Getting Sassy with CSS

Julie Cameron
@jewlofthelotus

self.conf 2014

About Me

@jewlofthelotus

articulate®

Girl Develop It

don't be shy. develop it.

About Me

@jewlofthelotus

articulate®

Girl Develop It

don't be shy. develop it.

Syntactically **A**wesome **S**tyle**s**heets

Sass

NOT an acronym...

Syntactically **A**wesome **S**tyle**s**heets

PRO TIP

NOT an acronym...

Syntactically **A**wesome **S**tyle**s**heets

"If you see something, say something.
Only you can prevent overcapitalization."

- Hampton Catlin, Creator Of Sass

"If you see something, say something.
Only you can prevent overcapitalization."

- Hampton Catlin, Creator Of Sass

… but don’t be a

#Sasshole

What Is Sass?

What Is Sass?

1. An **extension** of CSS

SassScript Language. Fully CSS-compatible syntax.

What Is Sass?

1. An **extension** of CSS

SassScript Language. Fully CSS-compatible syntax.

2. A CSS **preprocessor**

SassScript → MAGIC → CSS

Where Did Sass Come From?

- 2007: Designed by **Hampton Catlin** (Haml)
- Developed by **Chris Eppstein** (Compass) and **Nathan Weizenbaum**

Why Did They Make Sass?

Why Did They Make Sass?

Because vanilla **CSS** just **wasn't cutting it any more.**

> **CSS stops short** of even more powerful features [...]. That is because these things give power-users a lot of rope, but less experienced users will unwittingly hang themselves; or, more likely, be so scared that they won't even touch CSS.

– Bert Bos, CSS Co-Inventor

What Does Sass Give Us That CSS Doesn't?

1. Reduced Repetition
2. Cleaner Modularity
3. Maintainability
4. REAL PROGRAMMING!

Moving The Web Forward

Setup & Workflow

Sass Installation

Ruby

OS X? Lucky you, it's preinstalled!

Windows? Try RubyInstaller.

Sass

```
$ gem install sass
```

Running Sass

Sass → CSS

```
$ sass source.sass output.css
```

```
$ sass source_dir output_dir
```

Running Sass

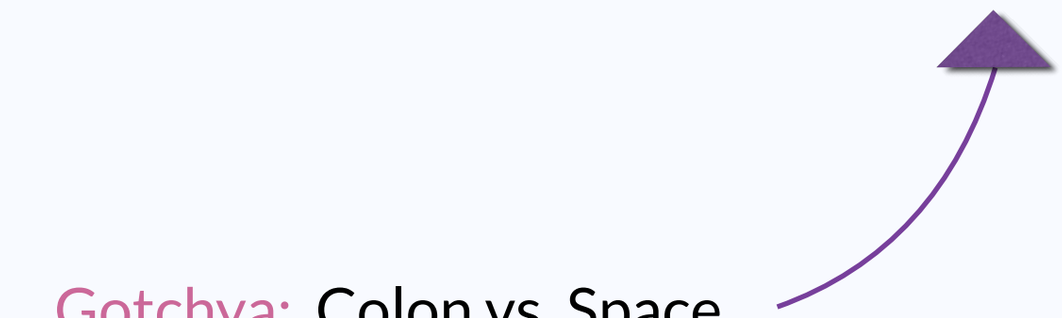

```
$ sass source.sass output.css
```

```
$ sass source_dir output_dir
```

Sass Is Watching You

This is the best thing. Ever.

```
$ sass --watch source.sass:output.css
```



Two Sass Syntaxes

Indented .sass

Original syntax; Haml-esque

```
.this
  color: #000
  background: #fff

.that
  color: #fff
  background: #000
```

Sassy CSS .scss

Default; Valid CSS == Valid SCSS

```
.this {
  color: #000;
  background: #fff;
}

.that {
  color: #fff;
  background: #000;
}
```

The Good Stuff

Or, Why You'll Never Write Plain Old CSS Again.

Nesting

Nesting

HTML has a clear hierarchy - elements are ***nested***.

We can apply the same concept in Sass.

