THE EXPERT'S VOICE® IN WEB DEVELOPMENT

Beginning CSS Preprocessors

With Sass, Compass, and Less

—

Anirudh Prabhu

Apress®

Beginning CSS Preprocessors

With Sass, Compass, and Less

Anirudh Prabhu

Apress®

Beginning CSS Preprocessors: With SASS, Compass.js, and Less.js

Copyright © 2015 by Anirudh Prabhu

ISBN-13 (pbk): 978-1-4842-1348-3

ISBN-13 (electronic): 978-1-4842-1347-6

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this text is available to readers at www.apress.com/. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Contents at a Glance

Contents

About the Author

Anirudh Prabhu is a software engineer with over six years of industry experience. He specializes in technologies such as HTML5, CSS3, PHP, jQuery, Twitter Bootstrap, Less, and Sass, and he also has knowledge of CoffeeScript and AngularJS. In addition to web development, he has been involved in building training materials and writing tutorials for these technologies.

About the Technical Reviewer

Lokesh Iyer completed his bachelor's in IT from KC College, Mumbai and received his MBA from Sydenham College, Mumbai. He is the founder and director of SI Technologies, a company focused on providing hardware and software solutions as well as web exposure and security solutions. Over the past three years, he completed over 60 projects on HTML5, CSS3, JavaScript, jQuery, PHP, C#, Android, and MySQL/SQLite databases with his team. Apart from his business ventures, he is a visiting faculty member at KC College of the bachelor's in IT program.

Acknowledgments

First and foremost, I wish thank the awesome team at Apress for offering me such a wonderful opportunity to write this book. When Celestin “discovered” me through LinkedIn and asked me if I would like to write a book, it sounded like a straight and easy task. Over the next month, he painstakingly guided me through the entire process of preparing an initial proposal for the book and helped me finalize it. Subsequently, when the real action started in terms of writing the chapters, Rita was always there. She was the scrum master who was always there to help. She gave me that gentle nudge to make sure that even as I was running behind schedule, I did everything that needed to be done to catch up and deliver the chapter in potentially shippable increments. Thanks, Celestin and Rita, for not giving up on me!

I also want to thanks Matt Moodie and the large team from Apress working in background, for all their efforts. In addition, the review feedback and critical inputs by reviewers is an author’s lifeline—it is the first feedback on a product that is still quite raw. I want to offer my sincere thanks to Lokesh Iyer for his technical review.

I can’t thank my professional network enough for enriching my learning journey through the years—my former employers, my clients, managers, colleagues, team members, students, readers of my blog, audience to my talks, and the noblest of them all—the fellow volunteers. Thanks for all the support and learning opportunities and for making me a better professional every single day.

Finally, despite all the diligent efforts of the editorial team and reviewers, I must accept responsibility for all the mistakes and shortcomings in this book. Let me know how I can make this book better.

Introduction

CSS preprocessor came into buzz a couple of years ago. The concept intrigued me: Allowing use of preprocessor files that could contain one or several things like variables, functions, mixins, and the like. After development, these special files would then be compiled into regular CSS files that all web browsers could understand.

Being a believer of phrase “if it's not broken, don’t fix it,” I avoided using CSS preprocessors. The initial thought that would come to my mind was, “Why add unnecessary processes to my workflow?”. Also, I feared the steep learning curve and the command-line interface provided me another reason to avoid CSS preprocessors.

Finally, after watching several podcasts and reading through many articles, I had an “a-ha” moment. It made me realize that, “Wow, I should be incorporating this in my workflow!”.

Since then, I've been using Sass and Less in my projects, and it seems to have made my development a lot simpler and more efficient. In this book, you will learn how both of these preprocessors work.

You’ll first start by learning about the concept of preprocessors and how they work. You also learn about the popular flavors of preprocessors available on the market. You then look into the GUI-based tools available for people who are not familiar with command-line interfaces.

As the chapters progress, you will learn all about these two preprocessors—Sass and Less—and learn about a popular framework based on Sass called Compass.

The knowledge shared in this book can help you improve your productivity and write maintainable and scalable CSS code.

CHAPTER 1

Introduction to Preprocessors

HTML5 and CSS3 are changing how web pages are designed. CSS3 provided web developers with advanced features such as gradients, transitions, and animations, etc. However, these new features increased the complexity of CSS code, thus making it more difficult to maintain.

Besides the complexity introduced by CSS3, writing CSS may turn painful with time, because programmers have to perform many of the same activities over and over again (such as having to look up color values in CSS and margin/padding declarations). These small repetitive tasks add up to quite a bit of inefficiency. Preprocessors are the solution to these, and a handful of other, inefficiencies.

