Extracted from:

Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

The Pragmatic Bookshelf

Raleigh, North Carolina

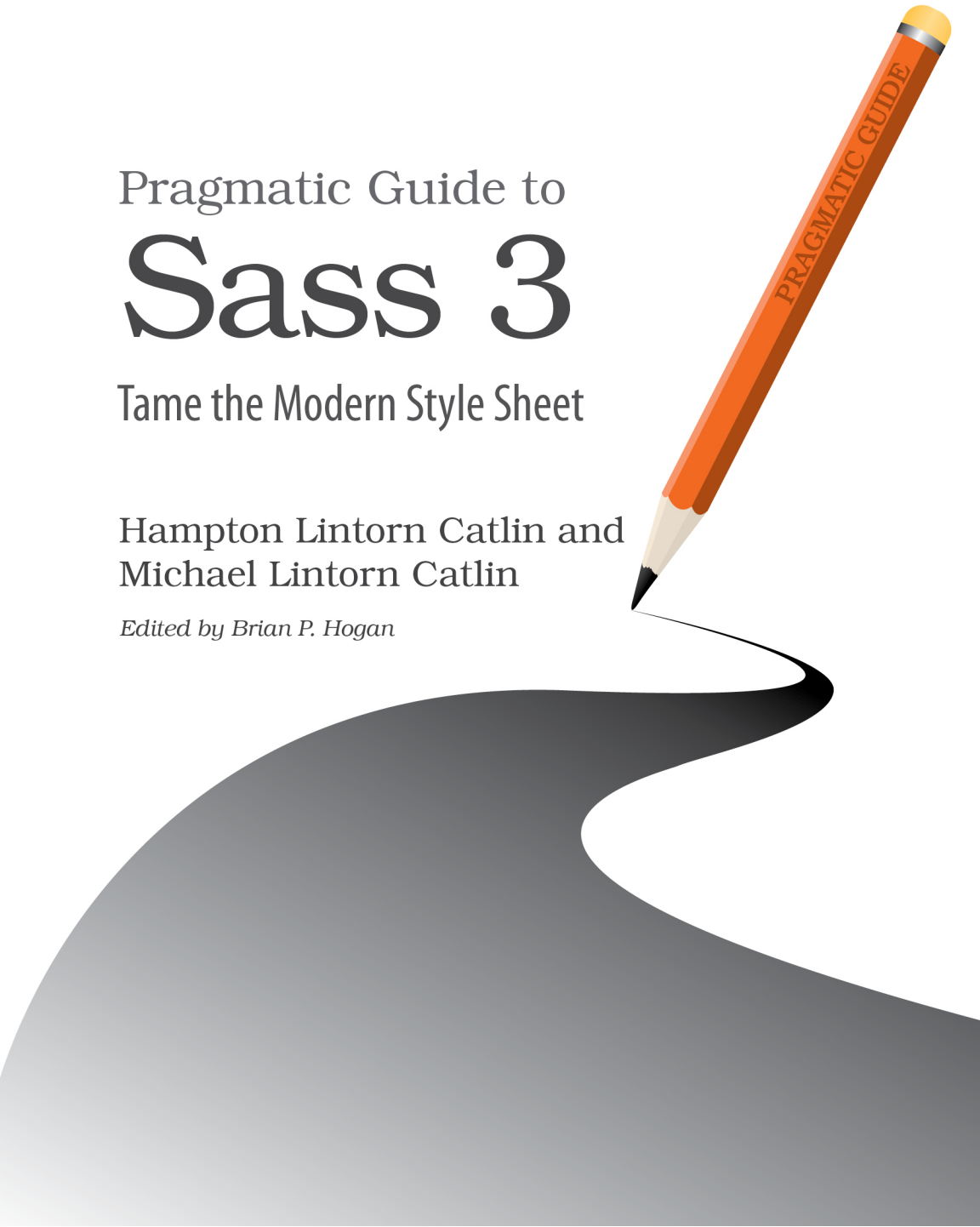
The Pragmatic Programmers

Pragmatic Guide to Sass 3

Tame the Modern Style Sheet

Hampton Lintorn Catlin and
Michael Lintorn Catlin

Edited by Brian P. Hogan

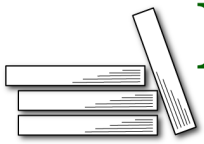


Pragmatic Guide to Sass 3

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

Welcome!

