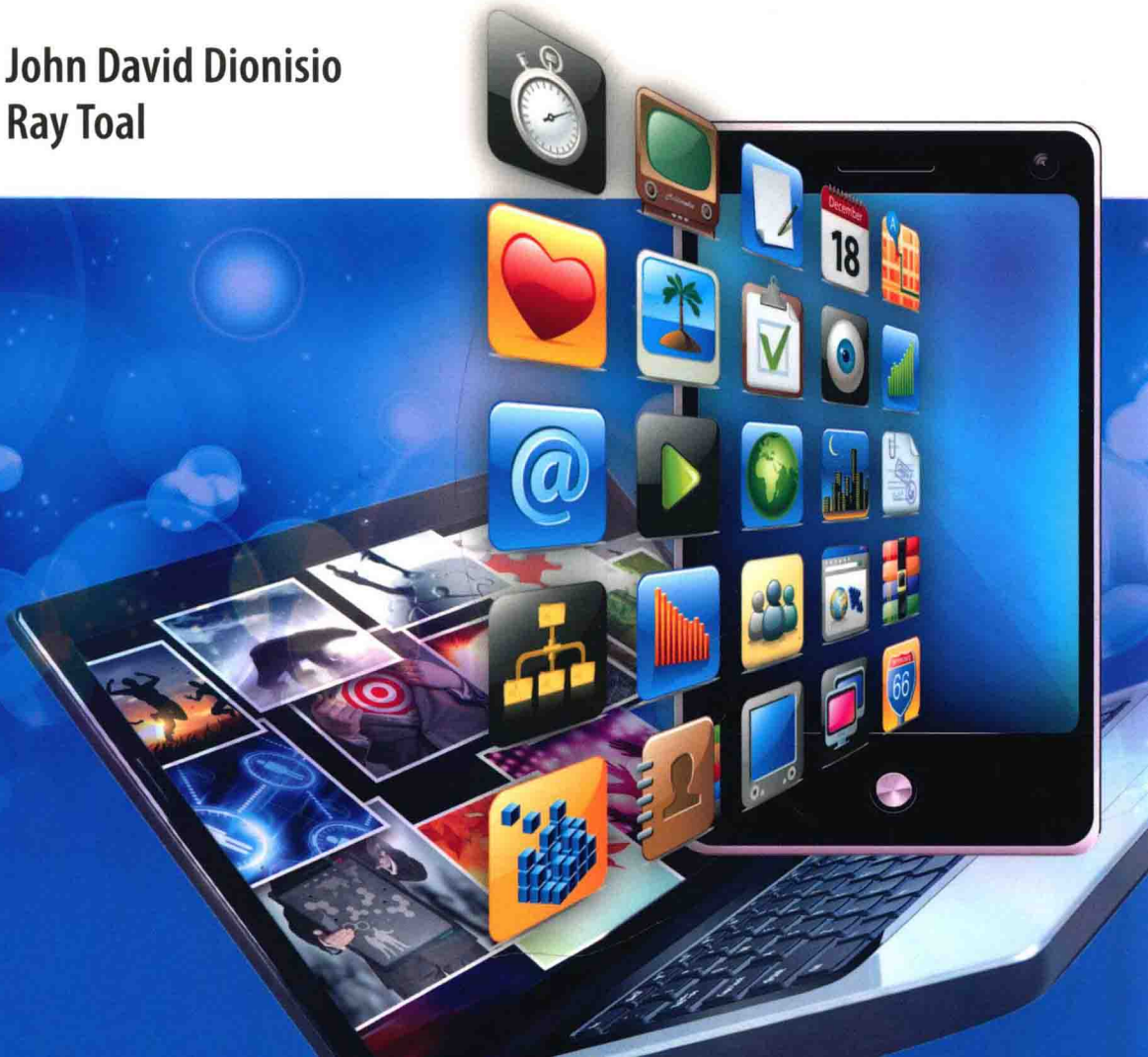


PROGRAMMING WITH

JavaScript

*Algorithms and Applications for Desktop
and Mobile Browsers*

John David Dionisio
Ray Toal



PROGRAMMING WITH

JavaScript

Algorithms and Applications for Desktop

Algorithms and Applications for Desktop and Mobile Browsers

John David Dionisio

Loyola Marymount University

Ray Toal

Loyola Marymount University



JONES & BARTLETT
LEARNING

World Headquarters
Jones & Bartlett Learning
5 Wall Street
Burlington, MA 01803
978-443-5000
info@jblearning.com
www.jblearning.com

Jones & Bartlett Learning books and products are available through most bookstores and online booksellers. To contact Jones & Bartlett Learning directly, call 800-832-0034, fax 978-443-8000, or visit our website, www.jblearning.com.