CSS preprocessors extend CSS with modern programming-language concepts. In order to use Sass (Syntactically Awesome Stylesheets), you must know how to code in CSS. CSS preprocessors allow you to use variables, functions, operations, and even rule or selector nesting while coding your CSS. With CSS preprocessors, you can apply the “Don’t Repeat Yourself” (DRY) principle to your CSS code. Following the DRY principle helps you avoid code repetition.

What Are Preprocessors?

A preprocessor takes one form of data and converts it to another. In the context of CSS, Less and Sass are popular preprocessor languages, and they take input in the Less or SCSS format and produce processed CSS.

These CSS preprocessors empower CSS by removing the inefficiencies and making web sites easier and more logical to build. The increase in popularity of preprocessors led to the rise of different frameworks based on them; one of the more popular is Compass.

Figure 1-1 shows how a preprocessor takes a preprocessor-formatted file and translates it to CSS that the browser understands.

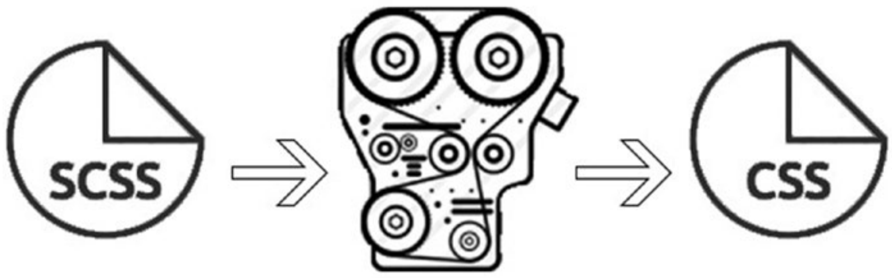

Figure 1-1. *Preprocessor-friendly file being translated to normal CSS*

With a preprocessor, you can structure CSS similar to other languages like PHP or JavaScript. Thus, a preprocessor brings peace of mind to the developer. It lets you write code that's future-proof and easily maintainable, thus saving time and energy.

Preprocessors are extensions of CSS, which means that valid CSS code is valid preprocessor code. Developers familiar with CSS won't have a learning curve while learning any preprocessor.

Why Use Preprocessors?

CSS uses a declarative form of programming. This means that the styles that you write in the code are used directly by the browser, without any compiling.

Many developers prefer to write stylesheets by hand. They believe that preprocessors would add extra complexity to their workflow or would have a steep learning curve. But in reality, CSS preprocessors make your daily work a lot easier. This book shows you how preprocessors can be more efficient for writing CSS without disturbing your workflow.

Let's consider an example where you need to replace multiple instances of a color that's used sitewide by finding it one instance at a time. Wouldn't it be great if CSS could simplify this process? Something like Listing 1-1 would be an example.

Listing 1-1. Reusable Variable for Color

```
$site-color:#eee;

a {
        color: $site-color;
}
#topBar {
        background-color: $site-color;
        color:#fff;
}
```

With a preprocessor, changing a value in one place changes the entire stylesheet. This is shown in Listing 1-2.

Listing 1-2. Output of Listing 1-1

```
a {
  color: #eee;
}

#topBar {
  background-color: #eee;
  color: #fff;
}
```

Let's consider another example of code repetition. Many times there are blocks of code used at various locations in your stylesheet, as shown in Listing 1-3.

Listing 1-3. Repeated Code Block

```
p{
        padding-bottom:45px;
        text-align:center;
}
footer {
        padding-bottom:45px;
        text-align:center;
}
```

With preprocessors, you can put these redundant rules into a *mixin*, which is defined once and can be included as needed. This is shown in Listing 1-4.

Listing 1-4. Creating and Using a Reusable Code Block in the Preprocessor

```
@mixin containerSettings {
        padding-bottom: 45px;
        text-align:center;
}
p {
        @include containerSettings;
}
footer {
        @include containerSettings;
}
```

Listing 1-5. Output of Listing 1-4

```
p {
  padding-bottom: 45px;
  text-align: center;
}

footer {
  padding-bottom: 45px;
  text-align: center;
}
```

CSS, which is the foundation of all preprocessors, has a steep learning curve when it comes to understanding how different properties work, understanding cascading, browser support for various properties, the selectors, the quirks, and so forth. In addition to all the previous points, consider that maintaining stylesheets in today's complex interfaces is a big challenge too.

Most of the time, stylesheets are immensely repetitive, with properties or groupings of properties, etc. The typical CSS file is a linear document. This makes a programmer from an object-oriented domain go crazy.

As per the DRY principle: *Every piece of knowledge must have a single, unambiguous, authoritative representation in a system.*