

ASSEMBLING SASS



PROPERTY OF:
CODE SCHOOL

PROJECT DATE:
SEP, 2012



This project was created and developed by
David E. Johnson, M.Ed., S. M.Ed.
and is the property of the Code School
Assembly Co. It is not to be used for
any other purpose without the written
consent of the Code School Assembly Co.
All rights reserved. © 2012

S

STANDARD
INSPECTION : NO 18



FOUNDATION



1.1 Sass, Not SASS

1.2 SCSS: Sassy CSS

1.3 Commenting

1.4 Importing

1.5 Nesting Selectors

1.6 The Parent Selector

1.7 Nesting Pitfalls



CSS is crafted to be **simple**,
but scaling simplicity is *difficult*.

1.1 Sass, Not SASS



At Scale

- Slight variations of colors, fonts, numbers, & other properties arise
- Effective curbing of repetition can decline
- Stylesheet size may become unmanageable

1.1 Sass, Not SASS



Enter Sass

- ◉ Syntactically Awesome Stylesheets
- ◉ Looks like CSS, but adds features to combat shortcomings
- ◉ Preprocessor, like CoffeeScript & Haml:



Sass File



Sass Compiler



CSS File

1.1 Sass, Not SASS



- Created by Hampton Catlin
- **Primary developers:**
Nathan Weizenbaum & Chris Eppstein
- Baked into Rails

1.1 Sass, Not SASS



Assembly Tip

~~SASS~~ Sass



1.1 Sass, Not SASS

1.2 SCSS: Sassy CSS

1.3 Commenting

1.4 Importing

1.5 Nesting Selectors

1.6 The Parent Selector

1.7 Nesting Pitfalls



- Sassy CSS (`.scss`) is the default file extension
- CSS **is valid** SCSS
- A second syntax (`.sass`) exists, but we'll focus on SCSS for the course

1.2 SCSS: Sassy CSS



application.scss

```
$main: #444;

.btn {
  color: $main;
  display: block;
}
.btn-a {
  color: lighten($main, 30%);
  &:hover {
    color: lighten($main, 40%);
  }
}
```

application.css

```
.btn {
  color: #444444;
  display: block;
}
.btn-a {
  color: #919191;
}
.btn-a:hover {
  color: #aaaaaa;
}
```

1.2 SCSS: Sassy CSS



Assembly Tip

Since CSS doubles as valid SCSS, try writing styles normally & slowly incorporate new techniques.



- Sass adds `//` for single line comments - **not** output after compile

1.3 Commenting



application.scss

```
// These comments will  
// not be output to the  
// compiled CSS file  
  
/* This comment will */
```

application.css

```
/* This comment will */
```

1.3 Commenting



application.css

```
/*  
 * Imports styles found in 'buttons.css'  
 * when the browser requests application.css  
 */
```

```
@import "buttons.css";
```



1.4 Importing



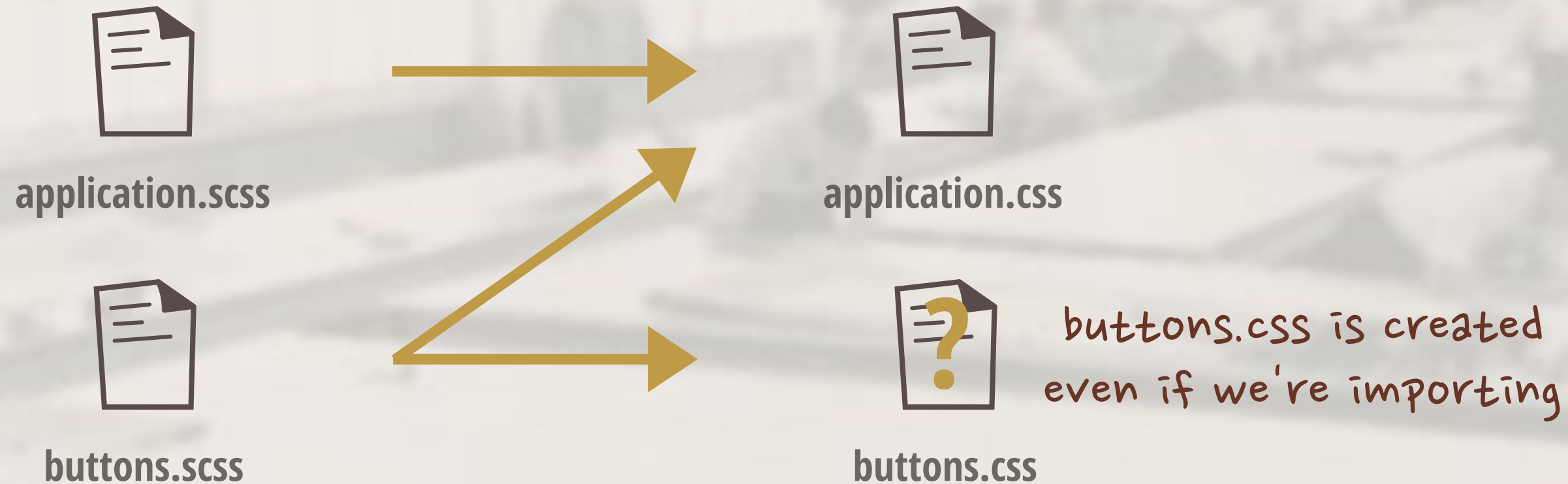
- The CSS `@import` rule has been avoided: prevents parallel downloading
- `@import` with `.scss` or `.sass` happens **during compile** rather than client-side
- File extension is optional

1.4 Importing



application.scss

```
// Imports styles found in 'buttons.scss'  
// when the compiler processes application.scss  
  
@import "buttons";
```



1.4 Importing



Partials

Adding an underscore creates a **partial**. Partials can be imported, but will not compile to `.css`

