<epam>

Pre & Post Processing of CSS

October 2022

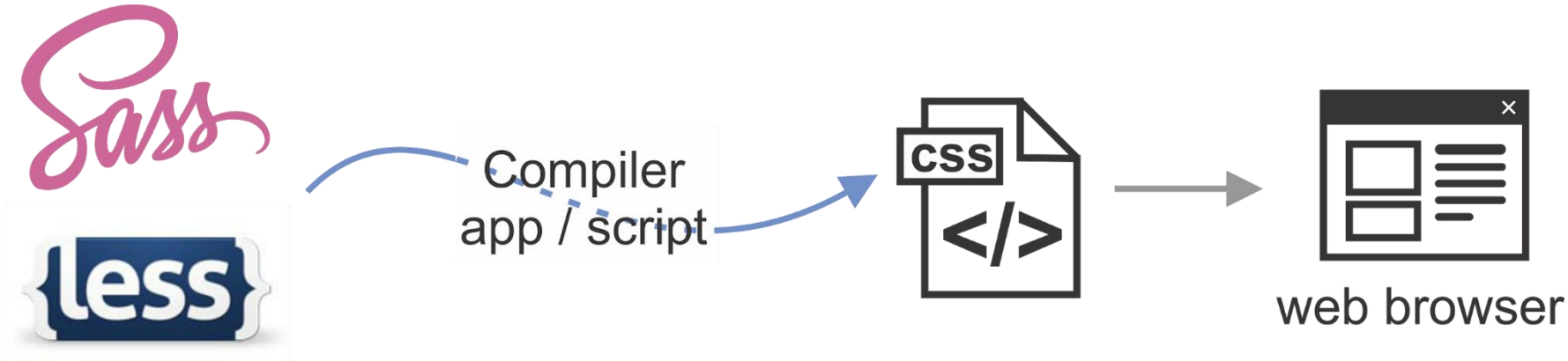
Agenda

1. INTRODUCTION
2. SASS, LESS, STYLUS
3. CSS POSTPROCESSOR
4. SUMMARY

INTRODUCTION

AN INTRO TO CSS PREPROCESSING

CSS preprocessor is basically a scripting language that extends CSS and then compiles it into regular CSS.



CSS PREPROCESSING: PROS & CONS

PROS

- Modularization for your styles.
- Reduced redundancy with variables and mixins.
- Code reuse across multiple projects.
- Nested.

CONS

- Not native syntax for browsers.
- Needs to be compiled.

SASS, LESS, STYLUS

SASS (SYNTATICALLY AWESOME STYLESHEETS)

Sass is a stylesheet language that’s compiled to CSS. It allows you to use variables, nested rules, mixins, functions, and more, all with a fully CSS-compatible syntax. Sass helps keep large stylesheets well-organized and makes it easy to share design within and across projects.

SASS and SCSS Syntax

The 'Sass' abbreviation stand for 'Syntactically Awesome Stylesheets. A more popular sass format has a different abbreviation and a file extension '.scss' which means 'Sassy CSS'. It was made to keep compatibility with CSS format.

sass

```
#main
color: blue
font-size: 0.3em

a
 font:
  weight: bold
  family: serif
 &:hover
  background-color: #eee
```

scss

```
#main {
 color: blue;
 font-size: 0.3em;

 a {
  font: {
   weight: bold;
   family: serif;
  }
  &:hover {
   background-color: #eee;
  }
 }
}
```

LESS

Less is a CSS pre-processor, meaning that it extends the CSS language, adding features that allow variables, mixins, functions and many other techniques that allow you to make CSS that is more maintainable, themable and extendable. Less runs inside Node, in the browser and inside Rhino.

STYLUS

Stylus is a dynamic stylesheet language, its design influenced by Sass and LESS. It's regarded as the third most used CSS preprocessor syntax.