Substantial discounts on bulk quantities of Jones & Bartlett Learning publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones & Bartlett Learning via the above contact information or send an email to specialsales@jblearning.com.

Copyright © 2013 by Jones & Bartlett Learning, LLC, an Ascend Learning Company

All rights reserved. No part of the material protected by this copyright may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

Programming with JavaScript: Algorithms and Applications for Desktop and Mobile Browsers is an independent publication and has not been authorized, sponsored, or otherwise approved by the owners of the trademarks referenced in this product. Some elements of this book may fall under the Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) license—contact Jones & Bartlett Learning for specific licensing information.

Production Credits

Publisher: Cathleen Sether
Senior Acquisitions Editor: Timothy Anderson
Managing Editor: Amy Bloom
Director of Production: Amy Rose
Marketing Manager: Lindsay White
V.P., Manufacturing and Inventory Control: Therese Connell
Associate Photo Researcher: Lauren Miller
Composition: Northeast Compositors, Inc.
Cover Design: Kristin E. Parker
Cover Image: Light: © yienkeat/Shutterstock, Inc.; Laptop: © Haywiremedia/Shutterstock, Inc.
Photo display: © James Thew/Shutterstock, Inc.; Smartphone: © lassedesignen/Fotolia.com
App Icons: © abdulsatarid/Shutterstock, Inc.
Printing and Binding: Courier Kendallville
Cover Printing: Courier Kendallville

Library of Congress Cataloging-in-Publication Data

Dionisio, John David N., 1970-
Programming with JavaScript : algorithms and applications for desktop and mobile browsers / John David Dionisio, Ray Toal.

p. cm.

Includes bibliographical references and index.

ISBN-13: 978-0-7637-8060-9 (pbk.)

ISBN-10: 0-7637-8060-X (pbk.)

1. JavaScript (Computer program language) 2. Computer algorithms. 3. Application software—Development. I. Toal, Ray. II. Title.

QA76.73.J38D57 2013

005.3—dc23

2011018738

Dedication

Love and thanks as always to Mei Lyn, Aidan, Anton, and Aila for their support.

JDND

Preface

What comes to mind when you hear the terms *programming* and *computer science*? Game-playing, socially challenged geeks? Computers? Those are certainly the popular images. In reality, however, *anyone* can program,¹ and computer science is about much more than computers. You are just as likely to see people programming phones, robots, navigation systems, and factory machinery as you are desktop computers.

Programming with JavaScript: Algorithms and Applications for Desktop and Mobile Browsers is an introduction to some of the main ideas and principles of computer science, with some forays into the related disciplines of software engineering and information technology. It aims to convey these principles *by encouraging you to develop fundamental skills in programming*. Computer science deals with many things—computation, algorithms, software systems, data organization, knowledge representation, language, intelligence, and learning—but it is programming experience that enables you to gain a better understanding of these topics, and the tools to explore them in depth.

Objectives

This book aims to:

- Introduce the field of computing by showing that it is a natural science, encompassing computer science, software engineering, computer engineering, information systems, and information technology.

¹“Anyone can” means that great programmers can, and do, come from any background, not that programming can be learned without effort [Bra07].

- Dispel common myths about what computing is and show that computing provides a foundation for careers in many different areas, including medicine, law, business, finance, entertainment, the arts, education, economics, biology, nanotechnology, and gaming.
- Teach a respect for programming aesthetics, standards, style conventions, and judicious commenting early in the text, with the goal of preventing common bad habits from ever forming.
- Convey the power of JavaScript (as compared to other languages) by covering difficult material that is traditionally not taught to beginners. Some of this advanced material is isolated into sections marked with an asterisk (*) or is included in the appendices.
- Show that programming is not just about getting programs to work correctly, but is also about constructing programs that are readable, easily modifiable, and that run efficiently.
- Provide relevant case studies in distributed computing, touch-based user interfaces on phones and tablets, and graphics for both the student looking forward to employment and the professional programmer looking to keep current in modern software technology.

