Pre & Post Processing CSS

UA Resource Development Unit

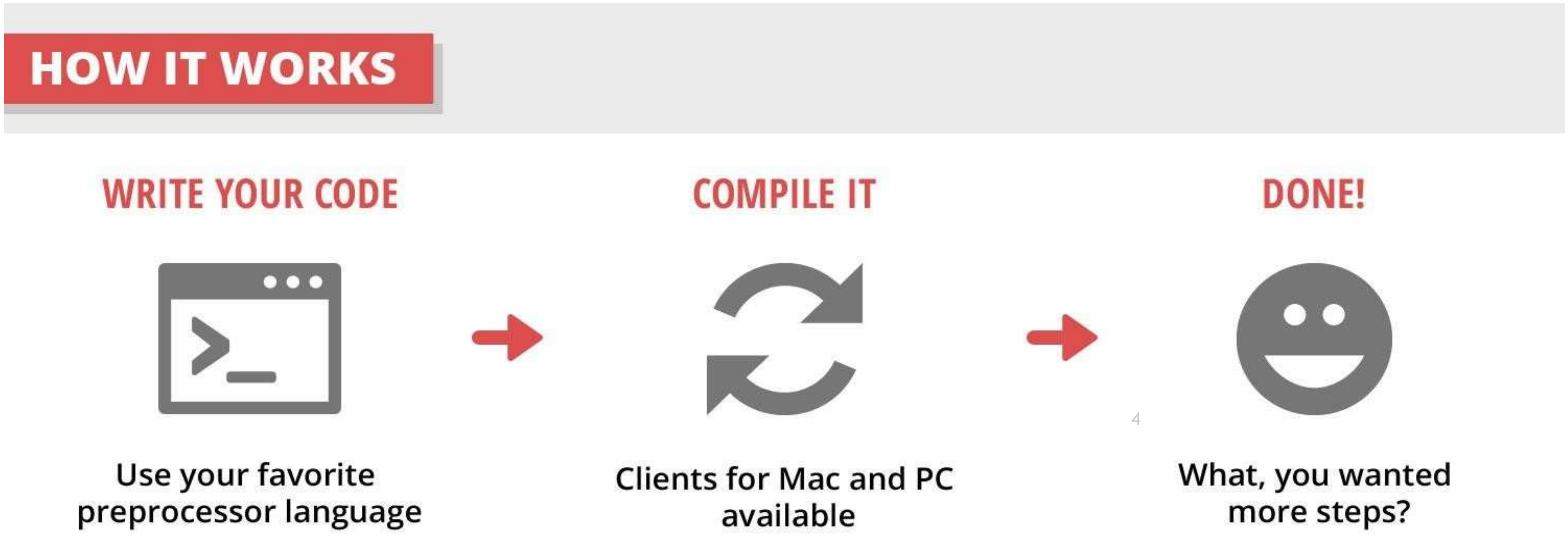
2020

AGENDA

CSS PREPROCESSORS
SASS, LESS, STYLUS

AN INTRO TO CSS PREPROCESSING

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


HOW IT WORKS
WRITE YOUR CODE
COMPILE IT
DONE!
Use your favorite preprocessor language
Clients for Mac and PC available
What, you wanted more steps?

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)

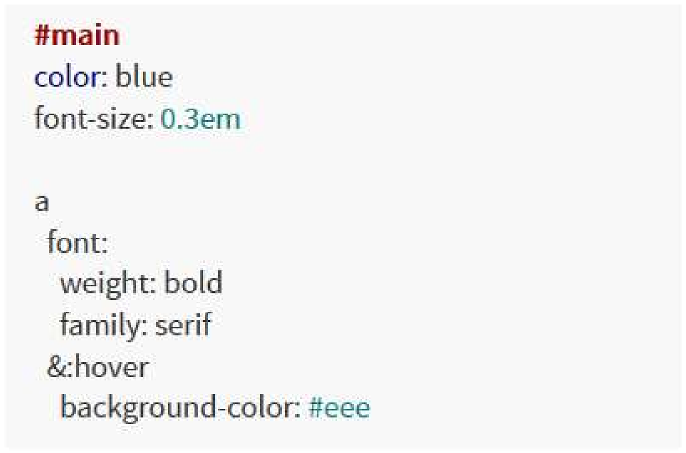
Sass (Syntactically Awesome Stylesheets) is a stylesheet language initially designed by Hampton Catlin and developed by Natalie Weizenbaum. After its initial versions, Weizenbaum and Chris Eppstein have continued to extend Sass with SassScript, a simple scripting language used in Sass files.

http://sass-lang.com/

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.

