

Now Playing Tracks

[+ Follow bguiz](#)

Brendan Graetz

SASS in 5 minutes

[SASS - syntactically awesome stylesheets](#) - is a meta-language that compiles to [CSS](#), that has gained much popularity of late. Having learnt it recently, I thought to put together some material that would cover the breadth of the topic so as to codify my own knowledge of it. After all, ["while we teach, we learn."](#)

This is a brief overview, and assumes prior knowledge of CSS - don't expect a deep dive!

SCSS vs SASS

Valid CSS is valid SCSS, but invalid SASS.

To get SASS from SCSS, you simply discard curly braces and semi-colons (`{ } ;`), relying on whitespace indentation instead, the same way blocks are implicit in python/ coffeescript. Another difference is that for mixins, instead of using `@include`, and `@mixin`, you use `+` and `=` instead.

I have chosen to write all the examples below in SASS over SCSS for the sake of brevity, and you can easily obtain SCSS by applying the reverse of the instructions above.

Import

```
//app.sass

@import "nav.sass"
@import "_heading.sass"
```

If a SASS file's name is prefixed with an underscore `_`, it is considered a partial, and will not be compiled into a CSS file. In this example, we will have an `app.css` and a `nav.css`, but not have a `heading.css`.

Nest

```
//app.sass

.a
  width: 100%
  .b
    color: #fa0
  &.c
    color: #0e7
  .d &
```

```
    color: #333

//app.css

.a {
  width: 100%;
}
.a .b {
  color: #fa0;
}
.a.c {
  color: #0e7;
}
.d .a {
  color: #333
}
```

When nesting selectors, the nested selector has an implicit parent selector, and a space prefixed. If you wish to get rid of the space, or for the parent selector to appear after the nested selected, as with the cases of `.c` and `.d` respectively, you have to explicitly specify the parent selector using the `&` symbol.

SASS makes it easy to do nested selectors, but remember that its output is still CSS. Stick to a maximum of 3 levels of nested selectors.

```
//app.sass

.a
  font
    family: Arial, sans-serif
    size: 18pt

//app.css

.a {
  font-family: Arial, sans-serif;
  font-size: 18pt;
}
```

Properties can also be nested, where they have matching namespaces - i.e. hyphen (-) separated property names.

Variables

```
//app.sass

@import "_sety.sass"
$x: 1.5em
$y: bottom !default
.a
  font-height: $x
  padding-#{ $y}: $x + 1.0em

//_sety.sass
```

```
$y: top

//app.css

.a {
  font-height: 1.5em
  padding-top: 2.5em
}
```

`$x` will always be set to `1.5em`, whereas `$y` will be set to `bottom`, only if it hasn't already been set, perhaps in an imported file.

RHS values may use the variable name directly when referencing it, e.g. `$x`; whereas LHS values - usually property names and selectors - must escape the variable Ruby-style, e.g. `#{ $x }`.

Mixin, Extend

```
//app.sass

=boxsz($type: border-box)
  -moz-box-sizing: $type
  -webkit-box-sizing: $type
  box-sizing: $type

%bord
  border: 2px solid

.a
  @extend bord
  +boxsz

//app.css

.a {
  border: 2px solid
  -moz-box-sizing: border-box
  -webkit-box-sizing: border-box
  box-sizing: border-box
}
```

Mixins are blocks of reusable code, which can accept parameters.

Mixins should be defined before they are included - import order matters.

Here `border-box` is defined as the default value of the `$type` parameter.

A `%` preceding a selector denotes it as a placeholder selector, meaning that it can be referenced by `@extend` but will not output to the CSS file. If you remove the `%` symbol, it becomes a regular selector, and will be output to the CSS. Use placeholder selectors to reduce bloat.

Use mixins when there are variations when reused, use `extend` when there are no variations when reused. Thus, if you find yourself using a mixin without a parameter, you may consider switching to

extend.

