Extracted from:

Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

The Pragmatic Bookshelf

Raleigh, North Carolina

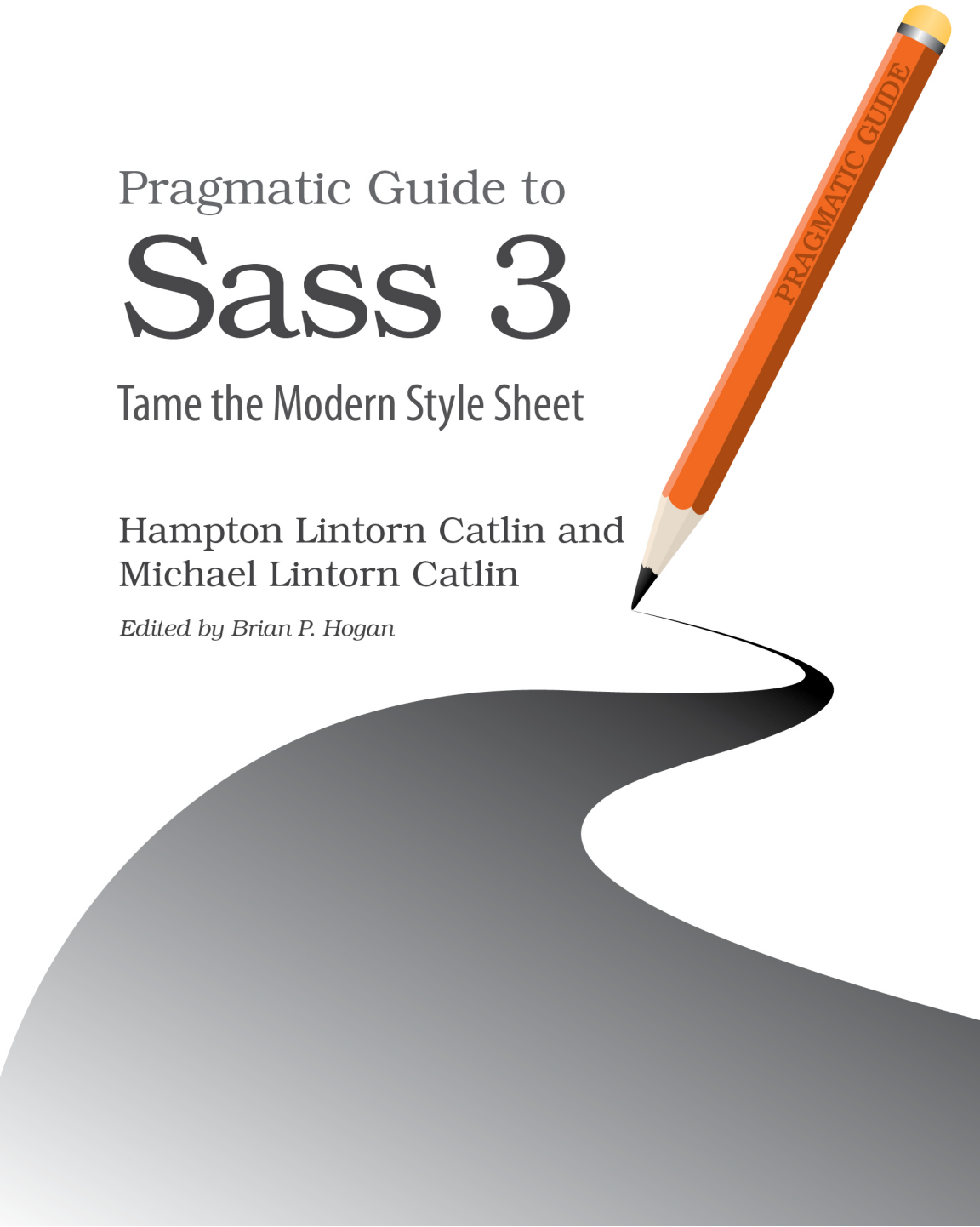
The Pragmatic Programmers

Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

Hampton Lintorn Catlin and
Michael Lintorn Catlin

Edited by Brian P. Hogan

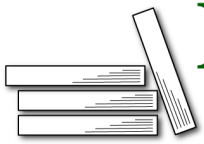


Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

Hampton Catlin
Michael Catlin

The Pragmatic Bookshelf
Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

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

14 Adding Mixin Arguments

When you created mixins in *Keeping Code Clean with Mixins*, you placed all the necessary styles in the mixin itself. However, the true power of mixins comes with using arguments.

Say you have a general notice style that you'd like to use in a mixin. However, sometimes the notice is green, and sometimes it's red. Making two mixins seems a little unnecessary. You can use an argument instead! Instead of putting a predefined background in the mixin block, put `$background` (or whatever you wish to call it) in. Then, when naming the mixin, include the `$background` part in parentheses after the name.

In the example (which uses a much simpler mixin, but you get the idea), see that you can manipulate the arguments you pass in. The background being used is darkened in order to get a border color.

When using the mixin, just pass in the arguments in the order you defined them. You can also specify which argument is which by placing the name in front of the argument—these are called *keyword arguments*. You can see that in the second mixin use in the example. Although this looks a little more wordy, it helps for a couple of reasons. It means that you don't have to remember the exact order of arguments. It also makes your code easier to read. If you have two color-based arguments, for example, using the keyword can help you remember which argument you're referencing.

➤ Create a mixin with arguments.

advancedmixins/create_mixin.scss

```
@mixin notice-box($background, $width) {
  background: $background;
  border: 1px solid darken($background, 20%);
  width: $width;
}
```

➤ Use a mixin that takes arguments.

advancedmixins/use_mixin.scss

```
@import "create_mixin";
.warning {
  @include notice-box(red, 100%);
}
.welcome {
  @include notice-box($width: 300px, $background: green);
}
```

This compiles to:

```
.warning {
  background: red;
  border: 1px solid #990000;
  width: 100%; }

.welcome {
  background: green;
  border: 1px solid #001a00;
  width: 300px; }
```

Related Tasks:

- Task 8, *Keeping Code Clean with Mixins*, on page ?
- Task 15, *Using More Mixin Argument Tricks*, on page ?