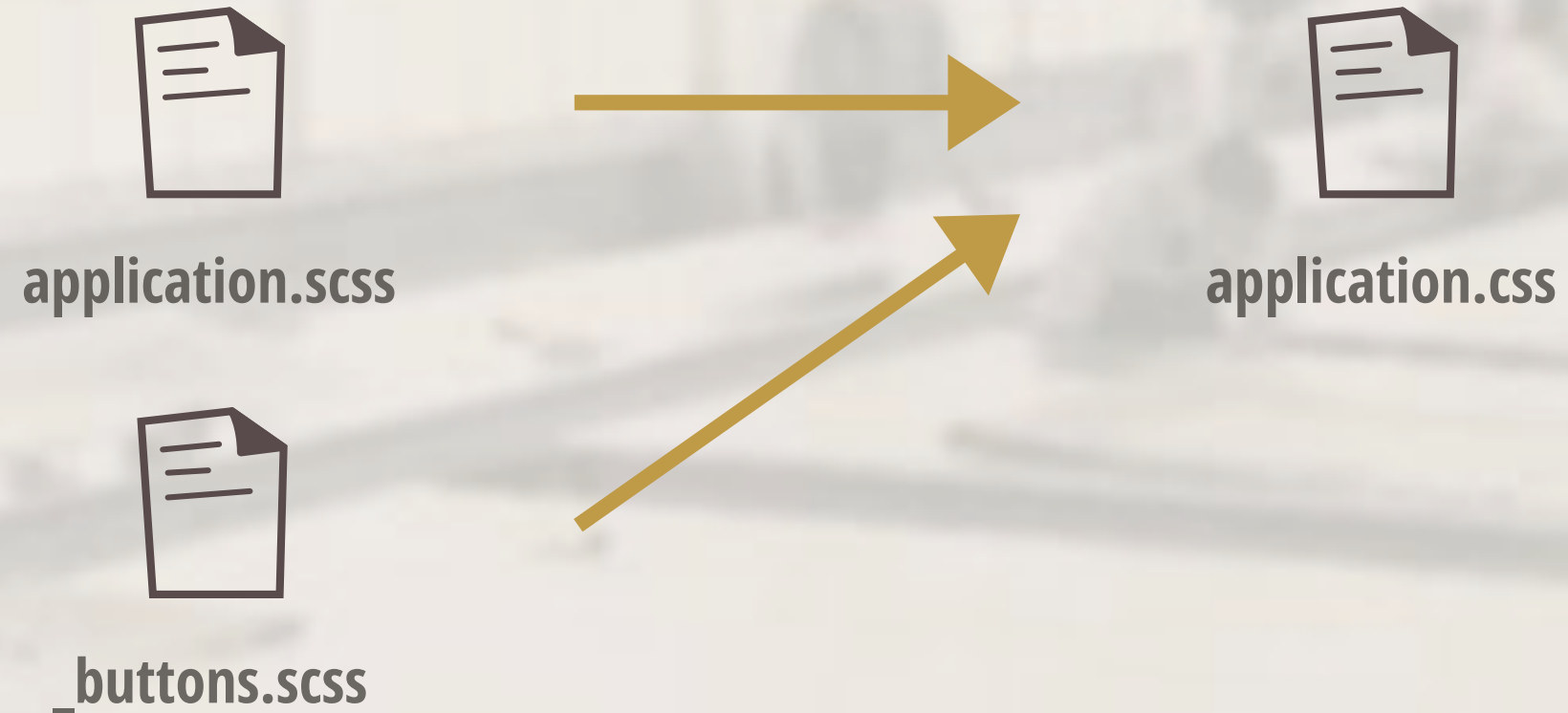


1.4 Importing



application.scss

```
// Will import _buttons.sass, buttons.sass,  
// _buttons.scss, or buttons.scss  
  
@import "buttons";
```



1.4 Importing



1.1 Sass, Not SASS

1.2 SCSS: Sassy CSS

1.3 Commenting

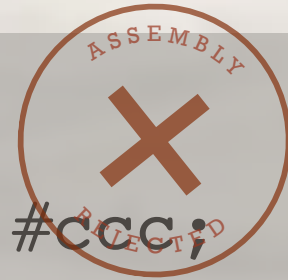
1.4 Importing

1.5 Nesting Selectors

1.6 The Parent Selector

1.7 Nesting Pitfalls





```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
▶ .content h2 {  
  font-size: 3em;  
  margin: 20px 0;  
}  
▶ .content p {  
  font-size: 1.5em;  
  margin: 15px 0;  
}
```

1.5 Nesting Selectors



application.scss

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.content h2 {  
  font-size: 3em;  
  margin: 20px 0;  
}  
.content p {  
  font-size: 1.5em;  
  margin: 15px 0;  
}
```

application.css

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.content h2 {  
  font-size: 3em;  
  margin: 20px 0;  
}  
.content p {  
  font-size: 1.5em;  
  margin: 15px 0;  
}
```

1.5 Nesting Selectors

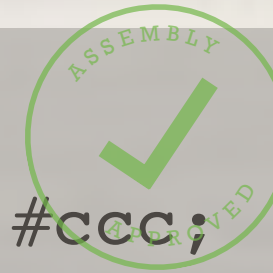


application.scss

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
  h2 {  
    font-size: 3em;  
    margin: 20px 0;  
  }  
  p {  
    font-size: 1.5em;  
    margin: 15px 0;  
  }  
}
```

application.css

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.content h2 {  
  font-size: 3em;  
  margin: 20px 0;  
}  
.content p {  
  font-size: 1.5em;  
  margin: 15px 0;  
}
```



1.5 Nesting Selectors



Nesting Properties

Certain properties with matching namespaces are nestable:

application.scss

```
.btn {  
  text: {  
    decoration: underline;  
    transform: lowercase;  
  }  
}
```

application.css

```
.btn {  
  text-decoration: underline;  
  text-transform: lowercase;  
}
```

1.5 Nesting Selectors



While nesting, the & symbol references the parent selector:

application.scss

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
  .callout {  
    border-color: red;  
  }  
  &.callout {  
    border-color: green;  
  }  
}  
references:  
  .content
```



application.css

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.content .callout {  
  border-color: red;  
}  
.content .callout {  
  border-color: green;  
}
```



1.6 The Parent Selector



application.scss

```
a {  
  color: #999;  
  &:hover {  
    color: #777;  
  }  
  &:active {  
    color: #888;  
  }  
}
```



application.css

```
a {  
  color: #999;  
}  
a:hover {  
  color: #777;  
}  
a:active {  
  color: #888;  
}
```

1.6 The Parent Selector



Parent Selector Nesting

Selectors can also be added **before** the & reference:

application.css

```
.sidebar {  
  float: right;  
  width: 300px;  
}  
.users .sidebar {  
  width: 400px;  
}
```