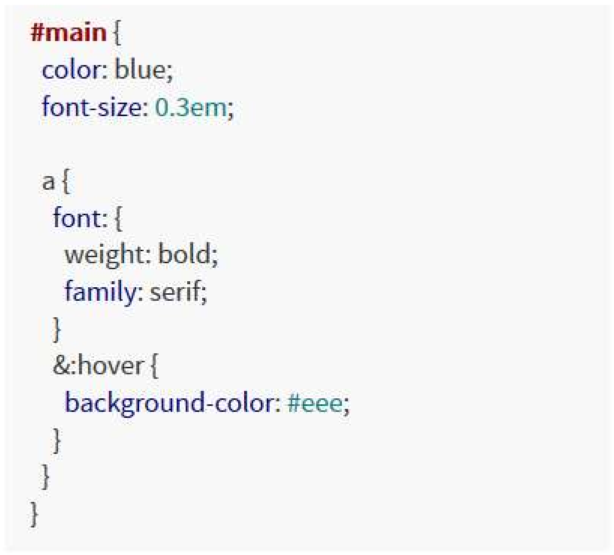
sass



```
#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

File extension:
.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.

http://lesscss.org/

STYLUS

stylus

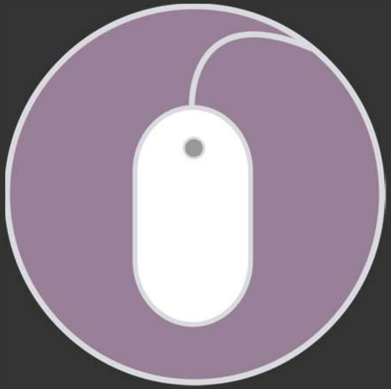
File extension:
.styl

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

https://stylus-lang.com/

INSTALLATION AND USAGE

APPLICATIONS WITH SASS SUPPORT



CodeKit (Paid)
Compass.app (Paid, Open Source)
Hammer (Paid)
Koala (Open Source)

LiveReload (Paid, Open Source)
Mixture (Free)
Prepros (Paid)
Scout (Open Source)

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).

WHAT IS NPM?

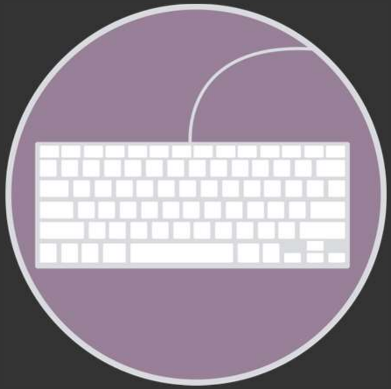
Installation

# From official site (All platform)
https://nodejs.org

npm (Node Package Manager) - is the package manager for JavaScript and the world’s largest software registry.

Use npm to install, share, and distribute code; manage dependencies in your projects; and share & receive feedback with others.

SASS INSTALLATION



```
$ npm install sass -g

# 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
```

USING LESS IN THE BROWSER

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

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

You can use less in the browser but recommended only for development or when you need to dynamically compile less and cannot do it 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.

LESS COMMAND LINE USAGE

```
$ npm install less -g

# 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
```

STYLUS COMMAND LINE USAGE

```
$ npm install stylus -g

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

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

Stylus is an open source project hosted on GitHub. You can install from source or you can simply use NPM

STYLUS COMMAND LINE USAGE

```
# 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
```

Stylus is an open source project hosted on GitHub. You can install from source or you can simply use NPM

USE GULP OR GRUNT

Its all about automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes.

