Extracted from:

Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

The Pragmatic Bookshelf

Raleigh, North Carolina

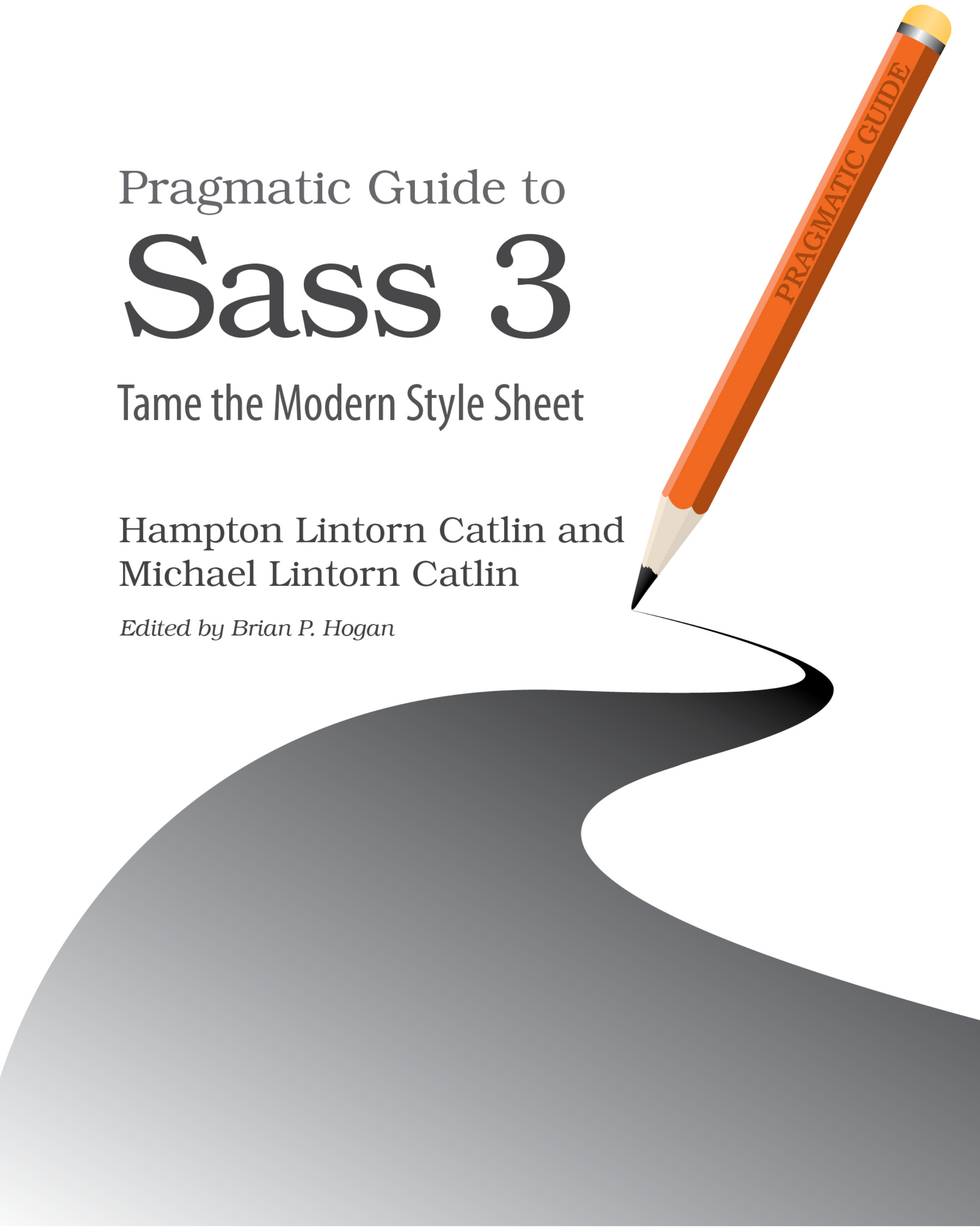
The Pragmatic Programmers

Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

Hampton Lintorn Catlin and
Michael Lintorn Catlin

Edited by Brian P. Hogan



Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

Hampton Catlin
Michael Catlin

The Pragmatic Bookshelf
Raleigh, North Carolina

The team that produced this book includes:

Brian P. Hogan (editor)
Potomac Indexing, LLC (index)
Nicole Abramowitz (copyedit)
Gilson Graphics (layout)
Janet Furlow (producer)

For sales, volume licensing, and support, please contact *support@pragprog.com*.

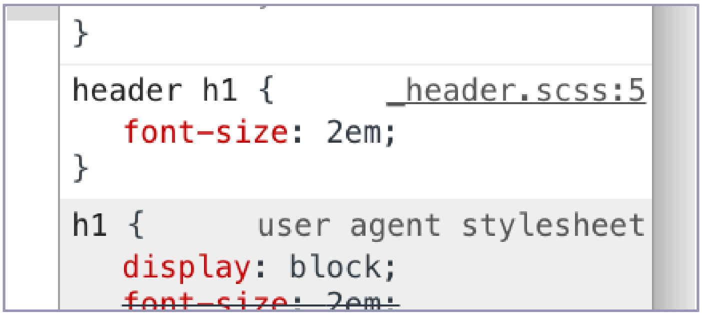
For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-176-6
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—July 2016

29 Debugging Your Sass

Sourcemaps are a feature that allow you to more directly debug your Sass from the browser. Browsers can't "read" Sass—they need it to be compiled into CSS. This means that when you're debugging your site locally, you see only the CSS output rather than the Sass that you've written. Rather than try to figure out which CSS line corresponds to which Sass line, sourcemaps show the exact Sass line in your code.

Sass sourcemaps are enabled by default when you compile your style sheets. When inspecting an element on the page, you can see the Sass file it came from. In the following case, you can see that the `<h1>` styles that are being implemented come from the `_header.scss` partial, on line 5.



```
}
header h1 {        _header.scss:5
   font-size: 2em;
}
h1 {       user agent stylesheet
   display: block;
   ~~font-size: 2em;~~
```

Another useful feature is `@warn`. This function allows you to let others know when there's an error in their code. Take a look at the example. There's a mixin to help with media queries. There's also a map of appropriate screen sizes for our project. The mixin checks that the value (`$size`) is found in the map. If not, the `@warn` function is used.

When someone uses the mixin incorrectly—by passing in a breakpoint that isn't present—in the command line, they will see the warning message and the appropriate line number.

➤ advancedlanguage/atwarn.scss

```
$breakpoints: (small: 400px,
               medium: 700px,
               large: 1200px);

@mixin larger-than($size) {
  @if map-has-key($breakpoints, $size) {
    @media (min-width: #{map-get($breakpoints, $size)}) {
      @content;
    }
  }
  @else {
    @warn "This is not a valid breakpoint.";
  }
}

/////////////////////////////////////////////

.hero-image {
  @include larger-than(extra-large) {
    width: 1400px;
  }
}
```

If you try to compile this Sass, this is generated on the command line:

```
WARNING: This is not a valid breakpoint.
 on line 12 of advancedlanguage/atwarn.scss, in 'larger-than'
 from line 17 of advancedlanguage/atwarn.scss
```

Related Tasks:

- Task 16, *Controlling Flow with @if*, on page ?
- Task 24, *Using Maps for More Detailed Collections*, on page ?