← modifies .sidebar in a
separate declaration



1.6 The Parent Selector



application.scss

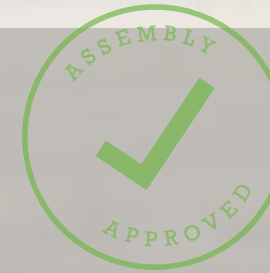
```
.sidebar {  
  float: right;  
  width: 300px;  
  .users & {  
    width: 400px;  
  }  
}
```

references:
.sidebar



application.css

```
.sidebar {  
  float: right;  
  width: 300px;  
}  
.users .sidebar {  
  width: 400px;  
}
```



1.6 The Parent Selector



application.scss

```
.sidebar {  
  float: right;  
  width: 300px;  
  h2 {  
    color: #777;  
    .users & {  
      color: #444;  
    }  
  }  
}
```

references:
.sidebar h2

application.css

```
.sidebar {  
  float: right;  
  width: 300px;  
}  
.sidebar h2 {  
  color: #777;  
}  
.users .sidebar h2 {  
  color: #444;  
}
```

1.6 The Parent Selector



- Nesting is easy, but **dangerous**
- **Do not nest unnecessarily**

1.7 Nesting Pitfalls



application.scss

```
.content {  
  background: #ccc;  
  .cell {  
    h2 {  
      a {  
        &:hover {  
          color: red;  
        }  
      }  
    }  
  }  
}
```



application.css

```
.content {  
  background: #ccc;  
}  
.content .cell h2 a:hover {  
  color: red;  
}
```



dangerous level of
specificity

1.7 Nesting Pitfalls



Assembly Tip

Try limiting your nesting to 3 or 4 levels and consider refactoring anything deeper.



2 VARIABLE



2.1 Variable Declaration + Use

2.2 Types

2.3 Scope

2.4 Interpolation



- Native CSS variable support is still in its infancy, but Sass affords us a way to set reusable values
- Variable names begin with \$, like \$base

2.1 Variable Declaration + Use



application.scss

```
$base: #777777;

.sidebar {
  border: 1px solid $base;
  p {
    color: $base;
  }
}
```

application.css

```
.sidebar {
  border: 1px solid #777777;
}
.sidebar p {
  color: #777777;
}
```

2.1 Variable Declaration + Use



The Default Flag

Variable definitions can optionally take the `!default` flag:

application.scss

```
$title: 'My Blog';  
$title: 'About Me';
```

```
h2:before {  
  content: $title;  
}
```

overrides the
first value

application.css

```
h2:before {  
  content: 'About Me';  
}
```

2.1 Variable Declaration + Use



application.scss

```
$title: 'My Blog';  
$title: 'About Me' !default;  
  
h2:before {  
  content: $title;  
}
```



Since a value exists,
it isn't overwritten

application.css

```
h2:before {  
  content: 'My Blog';  
}
```

2.1 Variable Declaration + Use



application.scss

```
$rounded: 5px;  
  
@import "buttons";
```

_buttons.scss

```
$rounded: 3px !default;  
  
.btn-a {  
  border-radius: $rounded;  
  color: #777;  
}  
  
.btn-b {  
  border-radius: $rounded;  
  color: #222;  
}
```

if a value isn't
defined elsewhere,
used by default

2.1 Variable Declaration + Use



application.scss

```
$rounded: 5px;  
  
@import "buttons";
```

application.css

```
.btn-a {  
  border-radius: 5px;  
  color: #777;  
}  
.btn-b {  
  border-radius: 5px;  
  color: #222;  
}
```

2.1 Variable Declaration + Use



Booleans

```
$rounded: false;  
$shadow: true;
```

Numbers - can be set with or without units:

```
$rounded: 4px;  
$line-height: 1.5;  
$font-size: 3rem;
```

2.2 Types



Colors

```
$base: purple;  
$border: rgba(0, 255, 0, 0.5);  
$shadow: #333;
```

Strings - can be set with or without quotes:

```
$header: 'Helvetica Neue';  
$callout: Arial;  
$message: "Loading...";
```

2.2 Types



Lists

```
$authors: nick, dan, aimee, drew;  
$margin: 40px 0 20px 100px;
```

Null

```
$shadow: null;
```



2.1 Variable Declaration + Use



2.2 Types



2.3 Scope



2.4 Interpolation



application.scss

```
p {  
  $border: #ccc;  
  border-top: 1px solid $border;  
}  
h1 {  
  border-top: 1px solid $border;  
}
```


\$border isn't available
outside of p

application.css

```
Syntax error: Undefined  
variable: "$border"
```

2.3 Scope



Reassignment in a Declaration

- Variables **set** inside a declaration (within { }) aren't usable outside that block
- Setting **new** values to variables set outside a declaration changes future instances



application.scss

```
$color-base: #777777;

.sidebar {
  $color-base: #222222;
  background: $color-base;
}

p {
  color: $color-base;
}
```

application.css

```
.sidebar {
  background: #222222;
}

p {
  color: #222222;
}
```

overwriting a variable in a
declaration is global

2.3 Scope



Use the Ruby-esque `#{$variable}` to shim variables into selectors, property names, and strings:

application.scss

```
$side: top;

sup {
  position: relative;
  #{$side}: -0.5em;
}

.callout-#{$side} {
  background: #777;
}
```

application.css

```
sup {
  position: relative;
  top: -0.5em;
}

.callout-top {
  background: #777;
}
```

2.4 Interpolation



Assembly Tip

Be considerate of variable naming. `$color-base` gets a lot more mileage than `$color-blue`.



3

MIXIN



3.1 Mixin Setup + Use

3.2 Adding Arguments

3.3 Multiple Arguments

3.4 Variable Arguments

3.5 Interpolation + Mixins



```
.btn-a {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}
```



repeating properties
in each declaration



3.1 Mixin Setup + Use



Mixins

Blocks of reusable code that take optional arguments:

`_buttons.scss`

```
► @mixin button {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}
```

3.1 Mixin Setup + Use



_buttons.scss

```
@mixin button {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-a {  
  @include button;  
  background: #777;  
}  
.btn-b {  
  @include button;  
  background: #ff0;  
}
```

application.css

```
.btn-a {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
  background: #777;  
}  
.btn-b {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
  background: #ff0;  
}
```

3.1 Mixin Setup + Use



Assembly Tip

Make sure the `@mixin` block comes before the `@include`, especially when importing files containing mixins.



