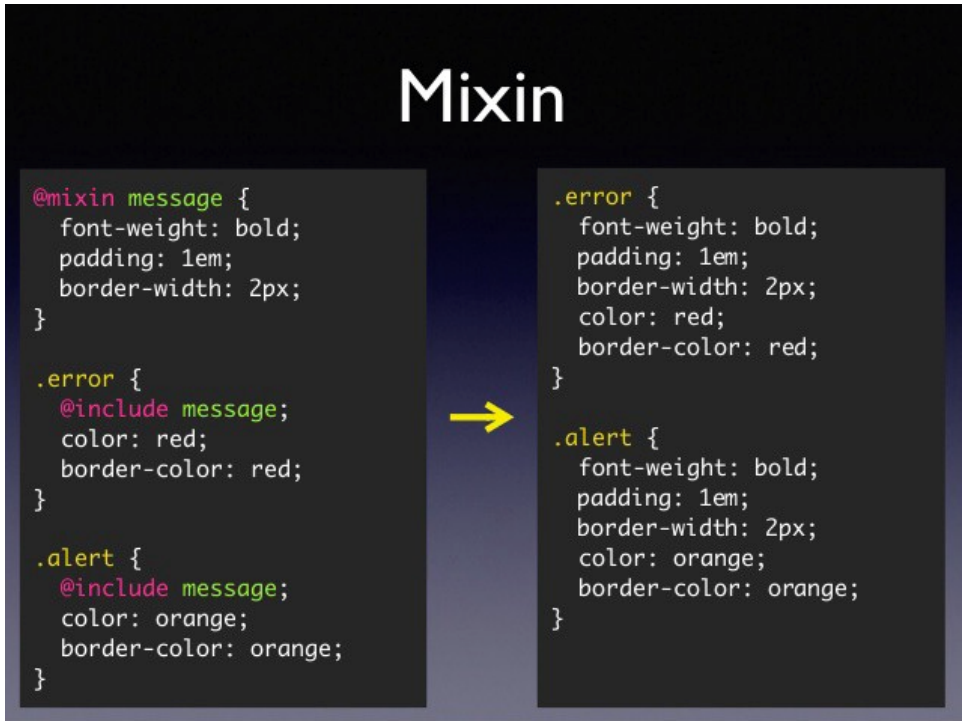
LESS

STYLUS

POSTCSS

Why should I use preprocessor?

- It makes CSS code more organized.
- With the power of using **variables** and **functions**, it provides more readable code.
- Maintaining code will be easy and in the long run it will be easier to edit as well.
- CSS preprocessors also provide the option of using **mixins** (reusability, cleaner and DRY-er code)



SASS / SCSS

SASS — (.sass) Syntactically Awesome Style Sheets.

SCSS — (.scss) Sassy Cascading Style Sheets.

- **Variables**
- **Nested Properties**
- **@import and Partials**
- **@mixin and @include**
- **@extend Directive**

There are **two types of syntaxes** while writing CSS in Sass.

1. Files with the **.sass** extension allows us to write CSS ***without semicolons*** and ***without curly brackets.***

```
.button
  padding: 3px 10px
  font-size: 12px
  border-radius: 3px
  border: 1px solid #e1e4e8
```

2. Files with the **.scss** extension are the CSS we normally write, where each CSS declaration ends ***with a semicolon*** and each selector is written ***in curly brackets***.

```
.button {
  padding: 3px 10px;
  font-size: 12px;
  border-radius: 3px;
  border: 1px solid #e1e4e8;
}
```

SASS / SCSS

SASS / SCSS - Variables

```
$variablename: value;
```

style.scss

```scss
$nav-font: 'Sansita Swashed', cursive;
$main-font: 'Lato', sans-serif;
$bg-color: #f5f5f5;
$main-dark: #0b0b0b;
$main-color: #606060;

body {
    font-family: $main-font;
    color: $main-color;
}
nav {
    width: 100%;
    height: 70px;
    background: $bg-color;
    box-shadow: 0 1px 5px $main-dark;
    ul {
        margin: 0 10%;
        li {
            display: inline-block;
            font-size: 25px;
            line-height: 70px;
            margin: 0 20px;
            a {
                color: $main-color;
            }
            &:hover a {
                color: $main-dark;
            }
        }
    }
}
```

style.css

```css
body {
  font-family: "Lato", sans-serif;
  color: #606060;
}

nav {
  width: 100%;
  height: 70px;
  background: #f5f5f5;
  -webkit-box-shadow: 0 1px 5px #0b0b0b;
          box-shadow: 0 1px 5px #0b0b0b;
}

nav ul {
  margin: 0 10%;
}

nav ul li {
  display: inline-block;
  font-size: 25px;
  line-height: 70px;
  margin: 0 20px;
}

nav ul li a {
  color: #606060;
}

nav ul li:hover a {
  color: #0b0b0b;
}
```