MECHANISMS

THE MOST COMMON MECHANISMS

$ Variables

--- >| Nesting

@ Import

() Mixins

@ Extend / Inheritance

$ Operators

SASS VARIABLES

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;
}
```

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

LESS VARIABLES

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;
}
```

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

STYLUS VARIABLES

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;
}
```

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.

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

LESS / SASS / STYL

```
a {
   color: #000;

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

CSS

```
a {
   color: #000;
}

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

The & symbol references the parent selector.

IMMEDIATE CHILDREN SELECTOR

LESS / SASS / STYL

```
a {
   color: #000;

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

CSS

```
a {
   color: #000;
}

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

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

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

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

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

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.

LESS MIXINS

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;
}
```

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.

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

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

.inline {
   color: red;
}
```

CSS

```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().

LESS FUNCTIONS

```
@lighten(@color, x%)
```

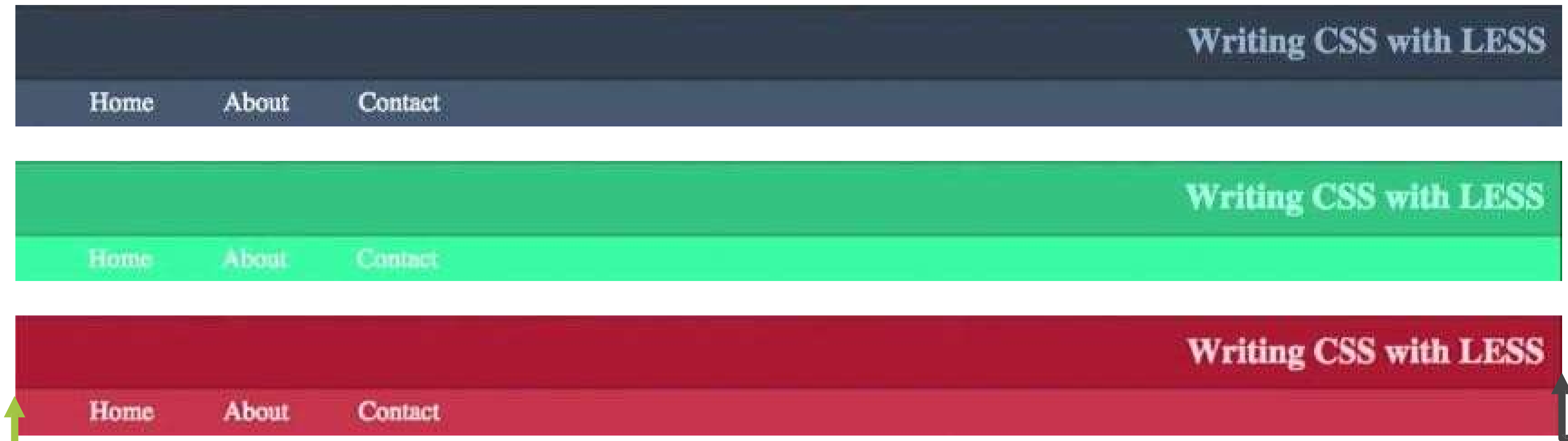
Increase the lightness of a color in the HSL color space by an absolute amount.

LESS FUNCTIONS

@darken(@color, x%)

Decrease the lightness of a color in the HSL color space by an absolute amount.

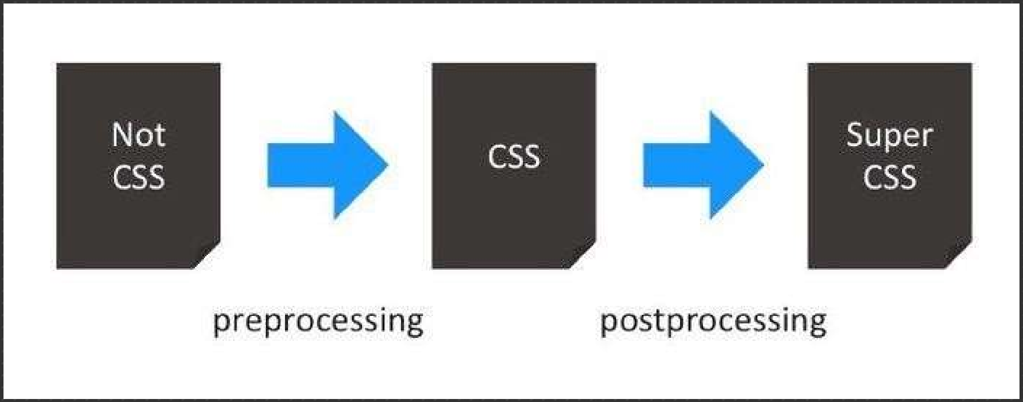
LESS FUNCTIONS



@lighten(@color, x%)