Assembly Tip

`@include` = use a mixin

`@import` = import a file



```
@mixin button {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-a {  
  @include button;  
  background: #777;  
}  
.btn-b {  
  @include button;  
  background: #ff0;  
}
```

```
.btn-a {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
  background: #777;  
}  
.btn-b {  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
  background: #ff0;  
}
```



repeating properties
in each declaration

3.1 Mixin Setup + Use



We're Just Repeating Properties

It's more efficient to use CSS here (for now):

application.css

```
.btn-a,  
.btn-b {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
}
```

3.1 Mixin Setup + Use



If that's the case, what are
mixins good for then?

3.1 Mixin Setup + Use



```
.content {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
  border: 1px solid #ccc;  
  padding: 20px;  
}
```

writing three
mostly-identical
properties gets old



3.2 Adding Arguments



```
@mixin box-sizing {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
}  
.content {  
  @include box-sizing;  
  border: 1px solid #ccc;  
  padding: 20px;  
}
```

```
.content {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
  border: 1px solid #ccc;  
  padding: 20px;  
}
```

still just copying
unchanging properties



3.2 Adding Arguments



Arguments

Values passed into a mixin, potentially altering output:

application.scss



```
@mixin box-sizing($x) {  
  -webkit-box-sizing: $x;  
  -moz-box-sizing: $x;  
  box-sizing: $x;  
}
```

3.2 Adding Arguments



```
@mixin box-sizing($x) {  
  -webkit-box-sizing: $x;  
  -moz-box-sizing: $x;  
  box-sizing: $x;  
}  
.content {  
  @include box-sizing(border-box);  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.callout {  
  @include box-sizing(content-box);  
}
```

```
.content {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.callout {  
  -webkit-box-sizing: content-box;  
  -moz-box-sizing: content-box;  
  box-sizing: content-box;  
}
```

3.2 Adding Arguments



Default Values

Optionally, what arguments will default to if not included:

application.scss



```
@mixin box-sizing($x: border-box) {  
  -webkit-box-sizing: $x;  
  -moz-box-sizing: $x;  
  box-sizing: $x;  
}
```

3.2 Adding Arguments



```
@mixin box-sizing($x: border-box) {  
  -webkit-box-sizing: $x;  
  -moz-box-sizing: $x;  
  box-sizing: $x;  
}  
.content {  
  @include box-sizing;  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.callout {  
  @include box-sizing(content-box);  
}
```

border-box, if no
argument is passed



```
.content {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.callout {  
  -webkit-box-sizing: content-box;  
  -moz-box-sizing: content-box;  
  box-sizing: content-box;  
}
```



3.2 Adding Arguments



3.1 Mixin Setup + Use

3.2 Adding Arguments

3.3 Multiple Arguments

3.4 Variable Arguments

3.5 Interpolation + Mixins



_buttons.scss

```
@mixin button($radius, $color) {  
  border-radius: $radius;  
  color: $color;  
}  
.btn-a {  
  @include button(4px, #000);  
}
```

arguments are
comma-separated and
passed in order

application.css

```
.btn-a {  
  border-radius: 4px;  
  color: #000;  
}
```

3.3 Multiple Arguments



_buttons.scss

```
@mixin button($radius, $color) {  
  border-radius: $radius;  
  color: $color;  
}  
.btn-a {  
  @include button(4px);  
}
```

too few arguments



application.css

Syntax error: Mixin button is missing argument \$color.

3.3 Multiple Arguments



_buttons.scss

```
@mixin button($radius, $color: #000) {  
  border-radius: $radius;  
  color: $color;  
}  
.btn-a {  
  @include button(4px);  
}
```

optional second argument



application.css

```
.btn-a {  
  border-radius: 4px;  
  color: #000;  
}
```

3.3 Multiple Arguments



_buttons.scss

```
@mixin button($color: #000, $radius) {  
  border-radius: $radius;  
  color: $color;  
}  
.btn-a {  
  @include button(4px);  
}
```

optionals come last



application.css

Syntax error: Required argument
\$color must come before any
optional arguments.

3.3 Multiple Arguments



_buttons.scss

```
@mixin button($radius, $color: #000) {  
  border-radius: $radius;  
  color: $color;  
}  
.btn-a {  
  @include button($color: #777777,  
$radius: 5px);  
}
```

keyword arguments allow
passing in any order

application.css

```
.btn-a {  
  border-radius: 5px;  
  color: #777777;  
}
```

3.3 Multiple Arguments



```
.btn-a {  
  -webkit-transition: color 0.3s ease-in, background 0.5s ease-out;  
  -moz-transition: color 0.3s ease-in, background 0.5s ease-out;  
  transition: color 0.3s ease-in, background 0.5s ease-out;  
}
```



commas can naturally
occur in CSS values

3.4 Variable Arguments



Passing valid, comma-separated CSS as a single value:

_buttons.scss

```
@mixin transition($val) {  
  -webkit-transition: $val;  
  -moz-transition: $val;  
  transition: $val;  
}  
.btn-a {  
  @include transition(color 0.3s  
ease-in, background 0.5s ease-out);  
}
```

application.css

Mixin transition takes 1
argument but 2 were passed.

3.4 Variable Arguments



Adding ... to an argument creates a **variable argument** (vararg):

_buttons.scss

```
@mixin transition($val...) {  
  -webkit-transition: $val;  
  -moz-transition: $val;  
  transition: $val;  
}  
.btn-a {  
  @include transition(color 0.3s  
ease-in, background 0.5s ease-out);  
}
```

application.css

```
.btn-a {  
  -webkit-transition: color 0.3s  
ease-in, background 0.5s ease-out;  
  -moz-transition: color 0.3s  
ease-in, background 0.5s ease-out;  
  transition: color 0.3s ease-in,  
background 0.5s ease-out;  
}
```

3.4 Variable Arguments



Variable arguments in reverse:

