Extracted from:

Dart 1 for Everyone

Fast, Flexible, Structured Code for the Modern Web

This PDF file contains pages extracted from *Dart 1 for Everyone*, 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 printed versions; the content is otherwise identical.

Copyright © 2014 The Pragmatic Programmers, LLC.

All rights reserved.

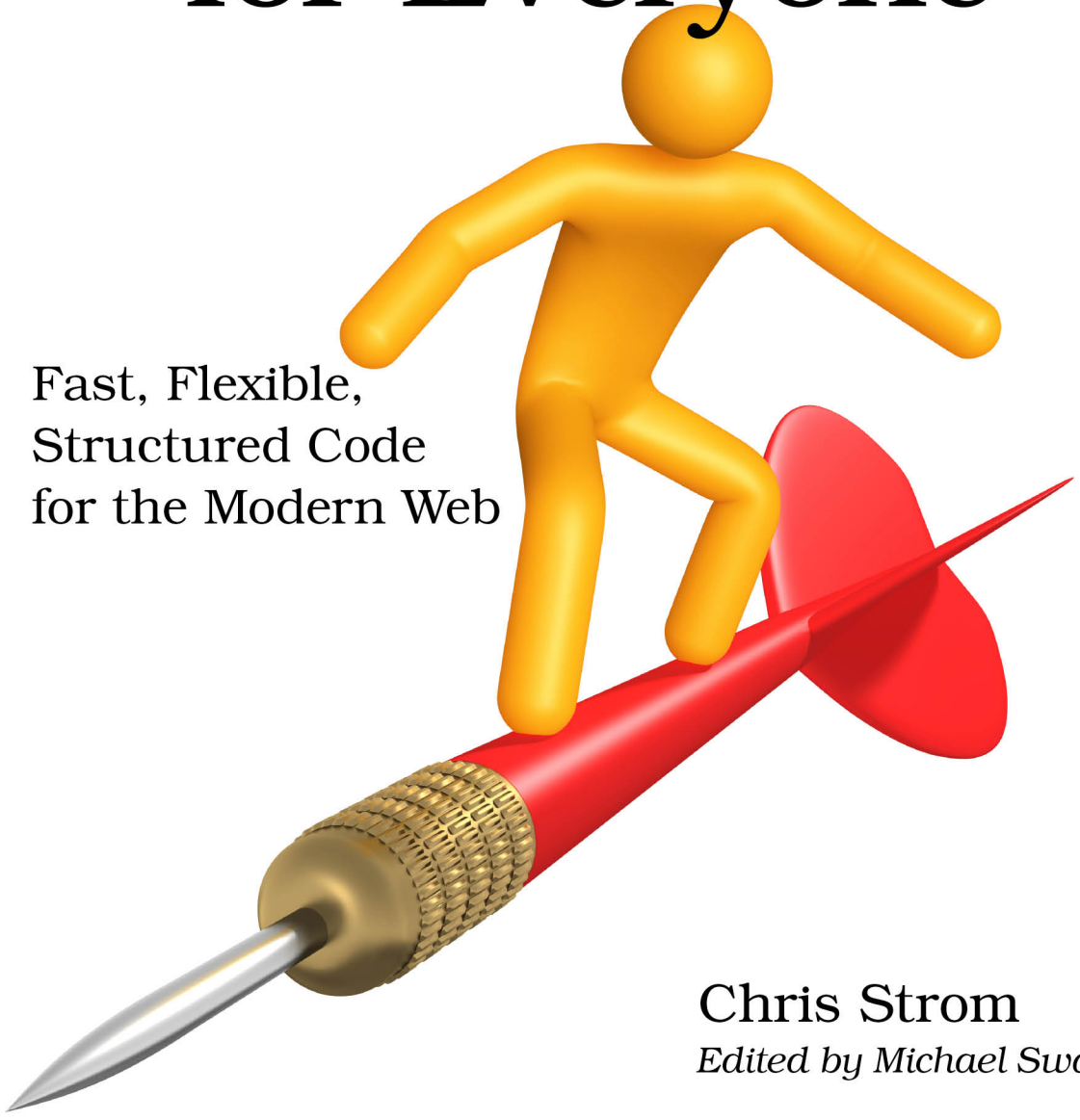
No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Dart 1 for Everyone



Fast, Flexible, Structured Code for the Modern Web

Chris Strom

Edited by Michael Swaine

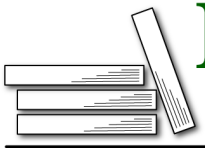
Dart 1 for Everyone

Fast, Flexible, Structured Code for the Modern Web

Chris Strom

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina



Pragmatic Bookshelf

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:

Michael Swaine (editor)
Potomac Indexing, LLC (indexer)
Liz Welch (copyeditor)
Dave Thomas (typesetter)
Janet Furlow (producer)
Ellie Callahan (support)

For international rights, please contact *rights@pragprog.com*.

