Extracted from:

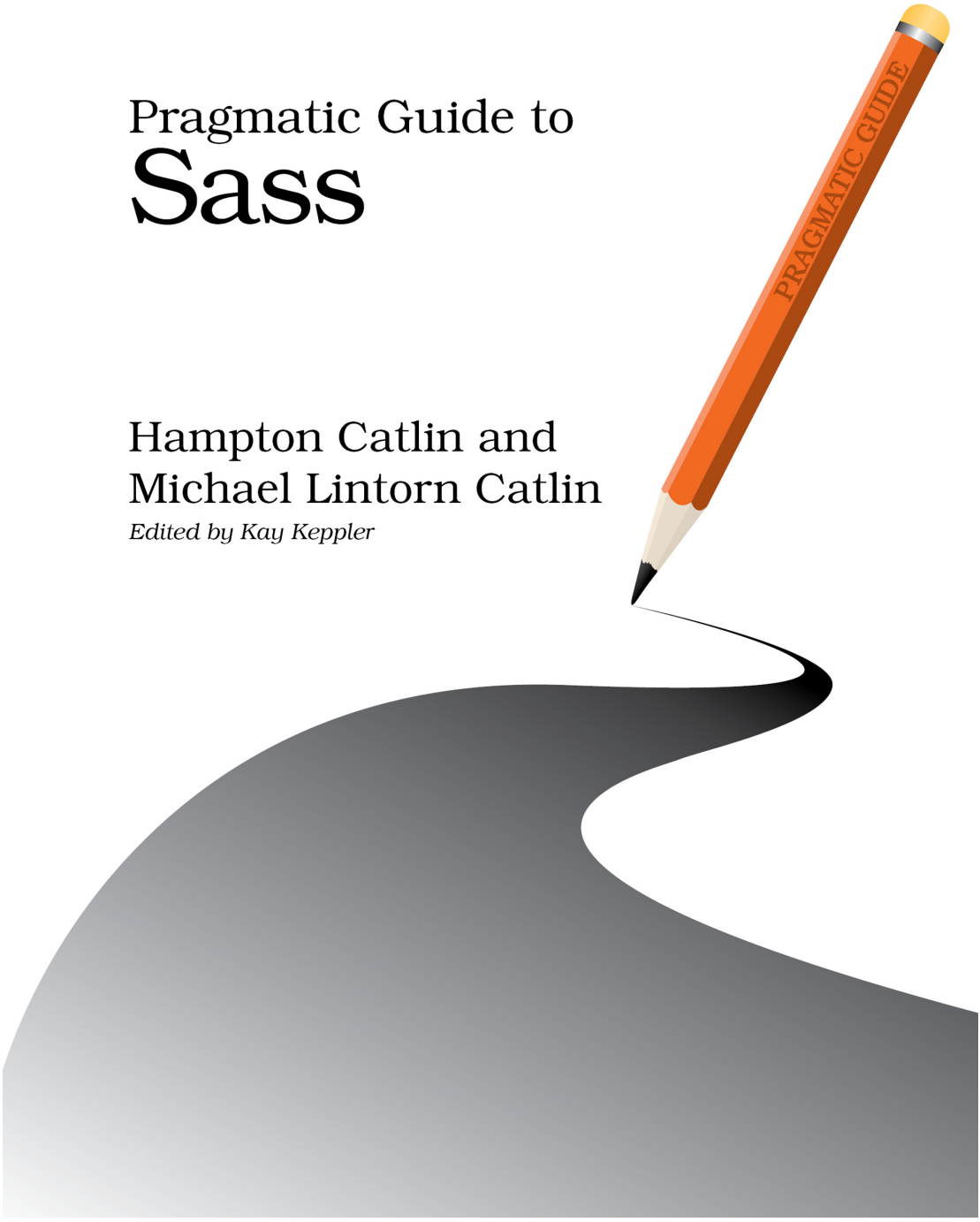
Pragmatic Guide to Sass

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic Guide to Sass



Hampton Catlin and
Michael Lintorn Catlin

Edited by Kay Keppler

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at *http://pragprog.com*.

The team that produced this book includes:

Kay Keppler (editor)
Potomac Indexing, LLC (indexer)
Molly McBeath (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-934356-84-5
Printed on acid-free paper.
Book version: P1.0—December 2011

Welcome to the *Pragmatic Guide to Sass*. Sass (Syntactically Awesome Style Sheets) enables you to do amazing things with your style sheets, helping you describe how HTML is laid out on a web page. Sass is an alternative way of writing CSS.

"What's wrong with regular ol' CSS?" we hear you cry. The fact is that CSS, with all its power and elegance, is missing some crucial, simple elements that other types of development take for granted. CSS can also be a bit complicated to read: Sass fixes that.