SASS INSTALLATION AND USAGE

APPLICATIONS WITH SASS SUPPORT

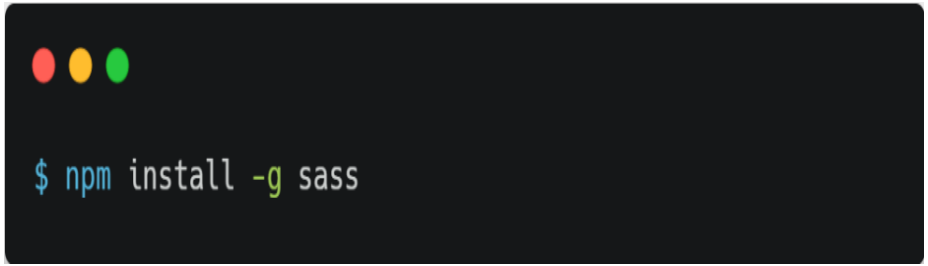
There are a good many applications that will get you up and running with Sass in a few minutes for Mac, Windows, and Linux. You can download most of the applications for free but a few of them are paid apps (and totally worth it).

- CodeKit (Paid) Mac
- Ghostlab (Paid) Mac Windows
- Hammer (Paid) Mac
- LiveReload (Paid, Open Source) Mac Windows
- Prepros (Paid) Mac Windows Linux
- Scout-App (Free, Open Source) Windows Linux Mac

SASS INSTALLATION

You can install Sass on Windows, Mac, or Linux by downloading the package for your operating system from GitHub and adding it to your PATH. That's all—there are no external dependencies and nothing else you need to install.

If you use Node.js, you can also install Sass using npm by running:



However, please note that this will install the pure JavaScript implementation of Sass, which runs somewhat slower than the other options listed here. But it has the same interface, so it'll be easy to swap in another implementation later if you need a bit more speed!

SASS USAGE


# compile input.scss to output.css
$ sass input.scss output.css
# compile all *.scss from folder and watch for changes
$ sass --watch app/scss:public/css

LESS INSTALLATION AND USAGE

LESS COMMAND LINE USAGE

```
# less install
$ npm install -g less

# compile bootstrap.less to bootstrap.css
$ lessc bootstrap.less  bootstrap.css

# compile bootstrap.less to bootstrap.css and minify
$ lessc -x bootstrap.less   bootstrap.css
```

USING LESS IN THE BROWSER

You can use less in the browser, but it recommended only for development or when you need to dynamically compile less and not on serverside. This is because less is a large JavaScript file and compiling less before the user can see the page means a delay for the user. In addition, consider that mobile devices will compile slower. For development consider if using a watcher and live reload (e.g. with grunt or gulp) would be better suited.

```
<link rel="stylesheet/less" type="text/css"
href="style.less">


<script src="less.js” type="text/javascript"></script>
```

STYLUS INSTALLATION AND USAGE

STYLUS COMMAND LINE USAGE

```
# stylus install
$ npm install -g stylus

# compile bootstrap.styl to bootstrap.css
$ stylus bootstrap.styl

# compile bootstrap.styl to bootstrap.css
$ stylus bootstrap.styl --out bootstrap.css
```

```
# compile bootstrap.styl to bootstrap.css and compress
$ stylus bootstrap.styl --out bootstrap.css --compress

# compile bootstrap.styl to bootstrap.css and watch
for changes
$ stylus --watch bootstrap.styl
```

PREPROCESSOR CAPABILITIES

PREPROCESSOR CAPABILITIES

- VARIABLES
- IMPORT
- EXTEND / INHERITANCE
- NESTING
- MIXINS
- OPERATORS

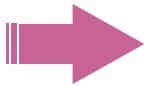
SASS VARIABLES

Sass variables are prepended with the $ symbol and the value and name are separated with a semicolon, just like a CSS property.

SCSS

```
$font-stack: Helvetica, sans-serif;
$primary-color:  #333;

body {
   font: 100% $font-stack;
   color: $primary-color;
}
```



CSS

```
body {
   font: 100% Helvetica, sans-serif;
   color: #333;
}
```

LESS VARIABLES

LESS variables are exactly the same as Sass variables, except the variable names are prepended with the @ symbol.