HTML

```
<nav class="navigation">
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#about">About</a></li>
    <li><a href="#contact">Contact</a></li>
  </ul>
</nav>
```

Sass

```
.navigation {
  float: right;

  li {
    display: inline-block;
    list-style-type: none;
    margin-left: 1.5em;
  }

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

Nesting Output

Sass

```
.navigation {
  float: right;

  li {
    display: inline-block;
    list-style-type: none;
    margin-left: 1.5em;
  }

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

MAGIC

CSS

```
.navigation {
  float: right;
}

.navigation li {
  display: inline-block;
  list-style-type: none;
  margin-left: 1.5em;
}

.navigation a {
  display: block;
  text-decoration: none;
}
```

Nesting Gotchya

Mirroring HTML nesting is SUPER easy.

Sass

```
body {
  .header {
    .navigation {
      ul {
        li {
          a {
            // ...
          }
        }
      }
    }
  }
}
```

MAGIC?

CSS

```
body .header .navigation ul li a {
  /* ... */
}
```

Nesting Gotchya

Mirroring HTML nesting is SUPER easy.

Overly verbose and overly specific.

Sass

```
body {
  .header {
    .navigation {
      ul {
        li {
          a {
            // ...
          }
        }
      }
    }
  }
}
```

MAGIC?

CSS

```
body .header .navigation ul li a {
  /* ... */
}
```

Nesting Gotchya

Mirroring HTML nesting is SUPER easy.

Overly verbose and overly specific.

Rule of thumb:
No more than 3 levels deep

Sass

```
body {
  .header {
    .navigation {
      ul {
        li {
          a {
            // ...
          }
        }
      }
    }
  }
}
```

MAGIC?

CSS

```
body .header .navigation ul li a {
  /* ... */
}
```

Nesting The Parent Selector

You can reference **parent selector(s)** with an **&**

Sass

```
a {
  color: #beedee;

  &:hover {
    color: #cbbebb;
  }

  &.btn {
    background: #deede6;
  }

  .btn {
    display: block;
  }
}
```

MAGIC

CSS

```
a {
  color: #beedee;
}

a:hover {
  color: #cbbebb;
}

a.btn {
  background: #deede6;
}

a .btn {
  display: block;
}
```

Nesting The Parent Selector

The & can follow other selectors, too!

Sass

```
a {
  .footer & {
    text-decoration: none;

    span {
      opacity: .5;
    }
  }

  span {
    .navigation & {
      display: block;
    }
  }
}
```

CSS

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

.footer a span {
  opacity: .5;
}

.navigation a span {
  display: block;
}
```

Nesting The Parent Selector

With Sass 3.3, you can also **append a suffix** to the **&**!

Sass

```
.button {
  &--primary { background: orange; }
  &--secondary { background: blue; }
}
```

CSS

```
.button--primary { background: orange; }
.button--secondary { background: blue; }
```

Nesting Media Queries

THIS IS FREAKING AWESOME.

Sass

```
.sidebar {
  width: 35%;

  @media (max-width: 800px) {
    width: 25%;
  }

  @media (max-width: 500px) {
    width: 100%;
  }
}
```

MAGIC

CSS

```
.sidebar {
  width: 35%;
}

@media (max-width: 800px) {
  .sidebar {
    width: 25%;
  }
}

@media (max-width: 500px) {
  .sidebar {
    width: 100%;
  }
}
```

Imports & Organization

Imports

CSS `@import` has always meant an extra file download.

Sass modifies `@import` to instead include the resource during compilation, rather than on the client side.

OMG YAY MODULARITY!!!

Sass

```
@import "vars";

@import "compass";
@import "fontawesome";

@import "utilities";

@import "grid";
@import "base";

@import "modules/all";
@import "pages/all";
```

Imports File Structure

- project_awesome
  - sass
    - lib
      - compass.scss
      - fontawesome.scss
    - base.scss
    - grid.scss
    - application.scss
  - css
    - application.css

Sass

```
// application.scss
@import "lib/compass";
@import "lib/fontawesome";
@import "grid";
@import "base";
```

```
$ sass sass/application.scss css/application.css
```

Imports File Structure

```
$ sass sass/ css/
```

- project_awesome
  - sass
    - lib
      - compass.scss
      - fontawesome.scss
    - base.scss
    - grid.scss
    - application.scss

- project_awesome
  - css
    - lib
      - compass.css
      - fontawesome.css
    - base.css
    - grid.css
    - application.css

Import Partials

```
$ sass sass/ css/
```

- project_awesome
  - sass
    - lib
      - _compass.scss
      - _fontawesome.scss
    - _base.scss
    - _grid.scss
    - application.scss
    - print.css

- project_awesome
  - sass
    - …
  - css
    - application.css
    - print.css

Variables

Variables