_buttons.scss

```
@mixin button($radius, $color) {  
  border-radius: $radius;  
  color: $color;  
}
```

```
$properties: 4px, #000;
```

```
.btn-a {  
  @include button($properties...);  
}
```

passes a list which is split
into arguments by the mixin

application.css

```
.btn-a {  
  border-radius: 4px;  
  color: #000;  
}
```

3.4 Variable Arguments



_buttons.scss

```
@mixin highlight-t($color) {  
  border-top-color: $color;  
}  
@mixin highlight-r($color) {  
  border-right-color: $color;  
}  
@mixin highlight-b($color) {  
  border-bottom-color: $color;  
}  
@mixin highlight-l($color) {  
  border-left-color: $color;  
}  
.btn-a {  
  @include highlight-r(#f00);  
}
```

application.css

```
.btn-a {  
  border-right-color: #ff0;  
}
```



3.5 Interpolation + Mixins



_buttons.scss

```
@mixin highlight($color, $side) {  
  border-#{$side}-color: $color;  
}  
.btn-a {  
  @include highlight(#f00, right);  
}
```

application.css

```
.btn-a {  
  border-right-color: #ff0;  
}
```



3.5 Interpolation + Mixins



4

EXTEND



4.1 Extend Setup + Use

4.2 Nesting + Extend

4.3 Extend Pitfalls

4.4 Placeholder Selectors



application.css

```
.btn-a {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}
```

simplifies to



application.css

```
.btn-a,  
.btn-b {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
}
```

4.1 Extend Setup + Use



Extend

Sass will track and automatically combine selectors for us:

_buttons.scss

```
.btn-a {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  @extend .btn-a;  
  background: #ff0;  
}
```

application.css

```
.btn-a,  
.btn-b {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
}
```

4.1 Extend Setup + Use



```
.btn-b {  
  @extend .btn-a;  
  background: #ff0;  
}
```

```
.btn-a,  
.btn-b {  
  ...  
}  
.btn-b {  
  ...  
}
```

adds a second declaration
for unique values

4.1 Extend Setup + Use



application.scss

```
.content {  
  border: 1px solid #ccc;  
  padding: 20px;  
  h2 {  
    font-size: 3em;  
    margin: 20px 0;  
  }  
}  
.callout {  
  @extend .content;  
  background: #ddd;  
}
```

application.css

```
.content,  
.callout {  
  border: 1px solid #ccc;  
  padding: 20px;  
}  
.content h2,  
.callout h2 {  
  font-size: 3em;  
  margin: 20px 0;  
}  
.callout {  
  background: #ddd;  
}
```

4.2 Nesting + Extend



```
.callout {  
  @extend .content;  
  background: #ddd;  
}
```

```
.content,  
.callout {  
  ...  
}  
.content h2,  
.callout h2 {  
  ...  
}  
.callout {  
  ...  
}
```

4.2 Nesting + Extend



4.1 Extend Setup + Use



4.2 Nesting + Extend



4.3 Extend Pitfalls

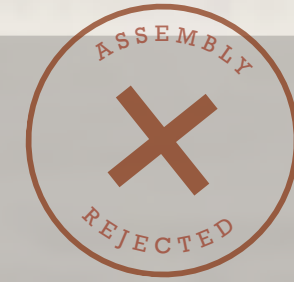


4.4 Placeholder Selectors



```
.btn-a {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  @extend .btn-a;  
  background: #ff0;  
}  
.sidebar .btn-a {  
  text-transform: lowercase;  
}
```

```
.btn-a,  
.btn-b {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
}  
.sidebar .btn-a,  
.sidebar .btn-b {  
  text-transform: lowercase;  
}
```



.btn-b is also
scoped here

4.3 Extend Pitfalls



- Since `.btn-b` extends `.btn-a`, every instance that modifies `.btn-a` also modifies `.btn-b`
- Stylesheet bloat, if these extra styles aren't needed
- We can counteract with **placeholder selectors**

4.3 Extend Pitfalls



Assembly Tip

Always, always check the CSS output of your Sass before using it on a live site.



- Placeholder selectors are denoted with a %
- Can be extended, but never become a selector of their own

4.4 Placeholder Selectors



_buttons.scss

```
.btn-a {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  @extend .btn-a;  
  background: #ff0;  
}  
.sidebar .btn-a {  
  text-transform: lowercase;  
}
```

application.css

```
.btn-a,  
.btn-b {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
}  
.sidebar .btn-a,  
.sidebar .btn-b {  
  text-transform: lowercase;  
}
```

4.4 Placeholder Selectors



```
%btn {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-a {  
  @extend %btn;  
}  
.btn-b {  
  @extend %btn;  
  background: #ff0;  
}  
.sidebar .btn-a {  
  text-transform: lowercase;  
}
```

```
.btn-a,  
.btn-b {  
  background: #777;  
  border: 1px solid #ccc;  
  font-size: 1em;  
  text-transform: uppercase;  
}  
.btn-b {  
  background: #ff0;  
}  
.sidebar .btn-a {  
  text-transform: lowercase;  
}
```

*.btn-b is no
longer scoped*



4.4 Placeholder Selectors



Extend common blocks to avoid extra HTML classes:

application.scss

```
%ir {  
  border: 0;  
  font: 0/0 a;  
  text-shadow: none;  
  color: transparent;  
  background-color: transparent;  
}  
.logo {  
  @extend %ir;  
}  
.social {  
  @extend %ir;  
}
```

application.css

```
.logo,  
.social {  
  border: 0;  
  font: 0/0 a;  
  text-shadow: none;  
  color: transparent;  
  background-color: transparent;  
}
```

4.4 Placeholder Selectors



Assembly Tip

Versions of IE prior to 9 have a limit of 4095 selectors-per-CSS file limit.



5 DIRECTIVE



5.1 Functions

5.2 If

5.3 Each

5.4 For + While

5.5 Mixin' In



Responsive Refresher

- Straight from Journey Into Mobile:

```
target / context
```

- If the target size of our sidebar is 350px and the context of its parent is 1000px:

```
350px / 1000px = 0.35  
0.35 * 100 = 35%
```

5.1 Functions



application.scss

```
@function fluidize() {  
  @return 35%;  
}  
.sidebar {  
  width: fluidize();  
}
```

always returns 35%

application.css

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

5.1 Functions



application.scss

```
@function fluidize($target, $context) {  
  @return ($target / $context) * 100%;  
}  
.sidebar {  
  width: fluidize(350px, 1000px);  
}
```

application.css

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

5.1 Functions



- More on responsive design + Sass later, including a built-in `fluidize` replacement
- Function arguments = **same rules** as mixin arguments

5.1 Functions



Using `@if`, we can conditionally output code:

application.scss

```
$theme: dark;

header {
  @if $theme == dark {
    background: #000;
  }
}
```

application.css

```
header {
  background: #000;
}
```

5.2 If



Comparisons

- ⦿ == equal to
- ⦿ != not equal to
- ⦿ > greater than *
- ⦿ >= greater than or equal to *
- ⦿ < less than *
- ⦿ <= less than or equal to *