LESS

```
@font-stack: Helvetica, sans-serif;
@primary-color: #333;

body {
font: 100% @font-stack;
color: @primary-color;
}
```

CSS

```
body {
    font: 100% Helvetica, sans-serif;
    color: #333;
}
```

STYLUS VARIABLES

Stylus variables don't require anything to be prepended to them, although it allows the $ symbol. As always, the ending semicolon is not required, but an equal sign in between the value and variable is.

STYLUS

```
font-stack = Helvetica, sans-serif
primary-color = #333

body
    font 100% font-stack
    color primary-color
```

CSS

```
body {
    font: 100% Helvetica, sans-serif;
    color: #333;
}
```

OPERATORS

SCSS / LESS / STYL

```
.container { width: 100%; }

article[role="main"] {
   float: left;
   width: 600px / 960px * 100%;
}

aside[role="complimentary"] {
   float: right;
   width: 300px / 960px * 100%;
}
```

CSS

```
.container {
   width: 100%;
}

article[role="main"] {
   float: left;
   width: 62.5%;
}

aside[role="complimentary"] {
   float: right;
   width: 31.25%;
}
```

SASS/LESS NESTING

SCSS / LESS

```
nav {
ul {
      margin: 0;
      padding: 0;
      list-style: none;
   }

   li { display: inline-block; }  a {
      display: block;  padding: 6px 12px;
      text-decoration: none;
   }
}
```

CSS

```
nav ul {
      margin: 0;
      padding: 0;
      list-style: none;
}

nav li {
   display: inline-block;
}

nav a {
      display: block;
      padding: 6px 12px;
      text-decoration: none;
}
```

STYLUS NESTING

STYL

```
nav

  ul
    margin 0
    padding 0
    list-style none

  li
    display inline-block

  a
    display block
    padding 6px 12px
    text-decoration none
```

CSS

```
nav ul {
    margin: 0;
    padding: 0;
    list-style: none;
}

nav li {
  display: inline-block;
}

nav a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
}
```

REFERENCING PARENT SELECTORS

The & symbol references the parent selector.

LESS / SASS / STYL

```
a {
  color: #000;

  &:hover {
    color: #ccc;
  }
}
```

CSS

```
a {
  color: #000;
}

a:hover {
  color:
  #ccc;
}
```

IMMEDIATE CHILDREN SELECTOR

We can write the children selectors inside the parent's brackets. All three preprocessors have the same syntax for nesting selectors.

LESS / SASS / STYL

```
a {
   color: #000;

   > span {
      color: #ccc;
   }
}
```

CSS

```
a {
   color: #000;
}

a > span {
   color: #ccc;
}
```

IMPORT

SCSS / LESS / STYL

```
// reset.scss

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

SCSS / LESS / STYL

```
/* base.scss */

@import 'reset';

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

CSS

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

body {
  font: 100% Helvetica, sans-serif;
  background-color: #efefef;
}
```

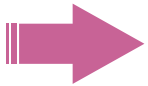
SASS MIXINS

Mixins are functions that allow the reuse of properties throughout our stylesheet. Rather than having to go throughout our stylesheet and change a property multiple times, we can now just change it inside our mixin.

SCSS

```scss
@mixin border-radius($radius: 5px) {
    -webkit-border-radius: $radius;
       -moz-border-radius: $radius;
        -ms-border-radius: $radius;
            border-radius: $radius;
}

.box {
   @include border-radius(10px);
}
```



CSS

```css
.box {
   -webkit-border-radius: 10px;
   -moz-border-radius: 10px;
   -ms-border-radius: 10px;
   border-radius: 10px;
}
```

LESS MIXINS

This can be really useful for specific styling of elements and vendor prefixes. When mixins are called from within a CSS selector, the mixin arguments are recognized and the styles inside the mixin are applied to the selector.

LESS

```
.border-radius(@radius: 5px) {
    -webkit-border-radius: @radius;
       -moz-border-radius: @radius;
        -ms-border-radius: @radius;
            border-radius: @radius;
}

.box { .border-radius(10px); }
```

CSS

```
.box {
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    -ms-border-radius: 10px;
    border-radius: 10px;
}
```

STYLUS MIXINS

STYL