Copyright © 2014 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: 978-1-941222-25-6
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—October 2014

Introduction

Why Dart?

Looking back more than two years to the first edition of this book (really? that long?), I realize that my answer to "why Dart?" has changed. A lot.

I am very much a language, library, and protocol hipster. I love learning new stuff. The fun of learning is very much its own reward, but that is not the main reason that I learn. It is always nice to add another tool to the developer's proverbial toolbox, but I have come to realize even that is not the main driving force to learning.

The main reason that I learn new things like Dart is that I find it to be a most effective way to understand problems from the perspective of others. The folks who write libraries and languages are invariably smarter and/or know the subject matter better than I do, so learning the solutions is a crash course on understanding the problems that currently exist. Even if I cannot use a shiny new tool right away, the challenges faced and overcome by tool authors stick with me.

So, when I first approached Dart, it was very much in that spirit. And wow, did it ever work! I learned a ton from the language and from interacting with people on the mailing list and in blogs. It profoundly affected my way of thinking—not only about languages like JavaScript (which I still love)—but about programming for the Web in general.

That would have been my answer from two years ago. I learn Dart—and I think others should learn it too—because it profoundly affects the way that one thinks about programming for the Web.

But since then…

I have really come to appreciate the intrinsic beauty and value of Dart. More and more, it is my "go to" tool to fulfill important needs. And I love using it.

• Click HERE to purchase this book now. discuss

Dart is not a JavaScript replacement. Let me repeat that: Dart is not meant to replace JavaScript. It is much more. It took me a long while to understand this, but Dart is not a competitor with JavaScript-the-language. It is a competitor with the platform that supports developing JavaScript. And Dart is so far ahead of JavaScript-the-platform that I would consider it a programming mortal sin to use JavaScript when Dart is an option.

I do not hate JavaScript-the-language. Far from it, in fact. Since the first edition of this book, I wrote a *kids* programming book that uses JavaScript to introduce programming to kids. I am not even a closet JavaScript hater who prefers to use "prettier" compile-to-JavaScript languages like CoffeeScript or TypeScript. Even though each of those compile-to-JavaScript languages offers some benefits over pure JavaScript, I like JavaScript enough to code it by itself.

What I do not care for in JavaScript-land is…everything else. Manipulating the DOM is a pain, so we must have a library for that. `XMLHttpRequest` is ugly, so we need a library for that. Testing is not built in, so a library is needed for that (and test runners and test assertions too!). And a library to manage large codebases. And, of course, we need a library to manage all the libraries. And (some of us) need a library to make JavaScript prettier. And don't even get me started on cross-browser testing, which never does seem to go as planned…

I dislike JavaScript-the-platform because I spend so much time building and maintaining it that I do not focus on what is important: building insanely great applications. With Dart, a consistently beautiful DOM library is built in. `HttpRequest`, which comes standard with Dart, makes Ajax a breeze. Libraries and packages are included from the outset. Testing is beautiful and easy.

Most importantly, Dart compiles to *cross-browser* JavaScript. Unlike every other compile-to-JavaScript language out there, Dart does not generate JavaScript and hope for the best. It targets a Dart-specific JavaScript compatibility layer that ensures that your crazy animations, visualizations, and effects work everywhere across the modern Web. (Note: The modern Web has a specific meaning for Dart. It is the latest two versions of each of the major browsers: Chrome, Firefox, Safari, Opera, and Internet Explorer. The main reason—some would argue the only reason—*not* to use Dart is when support for old versions of Internet Explorer—9 and lower—is a hard requirement.)

Oh, and did I mention that Dart is a pretty language? Well, it is and it lets you write beautiful, maintainable code.

So my answer to "Why Dart?" has changed. Dart is the ideal language *and* platform for development on the modern Web. It will still profoundly affect

• Click HERE to purchase this book now. discuss

how you think. But today it is more. It is the best way to create and maintain insanely great applications on the modern Web.

Who Should Read This Book (Besides Hipsters)?

This book is primarily intended for programmers eager to build rock-solid, cross-browser web applications. JavaScript was originally meant to enhance web pages. It is a testament to Brendan Eich, the inventor of JavaScript, and the resourcefulness of those who followed that JavaScript means so much to the Web. But JavaScript is not built to facilitate rock-solid web applications. Dart has everything needed and more: real modules for organizing large codebases, central package management to enable sharing, cross-browser compatibility, and built-in beautiful testing.