Organization

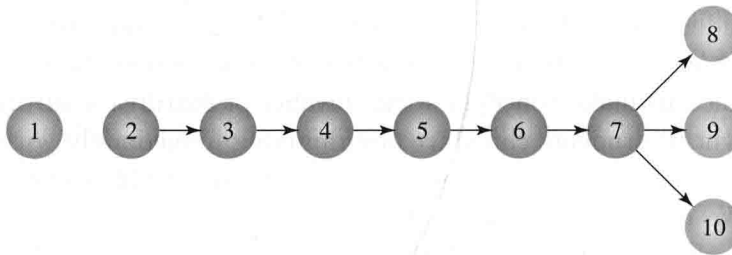
We've structured this text so that you can read it cover to cover if you like. It tells a story about computing, programming, and especially JavaScript, outlined as follows:

- Introduction to the field of computing (*Chapter 1*)
- Theory and practice of (JavaScript) programming (*Chapters 2–8*)
 - Getting started with programming (*Chapter 2*)
 - Data (*Chapter 3*)
 - Programming in the small I: Statements (*Chapter 4*)

- Programming in the small II: Functions (*Chapter 5*)
- Programming in the small III: Events (*Chapter 6*)
- Programming in the large I: Software systems construction (*Chapter 7*)
- Programming in the large II: Distributed computing (*Chapter 8*)

■ Advanced topics (*Chapters 9–10*)

While you need not read the text exactly cover to cover, you may want to keep the chapter dependencies in mind, shown here:



Note that Chapter 1 stands alone: it's optional. Readers who want to jump right in to programming can start with Chapter 2.

Audience

This text is designed as a primary resource for a first-year college course in computer science or software engineering. No previous programming experience is assumed. However, advanced students and professional programmers new to JavaScript should also find the text useful, as we do not shy away from technical areas of the language perceived as difficult or “advanced.” In fact, we believe that professional programmers can benefit a great deal from the numerous review questions and exercises spread throughout the text, as well as our coverage of modern topics in the JavaScript world, including ECMAScript 5, HTML 5, Ajax, jQuery, Graphics, and Animation.

JavaScript

A note to instructors: We enthusiastically adopt JavaScript as the language with which to train new computer scientists. JavaScript has not traditionally found much traction in introductory university-level computer science courses; this is probably due to various misunderstandings about the language [Cro01]. We argue, however, that JavaScript is an *ideal* language for such courses.

First, thanks to the ubiquity of web browsers, every student already has access to a JavaScript interpreter; no download or installation is required. Second, the language finds middle ground in the debate between professors who claim that beginning students should focus not on programming but on abstract algorithms given in pseudocode, and those who argue that students require hands-on programming experience to make concepts stick. JavaScript features a surprisingly clear and simple syntax; students can start programming immediately without fretting about classes, “public static” methods, the mysterious `void`, consoles, packages, and so on. We realize many schools have tried the simple-language approach in CS1 with ML, Scheme, Ruby, or Python, but with the rise of the Web as a platform for running applications (both on desktop and mobile devices), none of these languages can boast nearly the same degree of popularity as JavaScript.

Finally, as functional programming, long thought of as being of interest only to academic computer scientists, becomes more important in the new world of multicore processors and Big Data, JavaScript as a teaching language makes a great deal of sense. Functional programming in JavaScript tends to be fairly accessible to beginning students, perhaps more so than languages known for having “too many parentheses” or a reliance on special constructs like blocks, continuations, or generators.

Additional Resources

Visit go.jblearning.com/Dionisio for answers to end-of-chapter exercises, source code, PowerPoint Lecture Outlines, errata, and additional bonus material outside the scope of this text.