Welcome to the *Pragmatic Guide to Sass 3*. Sass (Syntactically Awesome Style Sheets) enables you to do amazing things with your style sheets and write more efficient, beautiful, and maintainable code.

"What's wrong with regular ol' CSS?" we hear you cry. The fact is that CSS, with all its power and elegance, is not designed for developer productivity. Instead, it's focused on helping the browser quickly display styles as simply as possible. There's nothing wrong with that, but it's not helpful for us as developers. That's where Sass comes in—to *extend* CSS, adding new features and syntax focused on making your life as a developer easier.

Sass isn't a replacement for CSS—it's a way to help us write *better* CSS files, which is essential for large projects. Sass helps us write clear, semantic style sheets. Sass updates our CSS development for the future. It's why our motto is "CSS with Superpowers."

Sass is by far the most popular of the CSS extension languages, according to almost every survey. It's used in almost every major web project and is the basis for most modern open source style frameworks.

Sass was originally conceived in 2006 as a component of the Haml templating language, and it was the first CSS extension language. It was initially envisaged by Hampton Lintorn Catlin, but it was primarily implemented by Natalie Weizenbaum, who has also been primarily responsible for the language's advanced features and serves as the primary language designer today. Chris Eppstein rounds out the language core team, being best known for Compass and Eyeglass, two popular Sass-related projects. We'll talk more

about Eyeglass later. Sass has grown up a lot since that original concept! In fact, there are now multiple implementations of Sass, and it's available on most platforms.

In this book, we'll be using the word Sass as an overarching concept that describes the engine we use to convert our files into CSS. We can use two syntaxes to write Sass—SCSS and Original Sass. We'll describe these a bit later in this preface.

Who Is This Book For?

This book is for people who know the pain of working on the CSS of a mature website—who have faced a CSS file that four people wrote and that mutated into a huge, sprawling, incoherent mess. We've looked the beast in the eye and barely survived.

You're probably already familiar with CSS, HTML, and web development. We'll assume that you're familiar with margins, padding, the box model, `@media` queries, and myriad other CSS-related technologies.

If you're looking for a CSS ninja power-up, you've come to the right place.

Nomenclature and Syntax

Some of the terms associated with CSS can be confusing, so we've added a short introduction to how we name things in the book. Also, we've distinguished between the two different syntaxes for writing Sass.

A Brief CSS Recap

We thought it would be useful to go through a couple of technical terms we'll be using for different aspects of CSS markup. If you're already familiar with selectors, declaration blocks, and the like, you can probably skip this part.

Let's use a small bit of CSS as an example:

```
p {
  color: #336699;
  font-size: 2em;
}
```

Here we have p, which we call the *selector*. What follows (the bit inside the curly braces) is the *declaration block*. The two lines—one defining the color and one defining the font size—are known as *declarations*. Each declaration has a *property* and a *value*. The property in this case is the color or the font size. The value is the color itself—for example, *#336699*, blue—or the size of the font—for example, *2em*.

The use of *classes* and *IDs* allows us to define sets of declarations that will only be applied to specific sections of our HTML. Sass allows us to create much richer selectors, as we'll see in Part I, *Language Basics*, on page ?.

SCSS? Sass? Indented Sass?

You'll see the preceding terms used around the Internet, and it can be a little confusing what they're talking about. "Do you prefer Sass or SCSS?" is a common question you might get asked...and it is actually a nonsensical question! Here's why: Sass was originally an indented language. It has all the same concepts as CSS, but the syntax was more inspired by the Haml language than by CSS.

Later, the Sass team realized that the indented syntax was difficult for new users to grasp, so we created an alternate syntax for Sass. This alternate syntax is a strict subset of CSS, known as *SCSS (Sassy CSS)*. People often get confused and think that SCSS is an entirely different project or language from *Sass*. The truth is that both syntaxes are supported, and both make up the language of *Sass*. When we refer to the original syntax, we refer to it as *indented Sass* to make the distinction clear.

When someone is referring to Sass today, they're most likely referring to either the project itself or a file written in the SCSS syntax. There is usually no need to use the word SCSS unless you're being very specific about the syntax itself...it's all just Sass!

Typically, the extension for indented Sass is .sass, and the extension for SCSS is .scss—furthering the confusion about whether something is considered a Sass file or not. Both of those file extensions are considered Sass extensions, and they are how we tell the Sass compiler what syntax to expect.