* numbers only

5.2 If



application.scss

▶ \$theme: light;

```
header {  
  @if $theme == dark {  
    background: #000;  
  }  
}
```

5.2 If



@else provides a fallback if everything evaluates false or null:

application.scss

```
$theme: light;

header {
  @if $theme == dark {
    background: #000;
  } @else {
    background: #fff;
  }
}
```

application.css

```
header {
  background: #fff;
}
```

5.2 If



@else if allows for multiple comparisons:

application.scss

```
$theme: pink;

header {
  @if $theme == dark {
    background: #000;
  } @else if $theme == pink {
    background: pink;
  } @else {
    background: #fff;
  }
}
```

application.css

```
header {
  background: pink;
}
```

5.2 If



5.1 Functions



5.2 If



5.3 Each



5.4 For + While



5.5 Mixin' In



Iterating Over a List

- The `@each` directive allows us to loop through each list item:

```
$authors: nick aimee dan drew;
```



application.scss

```
$authors: nick aimee dan drew;

@each $author in $authors {
  .author-#{ $author } {
    background: url(author-
#{ $author }.jpg);
  }
}
```

\$author →

- 1: nick
- 2: aimee
- 3: dan
- 4: drew

application.css

```
.author-nick {
  background: url(author-nick.jpg);
}
.author-aimee {
  background: url(author-aimee.jpg);
}
.author-dan {
  background: url(author-dan.jpg);
}
.author-drew {
  background: url(author-drew.jpg);
}
```

5.3 Each



```
.item {  
  position: absolute;  
  right: 0;  
  @for $i from 1 through 3 {  
    &.item-#{ $i } {  
      top: $i * 30px;  
    }  
  }  
}
```

$\$i \longrightarrow$

1:	1
2:	2
3:	3

```
.item {  
  position: absolute;  
  right: 0;  
}  
.item.item-1 {  
  top: 30px;  
}  
.item.item-2 {  
  top: 60px;  
}  
.item.item-3 {  
  top: 90px;  
}
```

5.4 For + While



- `@for` and `@while` = `@each` with more control
- `@while` requires manually updating the index

5.4 For + While



▶ `$i: 1;`

▶ `.item {`
`position: absolute;`
`right: 0;`
 ▶ `@while $i < 4 {`
 `&.item-#{ $i } {`
 `top: $i * 30px;`
 `}`
 `$i: $i + 1;`
`}`
`}`

`$i` →

1:	1
2:	2
3:	3
4:	4

```
.item {
  position: absolute;
  right: 0;
}
.item.item-1 {
  top: 30px;
}
.item.item-2 {
  top: 60px;
}
.item.item-3 {
  top: 90px;
}
```

5.4 For + While



 `$i: 2;` `.item {
 position: absolute;
 right: 0;
 @while $i <= 6 {
 &.item-#{ $i } {
 top: $i * 30px;
 }
 $i: $i + 2;
 }
}``$i` `1: 2``2: 4``3: 6``4: 8`

```
.item {  
  position: absolute;  
  right: 0;  
}  
.item.item-2 {  
  top: 60px;  
}  
.item.item-4 {  
  top: 120px;  
}  
.item.item-6 {  
  top: 180px;  
}
```

5.4 For + While



5.1 Functions



5.2 If



5.3 Each



5.4 For + While



5.5 Mixin' In



Mixins

- *Similar* sets of properties used multiple times with small variations

Extend

- Sets of properties that match *exactly*

Functions

- Commonly-used operations to determine *values*



_buttons.scss

```
@mixin button($color, $rounded: true) {  
  color: $color;  
  @if $rounded == true {  
    border-radius: 4px;  
  }  
}  
.btn-a {  
  @include button(#000, false);  
}  
.btn-b {  
  @include button(#333);  
}
```

application.css

```
.btn-a {  
  color: black;  
}  
.btn-b {  
  color: #333333;  
  border-radius: 4px;  
}
```

5.5 Mixin' In



```
@mixin button($color, $rounded: false) {  
  color: $color;  
  @if $rounded {  
    border-radius: $rounded;  
  }  
}  
.btn-a {  
  @include button(#000);  
}  
.btn-b {  
  @include button(#333, 4px);  
}
```

used if \$rounded
isn't false or null

```
.btn-a {  
  color: black;  
}  
.btn-b {  
  color: #333333;  
  border-radius: 4px;  
}
```

5.5 Mixin' In



6

MATH + COLOR



6.1 Basic Arithmetic

6.2 Differing Units

6.3 Math Functions

6.4 Math + Color

6.5 Color Shortcuts

6.6 Color Functions



Number Operations

- ⦿ + addition
- ⦿ – subtraction
- ⦿ * multiplication
- ⦿ / division
- ⦿ % modulo

Modulo = **remainder** from a division operation. $12 \% 3$ results in 0, while $12 \% 5$ returns 2.

6.1 Basic Arithmetic



Assembly Tip

Sass defaults to returning (up to) five digits after a decimal point.



Division

- The trickiest of the number operations, due to font:

```
font: normal 2em/1.5 Helvetica, sans-serif;
```

6.1 Basic Arithmetic



Triggering Division

- Variable involved - `$size / 10`
- Parenthesis - `(100px / 20)`
- Another arithmetic operation - `20px * 5 / 2`



String Addition

Addition on strings concatenates them:

```
$family: "Helvetica " + "Neue"; // "Helvetica Neue"
```

Initial left-side string determines post-concatenation quotes:

```
$family: 'sans-' + serif // 'sans-serif'  
$family: sans- + 'serif' // sans-serif
```