- Variables
- Nested Properties
- @import and Partials
- @mixin and @include
- @extend Directive

SASS / SCSS – Variable Scope

Sass variables are **only available** at the level of nesting **where they are defined.**

```scss
$myColor: red;

h1 {
  $myColor: green;
  color: $myColor;
}

p {
  color: $myColor;
}
```

SCSS Syntax

```css
h1 {
  color: green;
}

p {
  color: red;
}
```

CSS Output

```scss
$myColor: red;

h1 {
  $myColor: green !global;
  color: $myColor;
}

p {
  color: $myColor;
}
```

```css
h1 {
  color: green;
}

p {
  color: green;
}
```

- Variables
- Nested Properties
- @import and Partials
- @mixin and @include
- @extend Directive

SASS / SCSS – Nesting

- Sass lets you nest CSS selectors in the same way as HTML.

```scss
nav {
    width: 100%;
    height: 70px;
    background: $bg-color;
    box-shadow: 0 1px 5px $main-dark;
    ul {
        margin: 0 10%;
        li {
            display: inline-block;
            font-size: 25px;
            line-height: 70px;
            margin: 0 20px;
            a {
                color: $main-color;
            }
            &:hover a {
                color: $main-dark;
            }
        }
    }
}
```

style.scss

```css
nav {
  width: 100%;
  height: 70px;
  background: #f5f5f5;
  -webkit-box-shadow: 0 1px 5px #0b0b0b;
          box-shadow: 0 1px 5px #0b0b0b;
}

nav ul {
  margin: 0 10%;
}

nav ul li {
  display: inline-block;
  font-size: 25px;
  line-height: 70px;
  margin: 0 20px;
}

nav ul li a {
  color: #606060;
}

nav ul li:hover a {
  color: #0b0b0b;
}
```

style.css

SASS / SCSS – Nesting

- Many CSS properties have the same prefix, like *font-family, font-size* and *font-weight* or *text-align, text-transform* and *text-overflow.*
- With Sass you can write them as nested properties

```
font: {
  family: Helvetica, sans-serif;
  size: 18px;
  weight: bold;
}

text: {
  align: center;
  transform: lowercase;
  overflow: hidden;
}
```

SCSS Syntax

```
font-family: Helvetica, sans-serif;
font-size: 18px;
font-weight: bold;

text-align: center;
text-transform: lowercase;
text-overflow: hidden;
```

CSS Output

- Variables
- Nested Properties
- @import and Partials
- @mixin and @include
- @extend Directive

SASS / SCSS – @import

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

```
@import ”file path”;
```

```scss
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
}
```

reset.scss

```scss
@import "reset";

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}
```

main.scss

```css
html, body, ul, ol {
  margin: 0;
  padding: 0;
}

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}
```

CSS Output (main.css)

SASS / SCSS – @import Partials

- By default, Sass transpiles all the .scss files **directly.**
- However, when you want to **import a file**, *you do not need the file to be transpiled directly.*

- If you start the filename with an **underscore**, Sass will *not transpile it.*
- Files named this way are called **partials** in Sass.

```
_filename;
```

SASS / SCSS – @mixin , @include

- 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.

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

mixin

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

include

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

main.scss

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

main.scss

```
.danger {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
  background-color: green;
}
```

CSS Output (main.css)

SASS / SCSS – @mixin , @include

- Mixins accept **arguments**. This way you can pass variables to a mixin.

```scss
/* 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
}
```

main.scss

```css
.myArticle {
  border: 1px solid blue;
}

.myNotes {
  border: 2px solid red;
}
```

CSS Output (main.css)

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

```scss
.myTips {
  @include bordered($color: orange);
}
```

```css
.myTips {
    border: 1px solid orange;
}
```

SASS / SCSS – @extend

- The **@extend** directive lets you share a set of CSS properties ***from one selector to another.***
- The @extend directive helps keep your Sass code very DRY.

```scss
.button-basic  {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  @extend .button-basic;
  background-color: red;
}

.button-submit  {
  @extend .button-basic;
  background-color: green;
  color: white;
}
```

main.scss

```css
.button-basic, .button-report, .button-submit {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  background-color: red;
}

.button-submit  {
  background-color: green;
  color: white;
}
```

CSS Output (main.css)

SASS / SCSS – &

- **&** can be used as shorthand for parent selectors.

```
a {
  text-decoration: none;

  &:hover {
    color: red;
  }
}
```

```
a {
  text-decoration: none;
}


a:hover {
  color: red;
}
```

```
.nav {
  &--link {
    display: inline-block;
  }
  &--title {
    font-size: 18px;
  }
}
```

```
.nav--link {
  display: inline-block;
}
.nav--title {
  font-size: 18px;
}
```