```
.border-radius(radius = 5px) {
    -webkit-border-radius: radius;
       -moz-border-radius: radius;
        -ms-border-radius: radius;
            border-radius: radius;
}

.box { border-radius(10px); }
```

CSS

```
.box {
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    -ms-border-radius: 10px;
    border-radius: 10px;
}
```

SASS/STYLUS EXTEND / INHERITANCE

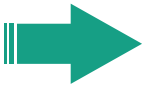
SCSS / STYL

```
.message {
  border: 1px solid #ccc;
  padding: 10px; color: #333;
}

.success {
  @extend   .message;
   border-color: green;
}

.error {
  @extend .message;
  border-color: red;
}

.warning {
  @extend .message;
  border-color: yellow;
}
```



CSS

```
.message, .success, .error, .warning {
    border: 1px solid #cccccc;
    padding: 10px; color: #333;
}

.success {
    border-color: green;
}

.error {
    border-color: red;
}

.warning {
    border-color: yellow;
}
```

SASS PLACEHOLDERS

SCSS

```
%message {
 border: 1px solid #ccc;
 padding: 10px; color: #333;
}

.success {
 @extend %message;
  border-color: green;
}

.error {
 @extend %message;
  border-color: red;
}

.warning {
 @extend %message;
  border-color: yellow;
}
```

CSS

```
.success, .error, .warning {
    border: 1px solid #cccccc;
    padding: 10px; color: #333;
}

.success {
    border-color: green;
}

.error {
    border-color: red;
}

.warning {
    border-color: yellow;
}
```

LESS EXTEND / INHERITANCE

LESS

```
.nav ul {
  &:extend(.inline);
  background: blue;
}

.inline {
  color: red;
}
```

CSS

```
.nav ul {
  background: blue;
}

.inline,
nav ul {
  color: red;
}
```

LOOPS

SCSS

```
@each $var in a, b, c, d {
   .#{$var}{
      background-image: url('#{$var}.png');
 }
}
```

CSS

```
a.  {
  background-image: url('a.png');
}

b.  {
  background-image: url('b.png');
}

c.  {
  background-image: url('c.png');
}

d.  {
  background-image: url('d.png');
}
```

LESS FUNCTIONS

LESS provides several functions with quite descriptive keywords that we can use to manipulate
colors: lighten(), darken(), saturate(), desaturate(), fadein(), fadeout(), fade(), spin() and mix().

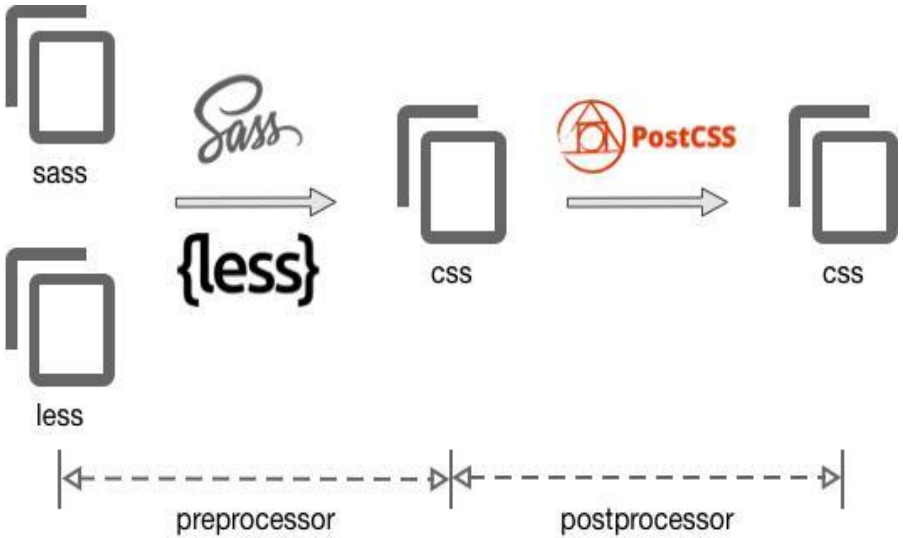
```
// Increase the lightness of a color in the HSL color space by an absolute amount.
@lighten(@color, x%)