6.1 Basic Arithmetic



If the units differ, Sass attempts combination:

application.scss

```
h2 {  
  font-size: 10px + 4pt;  
}
```

application.css

```
h2 {  
  font-size: 15.33333px;  
}
```

6.2 Differing Units



Incompatible units will throw an error:

application.scss

```
h2 {  
  font-size: 10px + 4em;  
}
```

application.css

```
Incompatible units: 'em'  
and 'px'.
```

6.2 Differing Units



Pre-Defined Math Utilities

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

6.3 Math Functions



Called the same way as custom functions:

application.scss

```
h2 {  
  line-height: ceil(1.2);  
}
```

application.css

```
h2 {  
  line-height: 2;  
}
```

6.3 Math Functions



percentage() replaces our custom fluidize():

application.scss

```
.sidebar {  
  width: percentage(350px/1000px);  
}
```

application.css

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

6.3 Math Functions



percentage() replaces our custom fluidize():

application.scss

```
$context: 1000px;

.sidebar {
  width: percentage(450px/$context);
}
```

application.css

```
.sidebar {
  width: 45%;
}
```

6.3 Math Functions



6.1 Basic Arithmetic



6.2 Differing Units



6.3 Math Functions



6.4 Math + Color



6.5 Color Shortcuts



6.6 Color Functions



Color Juggling

- ⦿ Easier recall through variables
- ⦿ Simplified alteration via **color utility functions**
- ⦿ Faster representation using **shorthand**



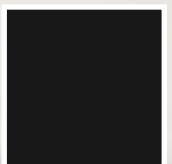
application.scss

```
$color-base: #333333;

.addition {
  background: $color-base + #112233;
}
.subtraction {
  background: $color-base - #112233;
}
.multiplication {
  background: $color-base * 2;
}
.division {
  background: $color-base / 2;
}
```

application.css

```
.addition {
  background: #445566;
}
.subtraction {
  background: #221100;
}
.multiplication {
  background: #666666;
}
.division {
  background: #191919;
}
```



6.4 Math + Color



Assembly Tip

Where possible, use color utility functions instead of color arithmetic: easier to predict and maintain.



application.scss

```
$color: #333333;  
  
.alpha {  
  background: rgba(51,51,51,0.8);  
}
```



manually finding the rgb
value of a color we
already have stored



application.css

```
.alpha {  
  background: rgba(51,51,51,0.8);  
}
```

6.5 Color Shortcuts



application.scss

```
$color: #333333;

.alpha {
  background: rgba($color, 0.8);
}
.beta {
  background: rgba(#000, 0.8);
}
```



can also use hex
values where appropriate



application.css

```
.alpha {
  background: rgba(51,51,51,0.8);
}
.beta {
  background: rgba(0,0,0,0.8);
}
```



6.5 Color Shortcuts



6.1 Basic Arithmetic



6.2 Differing Units



6.3 Math Functions



6.4 Math + Color



6.5 Color Shortcuts



6.6 Color Functions



Color utility functions:
workflow-altering convenience



application.scss

```
$color: #333;  
  
.lighten {  
  color: lighten($color, 20%);  
}  
.darken {  
  color: darken($color, 20%);  
}
```

application.css

```
.lighten {  
  background: #666666;  
}  
.darken {  
  background: black;  
}
```



6.6 Color Functions



application.scss

```
$color: #87bf64;

.saturate {
  color: saturate($color, 20%);
}
.desaturate {
  color: desaturate($color, 20%);
}
```

application.css

```
.saturate {
  background: #82d54e;
}
.desaturate {
  background: #323130;
}
```



6.6 Color Functions



application.scss

```
.mix-a {  
  color: mix(#ffff00, #107fc9);  
}  
.mix-b {  
  color: mix(#ffff00, #107fc9, 30%);  
}
```

application.css

```
.mix-a {  
  background: #87bf64;  
}  
.mix-b {  
  background: #57a58c;  
}
```



6.6 Color Functions



application.scss

```
$color: #87bf64;

.grayscale {
  color: grayscale($color);
}
.invert {
  color: invert($color);
}
.complement {
  color: complement($color);
}
```

application.css

```
.grayscale {
  color: #929292;
}
.invert {
  color: #78409b;
}
.complement {
  color: #9c64bf;
}
```



6.6 Color Functions



Assembly Tip

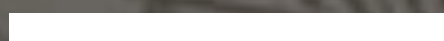
But wait, there's more!

<http://sass-lang.com/docs/yardoc/Sass/Script/Functions.html>



7

RESPONSIVE



7.1 The Movement

7.2 Nested Media Queries

7.3 Respond-To

7.4 Responsive Pitfalls



Responsive design rapidly
progressed beyond *good idea*
and into **common practice**

7.1 The Movement



Media Queries

- Easier fluid calculation and media query handling
- Journey Into Mobile

7.1 The Movement



Media Queries

Basic construction:

application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
▶ @media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```

7.1 The Movement



```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```

7.2 Nested Media Queries



application.scss

```
.sidebar {  
  border: 1px solid #ccc;  
  @media (min-width: 700px) {  
    float: right;  
    width: 30%;  
  }  
}
```

application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```



7.2 Nested Media Queries



7.1 The Movement

7.2 Nested Media Queries

7.3 Respond-To

7.4 Responsive Pitfalls



- @content - pass a block of properties to a mixin



application.scss

```
.sidebar {  
  border: 1px solid #ccc;  
  @media (min-width: 700px) {  
    float: right;  
    width: 30%;  
  }  
}
```

application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```

7.3 Respond-To



application.scss

