

JavaScript Object Programming



Martin Rinehart

Apress®

JavaScript Object Programming

Copyright © 2015 by Martin Rinehart

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

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

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

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

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

About the Author	xv
A Note for the Implementers	xvii
Introduction	xix
■ Chapter 1: Creating Objects.....	1
■ Chapter 2: Object Programming	15
■ Chapter 3: Inheritance Theory	25
■ Chapter 4: Inheritance Practice.....	37
■ Chapter 5: On OOP Principles	51
■ Chapter 6: More <i>Ex Nihilo</i> Objects	59
■ Chapter 7: Inheritance Alternatives.....	65
■ Chapter 8: Designing for JavaScript.....	83
■ Chapter 9: On Constructors	89
■ Chapter 10: Appendices	97
Index.....	107

Contents

About the Author	xv
A Note for the Implementers	xvii
Introduction	xix
■ Chapter 1: Creating Objects.....	1
Reasons for Objects	1
Objects Do Methods.....	1
Event-Driven Programming	2
Taming Exponential Complexity.....	2
Class-Based vs. Prototypal.....	2
Simula.....	2
Smalltalk.....	2
C++ and Java.....	3
Self and JavaScript.....	3
Objects Up Close	4
Data Properties.....	4
Methods (Code Properties)	4
Ex Nihilo Object Creation.....	5
The Object Constructor	5
Object Literals.....	6
More Ex Nihilo Objects.....	7

- OOP-Style Object Creation..... 8
 - Constructors8
 - Assigning Initial Property Values8
 - Creating Instance Methods.....9
 - Creating Class Statics.....9
 - Getters and Setters.....9
 - Default Values..... 10
- Prototypal Object Creation..... 10
 - Object Prototypes 10
 - The Prototype Chain 12
 - Object Prototype Cloning 12
- Summary 14
- **Chapter 2: Object Programming 15**
 - JSWindows Sample System..... 15
 - OP Removes Restrictions 15
 - OP Defined..... 16
 - Programming with Properties 16
 - Dot Notation..... 16
 - Subscript Notation 17
 - Object Programming Examples 17
 - Object Sum 17
 - OP for Inheriting Prototypes 19
 - OP in the JSWindows Library 19
 - DOM Related..... 19
 - Utility21
 - Summary 24

■ Chapter 3: Inheritance Theory	25
Classes	25
Constructors	26
Instance Methods	26
Class (Family-Wide) Properties	27
Class-Based Inheritance	27
Property Sets	28
Constructing an Extending Instance	29
Overriding Properties	29
Inheritance Chains	30
Prototypal Inheritance	31
Inheritance vs. Composition	32
Composition in Theory	32
Composition in JSWindows	32
Summary	35
■ Chapter 4: Inheritance Practice	37
Cascading <code>init()</code> Methods for Data	37
A Theoretical Example	39
A Practical Example	42
Prototypes for Methods	44
Prototype Inheritance Alternatives	46
Prototype Alternatives	48
JSWindows Inheritance	49
Summary	49

- **Chapter 5: On OOP Principles 51**
 - Ranking OOP Principles..... 51
 - Inheritance 52
 - Encapsulation..... 52
 - Access Specifiers 52
 - Closures..... 53
 - Polymorphism 53
 - Subtype Polymorphism..... 53
 - Parametric Polymorphism 54
 - Ad Hoc and Other Polymorphism..... 55
 - JavaScript and Polymorphism..... 55
 - Classes, Abstraction, and Interfaces 56
 - Classes 56
 - Abstraction 57
 - Interfaces..... 57
 - Other OOP Principles 58
 - Summary 58
- **Chapter 6: More *Ex Nihilo* Objects 59**
 - The Ex Nihilo Namespace Object 59
 - The Ex Nihilo Class..... 60
 - Returning Ex Nihilo Objects 61
 - The Function as an Ex Nihilo Class..... 61
 - Summary 62

Chapter 7: Inheritance Alternatives.....	65
Multiple Inheritance	65
Interfaces	67
Capabilities.....	68
The Window[_M[_BS]] Problem	68
Mixins.....	70
Calling Capability Methods.....	71
Capabilities as Constructor Properties	71
Capabilities as Single Properties.....	71
Capability Prototype Methods.....	72
Examples.....	72
Closable	73
Maskable	74
Button_sizable.....	76
Summary.....	81
Chapter 8: Designing for JavaScript.....	83
Use Ex Nihilo Constantly.....	83
Array Literals	83
Styles Objects.....	84
Other Objects.....	84
Use Composition Liberally	84
Mature Pos_size	85
Use Capabilities Liberally	86
Use Inheritance Conservatively	87
Summary.....	88

- **Chapter 9: On Constructors** **89**
 - Constructor Magic 89
 - The new Operator 89
 - The this Parameter 90
 - The constructor.prototype..... 90
 - The “[[prototype]]” Property 91
 - The Prototype’s Prototype 92
 - “[[prototype]]” Implies..... 92
 - The Dynamic Prototype 93
 - A Bit More Magic 93
 - The Constructor Returns this..... 93
 - The “Magic” Summarized..... 94
 - Constructors Are Not for Inheritance 94
 - Summary 95
- **Chapter 10: Appendices** **97**
 - A Surveyed Pages, OOP Principles 97
 - B Selected Books 98
 - C++ 98
 - Java 99
 - JavaScript..... 99
 - Python..... 99
 - Visual Basic 99
 - C Selected Websites 99
 - Wikipedia on Object-Oriented Programming, Class-Based Inheritance and Prototypal Inheritance 99
 - The Author’s Web Site on Class-Based Inheritance and JavaScript Programming 100
 - Other Web Sites on Class-Based Inheritance and Prototypal Inheritance 100

D Defined Terms	102
E Support for Selected Statements	102
F Simple Closure	103
G Sealing and Freezing Objects.....	104
H Configuring Properties	105
I Dynamic Properties and Me	106
Index.....	107

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.