Most programmers are familiar with the concept of *DRY*—*D*on't *R*epeat *Y*ourself. It saves time and effort when writing code. A core philosophy of Sass is to reduce repetition in style sheets, and we'll be coming back to DRY a few times throughout the guide.

Sass isn't really 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 CSS development for the future.

Hampton originally designed Sass while he was working at Unspace in Toronto, and Nathan Weizenbaum and Chris Eppstein now maintain it. A lot of Sass functionality depends on Ruby. (But don't worry, we'll learn how to install Ruby in Part I, *Basics*, on page ?.)

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. These will be described 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 the ideals of semantic web development. We can all agree that markup should be about logic instead of about presentation

(as much as possible). And we'll assume that you're familiar with margins, padding, the box model, @media queries, and the myriad of other CSS-related technologies.

If you are 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 quite confusing, so we've added a short introduction to how we name things in the book. Also, there are two different syntaxes for writing Sass that need to be distinguished.

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, *20px*.

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 you to create much richer selectors, as we'll see in Part I, *Basics*, on page ?.

SCSS: A More CSS-like Way to Write Sass

SCSS, which stands for *S*assy CSS, is one of the syntaxes we use to write Sass. The grand aim of SCSS is to keep the look of CSS while introducing the units of Sass. If you're familiar with CSS, it's pretty easy to read. We still use selectors,

classes, and IDs. We open a curly brace to start the declaration block, and we separate out declarations with semicolons. What's extra is the added functionality.

When we use the word *Sass*, we'll mostly be referring to the SCSS syntax.

Original Sass: A Stripped-down Way to Write Sass

Before SCSS, there was Original Sass, which strips out some of the unnecessary elements of CSS and SCSS. Original Sass can be compiled just the same as SCSS, via the Sass engine.

A great example of unnecessary elements are curly braces. Look at this:

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

We know by the use of . or # that something is a selector. Using *whitespace* (two spaces or a soft tab that indents the properties) helps us. In the example above, the indentation lets us know that `color` and `font-size` refer only to the `fab_text` class. The curly braces aren't needed. Why not just strip them out?

```
.fab_text
  color: #336699;
  font-size: 2em;
```

Look at that! Doesn't the code already look a lot cleaner, a lot simpler?

While we're at it, we might as well take away the semicolons at the end of the values. They don't add much, do they?

```
.fab_text
  color: #336699
  font-size: 2em
```

And this is how Original Sass is written. As you can see, it's more different from CSS than from SCSS, as it involves removing bits we're used to. So in the examples we use in the book, we'll mostly be using SCSS to describe things. Once you're used to it, though, Original Sass should be more readable at a quick glance.

Aside from the curly braces and semicolons, most of the features we'll look at are written the same in both SCSS and Original Sass. When they're not, we'll point out how they differ. It's really up to you whether you use SCSS or Original Sass syntax.

Overview

In Part I, *Basics*, on page ?, we'll take you through the very first things you'll need to know about Sass and SCSS, like how to install (Task 1, *Installing Sass*, on page ?). We'll also take you through variables, where Sass gets really exciting (Task 9, *Defining Variables*, on page ?).

We'll take things to the next level in Part II, *Advanced*, on page ?. One of the main things we'll look at is mixins (Task 16, *Keeping Code Clean with Mixins*, on page ?). We'll also take a look at some more programmer-style functions of Sass, such as `@each` and `@if` (in Task 22, *Stop Repeating Yourself with @each*, on page ?, and Task 23, *Determining Conditions with @if*, on page ?, respectively).

Chris Eppstein's Compass is a great way to style pages, and we'll go through it in Part III, *Compass*, on page ?. We'll cover things like adding columns to your text (Task 33, *Jazzing Up Layouts with Columns*, on page ?) and making a sticky footer (Task 29, *Sticking a Footer to a Window*, on page ?).

In Part IV, *Blueprint CSS*, on page ?, we'll look at a framework that makes things even simpler than Compass. Among other things, it provides a great predefined structure to help you customize buttons, which we describe in Task 37, *Making Beautiful Buttons*, on page ?.

How to Read This Book

The book is arranged into tasks. These are short snippets of information. On the left you'll find a description of the task at hand. On the right you'll find the code you need to write to get results.

We've tried to arrange the book to go from the most basic tasks to the most advanced. 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 very basics (such as installing), you'll probably 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. For example, join the Sass Lang Google group.[1] Also, 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 probably 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 more familiar with the CSS output, so you can easily compare 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 on the box, the code sample shown in the book

1. http://groups.google.com/group/sass-lang
2. http://sass-lang.com/docs/yardoc/file.SASS_REFERENCE.html

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

3. http://pragprog.com/book/pg_sass/pragmatic-guide-to-sass