```
@mixin respond-to {  
  @media (min-width: 700px) {  
    @content  
  }  
}  
.sidebar {  
  border: 1px solid #ccc;  
  @include respond-to {  
    float: right;  
    width: 30%;  
  }  
}
```

always outputs the
same media query



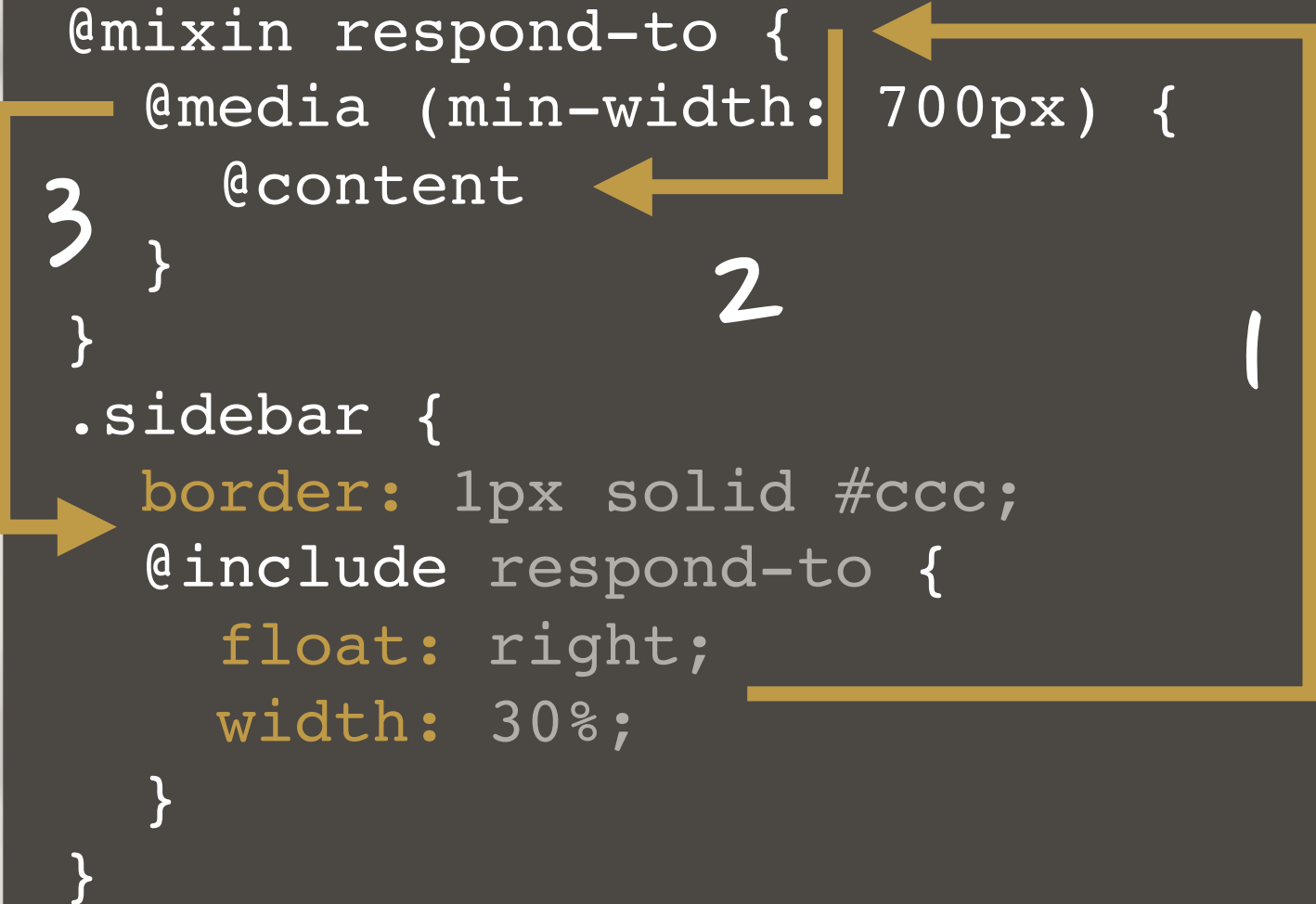
application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```

7.3 Respond-To



```
@mixin respond-to {  
  3  @media (min-width: 700px) {  
    2    @content  
  }  
}  
.sidebar {  
  border: 1px solid #ccc;  
  @include respond-to {  
    float: right;  
    width: 30%;  
  }  
}
```



```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```



7.3 Respond-To



application.scss

```
@mixin respond-to($media) {  
  @if $media == tablet {  
    @media (min-width: 700px) {  
      @content  
    }  
  }  
}  
  
.sidebar {  
  border: 1px solid #ccc;  
  @include respond-to(tablet) {  
    float: right;  
    width: 30%;  
  }  
}
```

application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
  
@media (min-width: 700px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```

7.3 Respond-To



application.scss

```
@mixin respond-to($query) {  
  @media (min-width: $query) {  
    @content  
  }  
}  
.sidebar {  
  border: 1px solid #ccc;  
  @include respond-to(900px) {  
    float: right;  
    width: 30%;  
  }  
}
```

application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (min-width: 900px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```

7.3 Respond-To



application.scss

```
@mixin respond-to($val, $query) {  
  @media ($val: $query) {  
    @content  
  }  
}  
.sidebar {  
  border: 1px solid #ccc;  
  @include respond-to(max-width,  
600px) {  
    float: right;  
    width: 30%;  
  }  
}
```

application.css

```
.sidebar {  
  border: 1px solid #ccc;  
}  
@media (max-width: 600px) {  
  .sidebar {  
    float: right;  
    width: 30%;  
  }  
}
```



7.3 Respond-To



7.1 The Movement



7.2 Nested Media Queries



7.3 Respond-To



7.4 Responsive Pitfalls



Declarations outside @media cannot be extended inside:

application.scss

```
.sidebar {  
  border: 1px solid #ccc;  
}  
.callout {  
  @media (min-width: 700px) {  
    @extend .sidebar;  
    width: 35%;  
  }  
}
```

deprecation warning for now,
compile error soon

7.4 Responsive Pitfalls



application.scss

```
@media (min-width: 700px) {  
  .content {  
    border: 1px solid #ccc;  
  }  
  .aside {  
    @extend .content;  
    width: 35%;  
  }  
}
```

extending something
in the same
media query is ok

application.css

```
@media (min-width: 700px) {  
  .content,  
  .aside {  
    border: 1px solid #ccc;  
  }  
  .aside {  
    width: 35%;  
  }  
}
```

7.4 Responsive Pitfalls



Matching media queries are not combined:

application.scss

```
.sidebar {  
  @media (min-width: 700px) {  
    width: 50%;  
  }  
}  
.callout {  
  @media (min-width: 700px) {  
    width: 35%;  
  }  
}
```

application.css

```
@media (min-width: 700px) {  
  .sidebar {  
    width: 50%;  
  }  
}  
@media (min-width: 700px) {  
  .callout {  
    width: 35%;  
  }  
}
```

7.4 Responsive Pitfalls



Sometimes, manual combination is best:

application.css

```
@media (min-width: 700px) {  
  .sidebar {  
    width: 50%;  
  }  
  .callout {  
    width: 35%;  
  }  
}
```

7.4 Responsive Pitfalls



ASSEMBLING SASS



PROPERTY OF:
CODE SCHOOL

PROJECT DATE:
SEP. 2012



This project was created and developed by
David E. Johnson, M.Ed., S. M.Ed.
and is licensed by Creative Commons
Attribution-ShareAlike (CC BY-SA).
All rights reserved. No part of this
work may be reproduced or transmitted
in any form or by any means
electronic, mechanical, photocopying,
recording, or by any information
storage and retrieval system, without
prior written permission from the
author or publisher.

S

STANDARD
INSPECTION : NO 18