Acknowledgments

We'd like to express our thanks to Loren Abrams, Turn Media; B. J. Johnson, Claremont Graduate University; Philip Dorin, Loyola Marymount University; Daniel Bogaard, Rochester Institute of Technology; Michael Hennessy, University of Oregon; and Laurence Toal, Wellesley College, for their careful readings of early drafts and many constructive comments. Thanks also to Kira Toal and Masao Kitamura for providing several images, and to Jasmine Dahilig, Tyler Nichols, and Andrew Fornery for their assistance in preparing ancillary materials. We are also grateful for the excellent support from the staff at Jones & Bartlett Learning, including Tim Anderson, Senior Acquisitions Editor; Amy Bloom, Managing Editor; and Amy Rose, Production Director, without whose professionalism and hard work this book would not have been possible. We'd also like to thank Caskey Dickson and Technocage, Inc., for hosting our cross-site scripting examples. Without them, there would be no sites to cross!

Contents

Preface	xv
1 The Field of Computing	1
1.1 Computing Is a Natural Science	2
1.2 The Five Disciplines of Computing	3
1.2.1 Computer Science	3
1.2.2 Software Engineering	4
1.2.3 Computer Engineering	5
1.2.4 Information Technology	5
1.2.5 Information Systems	6
1.3 Careers in Computing	7
1.4 Myths about Computing	9
Exercises	11
2 Programming	15
2.1 Learning to Program	16
2.2 Getting Started	17
2.2.1 The Browser Address Box	18
2.2.2 Runner Pages	19
2.2.3 Interactive Shells	23
2.2.4 Files	25
2.3 Elements of Programs	32
2.3.1 Expressions	32
2.3.2 Variables	35

2.3.3	Statements	39
2.4	The Practice of Programming	41
2.4.1	Comments	42
2.4.2	Coding Conventions	43
2.4.3	Code Quality Tools	44
2.5	The JavaScript Programming Language	46
	Exercises	48
3	Data	55
3.1	Data Types	56
3.2	Truth Values	57
3.3	Numbers	59
3.3.1	Numeric Operations	59
3.3.2	Size and Precision Limits	60
3.3.3	NaN	62
3.3.4	Hexadecimal Numerals	63
3.4	Text	64
3.4.1	Characters, Glyphs, and Character Sets	64
3.4.2	String Operations	69
3.5	Undefined and Null	70
3.6	Objects	71
3.6.1	Object Basics	71
3.6.2	Understanding Object References	74
3.6.3	Object Prototypes	76
3.6.4	Self-Referential Objects	79
3.7	Arrays	80
3.8	Type Conversion	83
3.8.1	Weak Typing	83
3.8.2	Explicit Conversion	86
3.8.3	Loose Equality Operators	89
3.9	The <code>typeof</code> Operator*	90
	Exercises	92

4	Statements	99
4.1	The Declaration Statement	100
4.2	The Expression Statement	101
4.3	Conditional Execution	104
4.3.1	The <code>if</code> Statement	104
4.3.2	The Conditional Expression	107
4.3.3	The <code>switch</code> Statement	107
4.3.4	Avoiding Conditional Code with Lookups	110
4.3.5	Short-Circuit Execution	115
4.4	Iteration	117
4.4.1	The <code>while</code> and <code>do-while</code> Statements	117
4.4.2	The <code>for</code> Statement	119
4.4.3	The <code>for-in</code> Statement	126
4.5	Disruption	128
4.5.1	<code>break</code> and <code>continue</code>	129
4.5.2	Exceptions	132
4.6	Coding Forms to Avoid	135
4.6.1	Blockless Compound Statements	136
4.6.2	Implicit Semicolon	138
4.6.3	Implicit Declarations	138
4.6.4	Increment and Decrement Operators	139
4.6.5	The <code>with</code> Statement	139
	Exercises	141
5	Functions	147
5.1	Black Boxes	148
5.2	Defining and Calling Functions	149
5.3	Examples and More Examples	152
5.3.1	Simple One-Line Functions	152
5.3.2	Validating Arguments	154
5.3.3	Passing Object References as Arguments	156
5.3.4	Preconditions	158
5.3.5	Separation of Concerns	160
5.3.6	The Fibonacci Sequence	163

