

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

Included in Sass are some programmer-style functions, which we'll look over in the next couple of tasks. We generally refer to these as *SassScripts*.

Let's start out with a general SassScript that allows you to dynamically generate style sheets. It's called *interpolation*. Oh, fancy sounding word—how we love you! It makes us sound smart just by saying it. You try it: *interpolation*. Feels good, doesn't it? OK, sorry—we got a bit distracted there.

Interpolation basically means “put this there.” Imagine we want to write a mixin that has a dynamic property or selector. And we don't mean a dynamic property value—that's easy stuff that we've already done. We mean if the very *name* of a property or selector could be dynamically generated. Well, you're in luck, because that's exactly what interpolation can do.

Just wrap the name of a variable in `#{ }` and you are done. For example, we could have `#{$myvar}`. The variable will be printed out wherever you put that. So, we could say `.red_#{ $carname }`. And, if `$carname` is set to `volvo`, it would generate the selector `.red_volvo`. Whabam! Victory!

You can pretty much use interpolation anywhere you want in your Sass files. Go crazy!

► Interpolate to create a dynamic selector.

Download [advanced/interpolation.scss](#)

```
@mixin car_make($car_make, $car_color) {
  // Set the $car_make with "_make" at the end as a class
  .car.#{$car_make_make} {
    color: $car_color;
    width: 100px;
    .image {
      background: url("images/#{$car_make}/#{$car_color}.png");
    }
  }
}
```

```
@include car_make("volvo", "green");
@include car_make("corvette", "red" );
@include car_make("bmw", "black");
```

This compiles to:

```
.car.volvo_make {
  color: "green";
  width: 100px; }
.car.volvo_make .image {
  background: url("images/volvo/green.png"); }

.car.corvette_make {
  color: "red";
  width: 100px; }
.car.corvette_make .image {
  background: url("images/corvette/red.png"); }

.car.bmw_make {
  color: "black";
  width: 100px; }
.car.bmw_make .image {
  background: url("images/bmw/black.png"); }
```