JavaScript Object Programming

Martin Rinehart

Apress®

JavaScript Object Programming

ISBN-13 (pbk): 978-1-4842-1786-3

ISBN-13 (electronic): 978-1-4842-1787-0

Managing Director: Welmoed Spahr
Lead Editor: Jeffrey Pepper
Editorial Board: Steve Anglin, Pramila Balan, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Mark Powers
Copy Editor: Kezia Endsley
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

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.springeronline.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 materials referenced by the author in this text is available to readers at www.apress.com/9781484217863. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

Dedicated to Brendan Eich. He wrote JavaScript in 1995 giving us its object literals and object programming.

JSW: JSWindows, v2

Contents at a Glance

Contents

About the Author

Martin Rinehart, a self-confessed JavaScript lover, set aside work on his five-volume frontend-engineering textbook project for long enough to write this small book on JavaScript inheritance. He wanted to eliminate some of the massive confusion surrounding this important subject. (Veterans of classical OOP backgrounds, and Martin is one, have to unlearn much of what they think they "know." Veteran JavaScripters have to stop abusing the prototype chain.) Martin is the author of over a dozen books on programming, and of the JSWindows system that brings a windowing UI to browser-based applications.

A Note for the Implementers

JavaScript, or more exactly, the subset of JavaScript Crockford identifies as "The Good Parts," is a beautiful language. It is small, yet expressive. Its functional programming and object programming gives it extraordinary depth. In nearly a half century of programming I have used dozens of languages. Only two of them, JavaScript being one, have been languages I've loved.

Today there are people working to free JavaScript from the browser, to further empower JavaScript (WebGL, to mention a personal favorite) and to bring it up to professional speed. My apologies to the latter group. In many ways, object programming is the enemy of compiled speed.

So a word of encouragement and advice to our courageous implementers. First, making JavaScript run at some reasonable fraction of C's speed is a magnificent goal. More power to you! (And yes, that's self-serving. You are giving more power to all of us who write JavaScript. We love it and we thank you for it.)

Second, removing object programming to gain speed cuts out the heart to save the patient. Object programming is not your enemy, it is the essence of the language. Look on it as a challenge, as the Everest of your profession. The view from the top will be spectacular. Object programming at half the speed of C will be breathtaking.

—Martin Rinehart, 15 November, 2015
Delaware Valley, Pennsylvania, USA

Introduction

Hanover, NH. September, 1965. Dartmouth's incoming freshmen were told to go to an hour-long lecture by Professor John G. Kemeny. Being incoming freshmen, we went. Along with Kurtz, Kemeny had designed a new computer language called BASIC. It was supposed to be much easier to learn than existing languages. In an hour Kemeny taught us how to use most of it. (And he made us laugh a lot, too. He was a great teacher.)

I was lucky to be there then. Kemeny told us to go write a program to compute the value of pi. Using Dartmouth's new, time-sharing computer system and a teletype terminal (which punched your program into paper tape in lieu of disk storage), I actually got pi to two decimal places. My career as a mathematician (the reason I had chosen Dartmouth) ended and my career as a programmer began. I've been coding ever since.

During the last half century I've watched software revolutions come and go. Mostly go. Only two have really stuck. First, structured programming let us get rid of `goto` statements in the late 70s. And then there was the object revolution that started in the 80s. Objects had completely taken over before the end of the century.

I remember when fourth and fifth generation languages were going to take over from C and BASIC. C and BASIC are still here.

Maybe some of you remember the component revolution that was going to replace object-oriented programming.

Threaded interpreters (remember Forth?) were going to take over the software world.

JavaScript, with its hybrid class-based/prototypal model, is leading the way in objects. I bet that others will be copying it. But I've learned that predicting the future is not a science. I've been right, sometimes. I was right when I decided to switch from C to C++. (There have been other times. Mercifully, I'm forgetting more these days.)

I left C for C++ in the early 90s. I left C++ (it was getting big and heavy) for Java in 95. When I first started using object-oriented programming in JavaScript (2006), I looked for all my old friends. Classes, for one. They weren't there. (More exactly, I didn't see them.) But the language was good, there were no thoughts I couldn't implement, and so in a week I'd tricked out JavaScript to behave like my old friends. And, in my ignorance, lost the best parts of JavaScript. By 2008, my JavaScript had started to look like JavaScript.

In 50 years I've used lots of languages. Two I've loved. JavaScript is one of them. Mostly it's the object programming that I love. (Well, that plus the functional programming, which I really mean to master one of these days.)

This book is about object programming. Hope you enjoy it.