@darken(@color, x%)

CSS POSTPROCESSORS



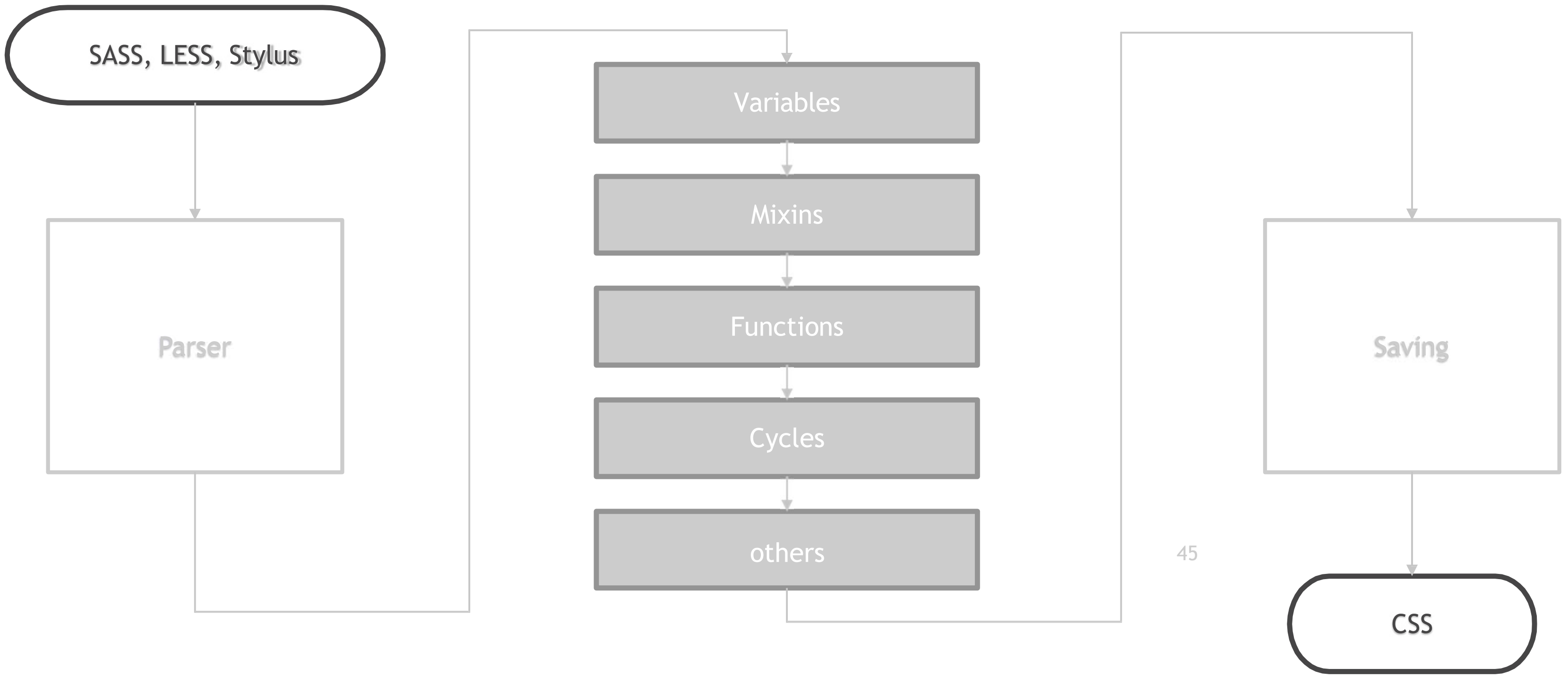
CSS Post-Processor

General meaning

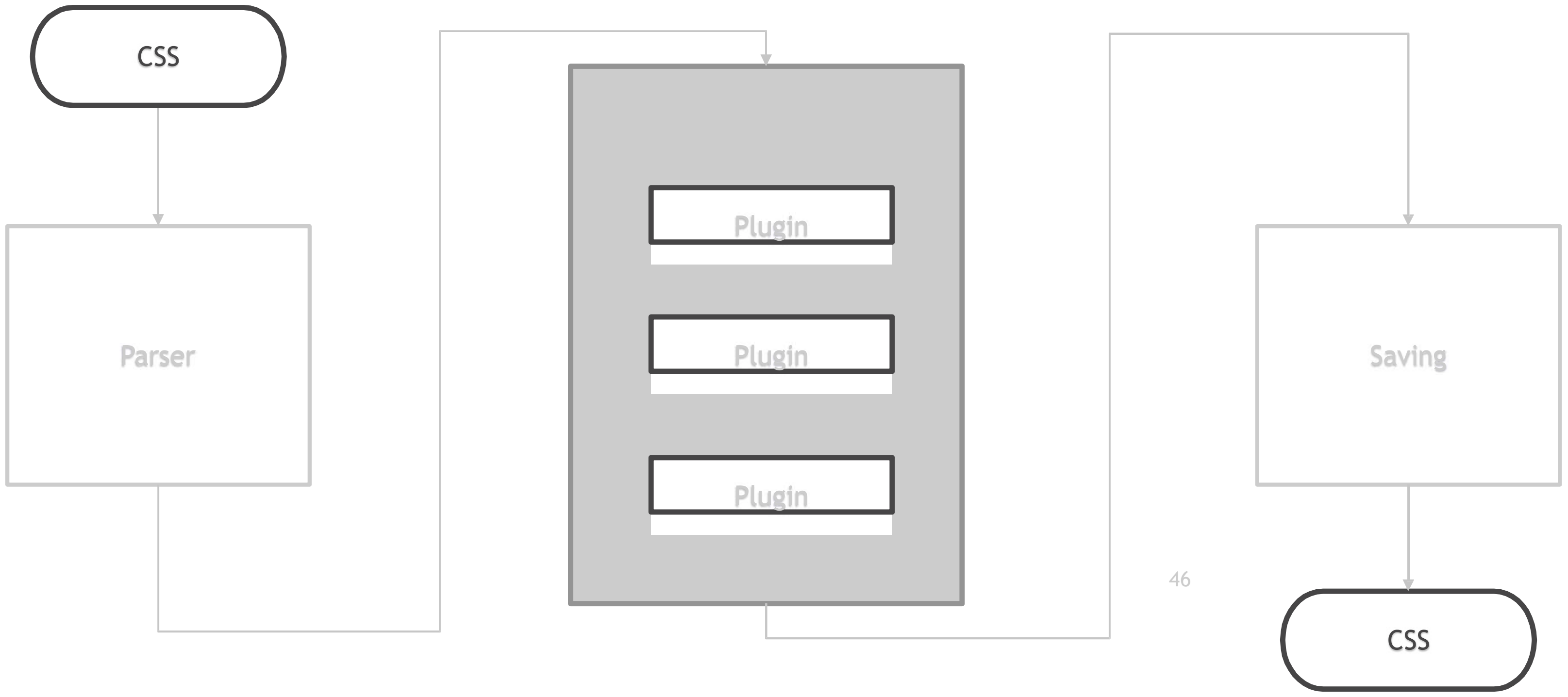
A post-processor applies changes to a CSS file after it's been hand-coded or generated by a pre-processor. We used post-processors to tweak it and improve it. To get more out of our CSS than we could do by ourselves.

DIFFERENCES

HOW CSS PREPROCESSORS WORK



HOW CSS POSTPROCESSORS WORK



EXAMPLES

1. CSSO - CSS minifier with structural optimizations

2. CSScomb - coding style formatter

3. Autoprefixer - add vendor prefixes to CSS rules

4. PostCSS

References

Related resources

- Sass official documentation http://sass-lang.com/documentation
- Sass tutorials http://thesassway.com/
- Sass cheatsheet https://devhints.io/sass
- https://www.hongkiat.com/blog/css-post-processors-tips-resources/
- https://medium.com/@ddprrt/deconfusing-pre-and-post-processing-d68e3bd078a3
- https://habr.com/ru/post/434098/

Q&A

DRIVEN CANDID CREATIVE ORIGINAL INTELLIGENT EXPERT

UA Frontend Online LAB