```
.header {
  background-color: #531946;
}
.header a { color: #fff; }
.header a:hover { color: #095169; }

.footer {
  background-color: #30162B;
  color: #fff;
}
.footer a { color: #095169; }
.footer a:hover { color: #fff; }

.feature a {
  background-color: #30162B;
  color: #fff;
}
.feature a:hover { color: #531946; }
```

WTF is #531946?

#30162b?
#095179?
WAT?

Variables

Sass

```sass
$white: #ffffff;
$eggplant: #531946;
$eggplantDark: #30162B;
$teal: #095169;

.header {
  background-color: $eggplant;
}
.header a { color: $white; }
.header a:hover { color: $teal; }

.footer {
  background-color: $eggplantDark;
  color: $white;
}
.footer a { color: $teal; }
.footer a:hover { color: $white; }
```

Variables let you
reuse single values.

Variable Uses

colors

font sizes

font families

font paths

padding

margins

breakpoints

border-radius

pseudo content

shadows

gradients

SELECTORS!

LOGIC!

ALL THE THINGS!!!1!

Variable Data Types

Numbers

Strings

Colors

Lists

Booleans

Null

Maps

Sass

```sass
$base-padding: 10px;
$line-height: 1.5;

$base-font: Verdana;
$content: "Loading...";

$feature-color: purple;
$feature-background: rgba(0, 255, 0, 0.5);

$base-margin: 20px 0 30px 10px;
$base-font: "Trebuchet MS", "Verdana", sans-serif;

$bordered: true;
$shadowed: false;

$secondary: null;

$map: (a: 1px, b: 2px, c: 3px);
```

Variable Dynamics

- **Mutable.**
- Can have **default values**.
- Can be **interpolated**.
- Have **scopes**.

Sass

```sass
$side: left;                // $side == left
$side: top;                 // $side == top
$side: right !default; // $side == top

.box-#{$side} {
  $box-border: 1px solid #ccc;

  border-#{$side}: $box-border;

  &:before {
    content: "It's on the #{$side} side";
  }
}

.box-alt {
  border-#{$side}: $box-border;
  // ERROR!! Undefined variable "$box-border"
}
```

Math & Color

Math

Sass lets us use basic math operations.

- `+` addition
- `-` subtraction
- `*` multiplication
- `/` division
- `%` modulus

Sass

```sass
$title: "Self.Conference";
$base-font-size: 16px;
$padding-default: 2em;

.header {
  padding: $padding-default / 2;
  font-size: $base-font-size * 2;

  &:before {
    content: "Welcome to " + $title;
  }
}
```

Math Functions

Sass comes with a set of math functions.

- `abs($num)` - absolute value
- `ceil($num)` - round up to closest whole number
- `floor($num)` - round down to closest whole number
- `percentage($num)` - convert to percentage
- `round($num)` - round to closest whole number
- `max($list)` - maximum list value
- `min($list)` - minimum list value

Color Math

We can even use math to manipulate colors.

Sass

```
.addition {
  color: #555555 + #112233; // => #667788
}
.subtraction {
  color: #555555 - #112233; // => #443322
}
.multiplication {
  color: #555555 * 2;       // => #aaaaaa
}
.division {
  color: (#555555 / 2);     // => #2a2a2a
}
```

Color Functions

Sass comes with **AWESOME** color functions.

- `rgba($hex, $alpha)`
- `lighten($color, $percent)`
- `darken($color, $percent)`
- `saturate($color, $percent)`
- `desaturate($color, $percent)`
- `mix($color1, $color2)`
- `grayscale($color)`
- `invert($color)`
- `complement($color)`
- AND MANY MORE!

But wait, there's more!

Mixins

Mixins

Variables let you **reuse** single values

Mixins let you reuse **blocks of styles**

Mixins Use

CSS

```css
.header {
  border-radius: 5px;
  padding: 5px 20px;
}

.footer {
  border-radius: 5px;
  padding: 5px 20px;
}

.feature a {
  border-radius: 5px;
  padding: 5px 20px;
}
```

Mixins Use

CSS

```css
.header {
  border-radius: 5px;
  padding: 5px 20px;
}

.footer {
  border-radius: 5px;
  padding: 5px 20px;
}

.feature a {
  border-radius: 5px;
  padding: 5px 20px;
}
```

Sass

```sass
@mixin rounded-box {
  border-radius: 5px;
  padding: 5px 20px;
}

.header {
  @include rounded-box;
}

.footer {
  @include rounded-box;
}

.feature a {
  @include rounded-box;
}
```

Mixin Output

Sass

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

.header { @include rounded-corners; }
.footer { @include rounded-corners(10px); }
```