// Decrease the lightness of a color in the HSL color space by an absolute amount.
@darken(@color, x%)
```

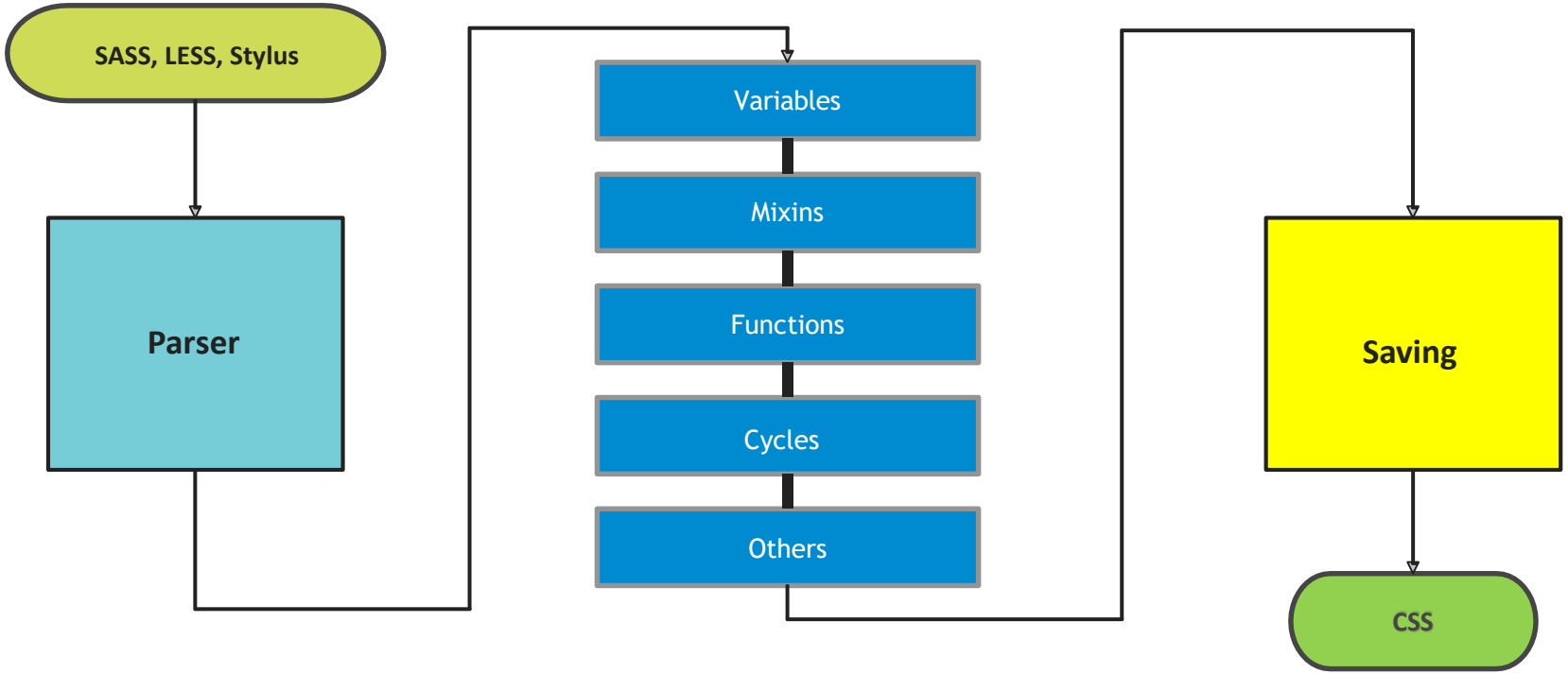
CSS POSTPROCESSORS

CSS POST-PROCESSOR

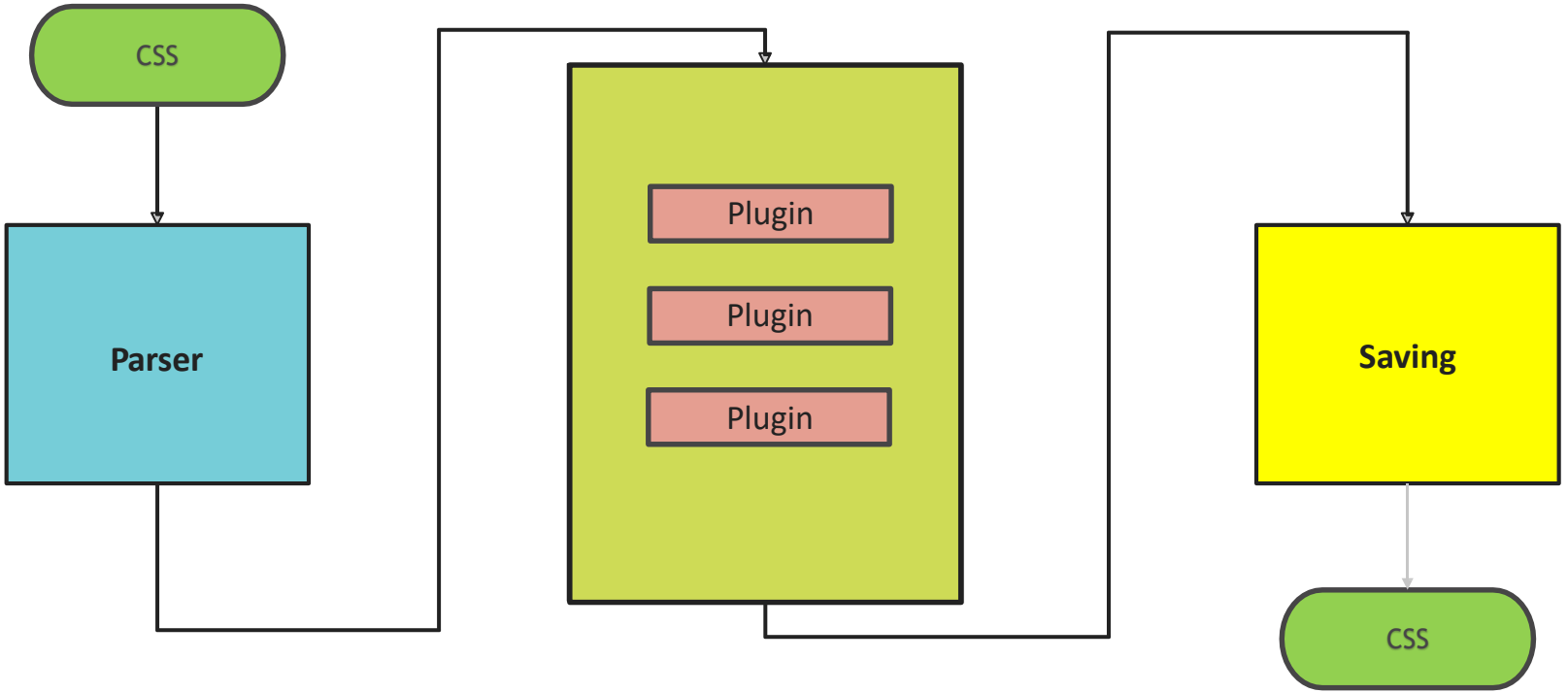
Post-processor applies changes to a CSS file after it's been hand-coded or generated by a pre-processor. We used post-processors to tweak it and improve it. To get more out of our CSS than we could do by ourselves.



HOW DOES CSS PREPROCESSOR WORK



HOW DOES CSS POSTPROCESSOR WORK



POSTPROCESSORS

- CSSO - CSS minifier with structural optimizations.
- CSScomb - coding style formatter.
- Autoprefixer - add vendor prefixes to CSS rules.
- PostCSS - tool that uses JavaScript-based plugins to automate routine CSS operation.

USEFUL LINKS

- Sass official documentation - http://sass-lang.com/documentation
- Sass tutorials - http://thesassway.com/
- Sass cheatsheet - https://devhints.io/sass
- PostCSS - https://postcss.org/

THANK YOU!