

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

Keeping things semantic is a philosophy where everything is named logically. We name items based on what they *do*, not what they look like. We don't want to name something `.blue_button`; we want to name it `.checkout_button`, which is far more useful when we're going through the code.

But what if you had a set of attributes—say a blue button—that needed to be applied to multiple buttons with different functions? You want to name the buttons after their function, but it would be a pain typing out the set of attributes over and over again.

This is where `@extend` comes in. `@extend` clones the attributes from one class or ID and adds them to another. Let's run with the example we had with the blue button. Say we want to use the blue button style for the checkout button. If we've defined the blue button class elsewhere, all we need to do is use `@extend`, followed by the `.blue_button` class in the declaration of your selector.

You'll notice that the CSS output has two selectors. What `@extend` does is merge all the properties and values from both selectors, with a list of selectors merged before the declaration block.

We can also tweak the style being copied. What if we needed the checkout button to be slightly darker than the regular blue button? We can just add those properties we need to change onto the end of the declaration block. The new attributes you add will override the old ones.

This saves us so much time when we're coding. There's far less copying and pasting: you'll barely ever use `Ctrl+C` again.

► Use @extend in a selector.

First we make sure we've described the class elsewhere:

Download [advanced/atextend\\_blueButton.scss](#)

```
.blue_button {  
  background: #336699;  
  font-weight: bold;  
  color: white;  
  padding: 5px; }
```

Then we can @extend the class to another:

Download [advanced/atextend\\_use.scss](#)

```
.checkout_button {  
  @extend .blue_button }
```

This compiles to:

```
.blue_button, .checkout_button {  
  background: #336699;  
  font-weight: bold;  
  color: white;  
  padding: 5px; }
```

► Modify a selector.

Download [advanced/atextend\\_use\\_modified.scss](#)

```
.checkout_button {  
  @extend .blue_button;  
  color: darken(#336699, 10%); }
```