Functions

```
//app.sass

@function calcwidth($w)
  $fullwidth: 720px
  @if $w <= 72px
    @return 10%
  @else
    @return percentage($w / $fullwidth)

.a
  width: calcwidth($w: 360px)

//app.css

.a {
  width: 50%;
}
```

Here we define a function `calcwidth` that takes in a single parameter that has no default value. This function has a single local variable `$fullwidth`, and makes use of the inbuilt SASS function `percentage`. This function also uses an `@if .. @else` conditional block to specify a minimum for its return value.

This function is called from within the `.a` selector. The parameter has been passed into this function using a more verbose syntax, which is usually used for functions with many parameters.

Colour

```
//app.sass

$mycolour: #fff
.a
  background-color: rgba($mycolour, 0.75)
  color: complement($mycolour)

//app.css

.a {
  background-color: rgba(255,255,255,0.75);
}
```

SASS identifies where a colour is used in the `rgba` function, and automatically converts it to the expected format.

Iteration

```
//app.sass

$dirs: top left

$m: 5px
$y: 0
.a
  @each $dir in $dirs
    margin-#{ $dir}: $m
    $m: $m + 10px
  @for $x from 0 through 1
    .for#{ $x}
      left: $x * 100px;
  @while $y < 3
    .while#{ $y}
      left: $y * 100px;
      $y: $y + 2

//app.css
```

```
.a {
  margin-top: 5px;
  margin-left: 15px;
}
.a .for0 {
  left: 0px;
}
.a .for1 {
  left: 100px;
}
.a .while0 {
  left: 0px;
}
.a .while2 {
  left: 200px;
}
```

SASS provides three different means to perform iteration: `@each`, `@for`, and `@while` - in increasing order of verbosity and control afforded.

Media Queries

```
//app.sass

=respond-to($mediaProperty, $expectedValue)
  @media ( $mediaProperty: $expectedValue)
    @content

.a
  +respond-to(max-width, 520px)
    width: 100%

//app.css

.a {
  width: 100%;
```

```
}
```

This pattern utilises media queries to detect if the current device's `max-width` is 520px.

The mixin in this case is passed not only parameters, but also a block. The mixin accesses this block using `@content`.

This pattern is useful in creating responsive design targeting multiple devices.

- [Jun 17](#)
- [0 Comments](#)
- [1](#)
- [html5](#) [css](#) [sass](#) [blog](#)
- [Facebook](#)
- [Twitter](#)
- [Google](#)
- [Tumblr](#)

0 Comments

Brendan Graetz

 Login ▾

Sort by Best ▾

Share  Favorite ★



Start the discussion...


Be the first to comment.

ALSO ON BRENDAN GRAETZ

WHAT'S THIS?

BackboneJs Grouped Collection Techniques - Four Approaches


2 comments • 5 months ago



bguiz — Yes, both are indeed great suggestions, and indeed should be considered when ...

A basic jQuery examination


2 comments • 6 months ago



Rob Pocklington — I'm sure your suggestions would break all the important unit tests they have ...

Accessors vs Dirty-checking in Javascript Frameworks

3 comments • 8 months ago



bguiz — Thanks for bringing this to attention - cannot believe that I have not noticed all this while. ...

Priceconomics Blog: The Tyranny of the Taxi Medallions

1 comment • 9 months ago





Prague transportation — Yes I uy hj

 Subscribe

 Add Disqus to your site

1 note

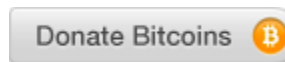
1.  [khayhurst](#) likes this
2.  [bguiz](#) posted this
- -

- [Random post](#)
- [Browse the Archive](#)
- [Get the RSS](#)
- [Ask me anything](#)
- [Submit](#)

Connect

- [Twitter](#)
- [Linkedin](#)
- [Facebook](#)
- [Github](#)

Donate



[To Tumblr, Love Pixel Union](#)