I am also writing for any developer looking to keep their JavaScript skills as fresh as possible. The best way to improve JavaScript skills is through practice and reading other people's code. But sometimes it can radically help to see what the competition is up to. In this case, as we explore what Dart brings to the table, we can better understand the gaps in an admittedly wonderful language.

This book should be of interest to developers learning languages for the sake of learning. I focus quite a bit on the Dart language, especially in those places that it surprised and delighted me.

And of course, hipsters should read this book as well. Dart is just different enough to make it intriguing to the typical language hipster and yet powerful enough to make it worthwhile for the hipster who hopes to change the world.

How the Book Is Organized

I am trying something different with this book. Rather than introducing slices of the language in each chapter, I bite off chunks. Each section starts with an actual Dart project, including some commentary on the choices being made. My goal in these sections is to leverage Dart's avowed familiarity to make significant headway in giving a real feel for what the language is. Since these are real projects, they are great opportunities to point out Dart's strengths and, yes, some of its weaknesses.

Each of these project chapters is followed by smaller, topic-specific chapters that go into a bit more depth about aspects of the language. I use these to cover material that is too detailed for the project chapters and material that cannot be found in current Dart reference material.

• Click HERE to purchase this book now. discuss

So, if you want a quick introduction to the language, you can certainly start by reading the project chapters alone. If you want a more traditional book, then skip the project chapters and read just the topic chapters. Or read it all—I will try to make it worth your time!

The first project is Chapter 1, *Project: Your First Dart Application*, on page ?. Supplementing that project are Chapter 2, *Basic Types*, on page ?; Chapter 3, *Functional Programming in Dart*, on page ?; Chapter 4, *Manipulating the DOM*, on page ?; and Chapter 5, *Dart and JavaScript*, on page ?.

The next project takes the simple Ajax application from *Project: Your First Dart Application* and whips it into something familiar to JavaScript hipsters: an MVC-like code structure. Looking at this in Chapter 6, *Project: MVC in Dart*, on page ? gives us the chance to really use Dart's fabulous object-oriented programming facilities and leads directly to a more detailed discussion in Chapter 7, *Classes and Objects*, on page ?. And, since no MVC application is complete without events, we talk about them in Chapter 8, *Events and Streams*, on page ?.

If you want to put a language through its paces, extracting code out into a library, especially an MVC library, is a great way to do it. We do just that in Chapter 9, *Project: Extracting Libraries*, on page ?. Following up on the MVC library, we get to talk about Dart's amazing support for libraries and packages in Chapter 10, *Libraries*, on page ?.

Next we'll take a look at dependency injection in Dart with Chapter 11, *Project: Varying Behavior*, on page ?. Unlike JavaScript, Dart is not primarily a dynamic language, though as you'll see in that chapter, it is still possible to perform some tricks of traditional dynamic languages. The follow-up to that project is an introduction to Dart testing, which might just be Dart's killer feature.

The last project chapter is Chapter 13, *Project: An End to Callback Hell*, on page ?, in which we further explore Dart "futures" as a higher-order replacement for traditional callback passing. This leads into a discussion of code isolation and message passing in Chapter 14, *Futures and Isolates*, on page ?.

Finally, we conclude the book with a brief exploration of various HTML5 technologies that are not covered elsewhere in the book.

• Click HERE to purchase this book now. discuss

What Is Not in This Book

We will not cover the Dart Editor. In some regards, this is something of a loss—strongly typed languages like Dart lend themselves to code completion, of which the Dart Editor takes advantage. Still, the focus of the book is meant to be the language, not the tools built around it. Besides, some people (myself included) will want to stick with their code editor of choice.

Although Dart boasts some pretty impressive server-side, Node.js-like features, we will not cover them in this book. Many of the code samples that are included use a web server, so peruse them if you are curious about how that looks.

This book is not intended as a language reference. It has been hard enough keeping even a book this size up-to-date as the language evolves! Still, the hope is that, by being a useful and pragmatic introduction to the language, it can serve as a strong supplement for the API documentation (which is very nicely done).[1]

About the Future

Since Dart continues to evolve, so does this book. Once or twice a year, depending on how quickly Dart changes, the content in this book is reviewed and then revised, removed, or supplemented.

If you identify any mistakes or areas in need of improvement, please record them in the errata: http://pragprog.com/titles/csdart/errata/add. Suggestions for new topics to cover are also welcome!

Conventions

Class names are camel-cased (for example, `HipsterModel`). Classes have filenames that are identical to the class names (for example, `HipsterModel.dart`). Variable names are snake-cased (for example, `background_color`), while functions and methods are lowercase camel-cased (for example, `routeToRegExp()`).

Let's Get Started

With the preliminaries out of the way, let's get started coding for the Web without the legacy of the Web. Let's code some Dart!

1. http://api.dartlang.org