

Extracted from:

Pragmatic Guide to Sass

This PDF file contains pages extracted from *Pragmatic Guide to Sass*, published by the Pragmatic Bookshelf. For more information or to purchase a paperback or PDF copy, please visit <http://www.pragprog.com>.

Note: This extract contains some colored text (particularly in code listing). This is available only in online versions of the books. The printed versions are black and white. Pagination might vary between the online and printer versions; the content is otherwise identical.

Copyright © 2010 The Pragmatic Programmers, LLC.

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

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





Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf, PragProg and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

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)

Copyright © 2011 The Pragmatic Programmers, LLC.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

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

Let's look at a core feature of Sass: nesting. If you've been working with CSS for a long time, you know the advantages of giving more specific selectors to your style sheets. Using `.sidebar p em` allows you greater specificity to the `em` element versus a standalone `em` selector. It gives you more freedom with reusing names and making your HTML more semantic and readable. We generally refer to this as *scoping*.

It's a good thing to scope, except it's not DRY. (Remember *Don't Repeat Yourself?*). We keep having to repeat our classes or IDs—for example, repeating an apply-to-all class like `.infobox`—on every line. Typing this by hand is laborious and makes us want to be lazy. When writing CSS, scoping can be very tedious. It involves a lot of copying and pasting. What's more, keeping track of parent-child relationships is tough. We can do better than that! Technology should support good behaviors. Sass is here to help us with *nesting*.

We can put a style such as a border color *inside* a declaration block, and Sass will automatically do the repetitive part for you when you generate CSS. I bet your fingers are thanking you already for saving all that typing. Cool, huh?

A small note: the CSS that's compiled in the example opposite looks a bit funny, doesn't it? Especially when we compare it to the original (repetitive) CSS example we wrote out. What happens is that the Sass engine keeps the indentation when it converts to CSS. All it does is insert the missing selectors.

► Look at this scoped CSS.

Look how much repetition there is in this file. Holy cow!

Download `basics/scoping.css`

```
.infobox          { width: 200px; }
.infobox .message { border: 1px solid red; }
.infobox .message .title { color: red; }
.infobox .user    { border: 2px solid black; }
.infobox .user .title { color: black; }
```

► See it in Sass.

Instead of repeating it, just nest it inside the parent selector.

Download `basics/example_nesting.scss`

```
.infobox {
  width: 200px;
  .message {
    border: 1px solid red;
    .title {
      color: red; } }
  .user {
    border: 2px solid black;
    .title {
      color: black; } } }
```

This compiles to:

```
.infobox {
  width: 200px; }
.infobox .message {
  border: 1px solid red; }
  .infobox .message .title {
    color: red; }
.infobox .user {
  border: 2px solid black; }
  .infobox .user .title {
    color: black; }
```