CSS

```
.header {
  -webkit-border-radius: 5px;
     -moz-border-radius: 5px;
          border-radius: 5px;
}

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

Mixin Features

- **multiple** arguments
- argument **defaults**
- **optional** arguments
- arguments == **$variables scoped to the mixin**

Mixin Alternative?

CSS

```css
.header {
  border-radius: 5px;
  padding: 5px 20px;
}

.footer {
  border-radius: 5px;
  padding: 5px 20px;
}

.feature a {
  border-radius: 5px;
  padding: 5px 20px;
}
```

Mixin Alternative?

CSS

```
.header {
  border-radius: 5px;
  padding: 5px 20px;
}

.footer {
  border-radius: 5px;
  padding: 5px 20px;
}

.feature a {
  border-radius: 5px;
  padding: 5px 20px;
}
```

Better CSS

```
.header,
.footer,
.feature a {
  border-radius: 5px;
  padding: 5px 20px;
}
```

@extend

@extend

Lets you group **selectors with blocks of styles**

Sass

```
.message {
  border: 2px solid;
  margin-bottom: 1em;
  padding: 1em;
  text-align: center;
}

.message-alert {
  @extend .message;
  @include msg-colors($yellow);
}

.message-error {
  @extend .message;
  @include msg-colors($red);
}
```

MAGIC

CSS

```
.message,
.message-alert,
.message-error,
.message-notice {
  border: 2px solid;
  margin-bottom: 1em;
  padding: 1em;
  text-align: center;
}

.message-alert {
  border-color: goldenrod;
  color: goldenrod;
}

.message-error {
  border-color: darkred;
  color: darkred;
}
```

Placeholders

Placeholders can be `@extend`ed but **will never compile to the CSS on their own**

Sass

```
%message {
  border: 2px solid;
  margin-bottom: 1em;
  padding: 1em;
  text-align: center;
}

.message-alert {
  @extend %message;
  @include msg-colors($yellow);
}

.message-error {
  @extend %message;
  @include msg-colors($red);
}
```

MAGIC

CSS

```
.message-alert,
.message-error,
.message-notice {
  border: 2px solid;
  margin-bottom: 1em;
  padding: 1em;
  text-align: center;
}

.message-alert {
  border-color: goldenrod;
  color: goldenrod;
}

.message-error {
  border-color: darkred;
  color: darkred;
}
```

Plus Lots More WIN

Functions

```
@function fluidize($target, $context) {
  @return ($target / $context) * 100%;
}

.sidebar {
  width: fluidize(350px, 1000px);
  // => width: 35%;
}
```

Conditions

```
$theme: light !default;

body {
  @if $theme == dark {
    background: #000;

  } @else if $theme == mid {
    background: #afafaf;

  } @else {
    background: #fff;
  }
}
```

Each Loops

```
$themes: dark dark-mid light-mid light;

@each $theme in $themes {
   .theme-#{$theme} {
      @include theme-colors($theme);
   }
}
```

For Loops

```
@for $i from 1 through 12 {
  .grid-col-#{$i} {
    width: $grid-col-width * $i;
  }
}
```

While Loops

```
$i: 1;

@while $i <= 12 {
  .grid-col-#{$i} {
    width: $grid-col-width * $i;
  }

  $i: $i + 1;
}
```

Demo Time

https://github.com/roytomeij/sassconf


Sass


Sass

Resources

Sass-Lang.com

The Sass Way

Sass For Web Designers

Assembling Sass

SassNews Newsletter

SassConf

SassConf 2013 Videos

SassMeister

Sass Style Guide

#teamSass

Closing Tips

- Take it **one step at a time.**
- Check your **output** regularly.
- Mind your `@mixins` and `@media` queries.
- Don't `@extend` *too* much.
- **Modularize** all the things.

Questions?

Slides: http://bit.ly/self_sass

@jewlofthelotus

Girl Develop It Ann Arbor

Launch Party w/ Pillar @ The Forge

June 4th 6:30pm

RSVP @ bit.ly/gdiaa

Questions?

Slides: http://bit.ly/self_sass

@jewlofthelotus

Girl Develop It Ann Arbor

Launch Party w/ Pillar @ The Forge

June 4th 6:30pm

RSVP @ bit.ly/gdiaa