We do plan on continuing to support the indented syntax into the future, and you can mix and match syntaxes across projects. In this book, we won't be using indented Sass at all. The indented syntax is far less common now. However, if you're interested in learning about indented Sass, a lot of good resources are available online for you to browse. All the code in this book will still work the same way, just with syntax variations.

Here's a quick summary of the main differences between the two syntaxes. Indented Sass has no parentheses or semicolons, whereas the SCSS syntax does. Here's a quick example comparing the two.

Indented syntax:

```
.modal
  color: #336699
  font-size: 2em
  a
    text-decoration: none
```

SCSS syntax:

```
.modal {
  color: #336699;
  font-size: 2em;
  a {
    text-decoration: none;
  }
}
```

The SCSS syntax, while a little more verbose, usually feels a lot more familiar to coders who are used to CSS syntax. That's why we're using it for our code examples in this book.

Overview

In Part I, *Language Basics*, on page ?, we'll take you through the first things you'll need to know about Sass, like how to install (Task 1, *Installing Sass*, on page ?). You'll also read about some of the core features of Sass, such as nesting (Task 2, *Scoping Selectors with Nesting*, on page ?), variables (Task 7, *Defining Variables*, on page ?), and mixins (Task 8, *Keeping Code Clean with Mixins*, on page ?).

In Part II, *Simple Use Cases*, on page ?, you'll learn about some real-world applications of Sass, such as better @media handling (Task 12, *Using Media Queries*, on page ?) and creating color themes (Task 10, *Creating Themes with Advanced Colors*, on page ?).

You'll dive deeper into mixins in Part III, *Advanced Mixins*, on page ?, looking at passing in arguments (Task 14, *Adding Mixin Arguments*, on page ?) and gating logic (Task 16, *Controlling Flow with @if*, on page ?).

In Part IV, *Values in Sass*, on page ?, things will get a little more technical. To grasp some of Sass's most advanced features, you'll learn about value types (Task 19, *Understanding Value Types in Sass*, on page ?). Then you can really get to grips with lists (Task 21, *Using Lists to Work with Multiple Properties*, on page ?) and maps (Task 24, *Using Maps for More Detailed Collections*, on page ?) in Sass.

In Part V, *Advanced Language Features*, on page ?, you'll learn about advanced features of Sass. In particular, you'll see how to use @extend (Task 31, *Using @extend as a Mixin Alternative*, on page ?), but you'll also see how to create functions (Task 28, *Creating Your Own Functions*, on page ?) and use the @at-root function (Task 34, *Escaping Indentation with @root*, on page ?).

Finally, in Part VI, *Libraries and Frameworks*, on page ?, you'll learn that sometimes it's more productive to look to libraries for help in creating a well-rounded website. You'll learn about some grid frameworks (Task 36, *Using Grid Systems for Layout*, on page ?) and Eyeglass (Task 37, *Introducing Eyeglass*, on page ?), a modular system to extend Sass's features.

How to Read This Book

The book is arranged into tasks. These are short snippets of information. You'll find a description of the task and on the next page will be the code you need to write to get results.

We've tried to arrange the book to go from basic to advanced tasks. However, you can definitely dip in and out of the book if you find a specific task you need to look at. Once you've

grasped the basics (such as installing), you should be set to do most of the tasks in the book.

Getting Help

There are several ways you can find help for your Sass troubles; Sass has a warm and welcoming community, and we love helping people. For example, Stack Overflow is a great place to ask questions and find answers.[1] The Sass documentation has a wealth of information that covers most of what we look at in this guide and even goes over a few other things as well.[2]

In addition, if you ever need help with the `sass` command, just type `sass --help`, and Sass will let you know about all the available ways to run it.

A Few Final Comments

We're almost ready to start, but here are some little bits that you'll find useful to know before we dive into the book.

- We'll be using the following phrase to show when we've converted some Sass into CSS:

  This compiles to:

  Hopefully, you'll be familiar with the CSS output, so you can see how much simpler Sass is compared to CSS.

- If you've downloaded the ebook, you'll notice that all the code samples are preceded by a little shaded box. If you click the box, the code sample shown in the book will be downloaded to your computer, allowing you to play around with our examples.

- You can get more information from the book's official web page.[3] There you'll find resources such as the book forum, code downloads, and any errata.

OK—now that we've got all that out of the way, are you ready to get Sassy?

1. http://stackoverflow.com/questions/tagged/sass
2. http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html
3. http://pragprog.com/book/pg_sass/pragmatic-guide-to-sass