5.4	Scope	164
5.5	Functions as Objects	168
5.5.1	Properties of Functions	168
5.5.2	Functions as Properties	169
5.5.3	Constructors	171
5.6	Context	178
5.7	Higher-Order Functions	180
5.8	Function Declarations Versus Function Expressions*	184
	Exercises	188
6	Events	197
6.1	User Interaction	198
6.1.1	A Programming Paradigm Shift	198
6.1.2	Events by Example: The Temperature Converter Web Page	201
6.2	Defining User Interface Elements	203
6.2.1	Web Pages Are Structured Documents	203
6.2.2	Elements That Produce User Interface Controls	206
6.3	Programmatically Accessing User Interface Elements	211
6.3.1	The <code>document</code> Object	212
6.3.2	Fun with DOM Properties	215
6.3.3	A Place to “Play”	217
6.3.4	Manipulating User Interface Controls	217
6.3.5	Walking the DOM*	221
6.4	Event Handlers	228
6.4.1	Anatomy of an Event Handler	229
6.4.2	Event Handlers Are Functions Are Objects	230
6.5	Event Objects	233
6.6	Event Implementation Details	235
6.6.1	Event Capturing and Bubbling	235
6.6.2	Default Actions	237
6.6.3	Assigning Event Handlers	240
6.6.4	Events Based on the Passage of Time	242
6.6.5	Multitouch, Gesture, and Physical Events	244

6.7	Case Study: Tic-Tac-Toe	252
6.7.1	Files and Connections	252
6.7.2	Initialization	254
6.7.3	Event Handling	258
6.7.4	The Business Logic	258
	Exercises	261
7	Software Construction	273
7.1	Software Engineering Activities	274
7.2	Object-Oriented Design and Programming	275
7.2.1	Families of Objects	275
7.2.2	Inheritance	280
7.2.3	Information Hiding	285
7.2.4	Property Descriptors*	289
7.3	JavaScript Standard Objects	292
7.3.1	Built-in Objects	293
7.3.2	Web Browser Host Objects	308
7.4	Modules	310
7.4.1	Simple Modules	311
7.4.2	The Tic-Tac-Toe Game as a Module	313
7.5	The jQuery JavaScript Library	318
7.6	Performance	325
7.6.1	Run-Time Efficiency	325
7.6.2	Space Efficiency	330
7.6.3	Load-Time Efficiency	331
7.6.4	User Interface Efficiency	333
7.7	Unit Testing	337
7.7.1	An Introductory Example	339
7.7.2	The QUnit Testing Framework	341
7.7.3	Testing in the Software Development Process	346
	Exercises	347
8	Distributed Computing	359
8.1	Distributed Computing Models	360

8.2	Data Interchange Formats	362
8.2.1	Plain Text	362
8.2.2	XML	364
8.2.3	JSON	369
8.2.4	YAML	373
8.3	Synchronous vs. Asynchronous Communication	375
8.4	Ajax	377
8.4.1	Ajax in jQuery	377
8.4.2	Ajax Without a Library	384
8.5	Designing Distributed Applications	388
8.5.1	Uniform Resource Identifiers	388
8.5.2	REST	393
8.5.3	Separation of Distributed Application Concerns	399
8.5.4	Server-Side Technologies*	404
8.6	Security	406
8.6.1	The Web, the Bad, and the Sandbox	407
8.6.2	The Same Origin Policy	409
8.6.3	Cross-Site Scripting	415
8.6.4	Mashups	420
8.7	Case Study: Events and Trending Topics	424
8.7.1	Date Selection User Interface	429
8.7.2	Ajax Connection	431
8.7.3	Result Processing	435
8.7.4	Data (Mashup) Display	439
	Exercises	443
9	Graphics and Animation	463
9.1	Fundamentals	464
9.1.1	Coordinate Spaces	464
9.1.2	Colors	466
9.1.3	Pixels vs. Objects/Vectors	467
9.1.4	Animation	470
9.2	HTML and CSS	471
9.2.1	HTML Elements for Graphics	471

9.2.2	CSS	473
9.2.3	Visual Properties	477
9.2.4	Absolute Position	483
9.2.5	Case Study: Bar Chart	485
9.2.6	Case Study: Towers of Hanoi Display	487
9.3	Animation in HTML and CSS	491
9.3.1	Constant Velocity	492
9.3.2	Fading In and Out	493
9.3.3	Animating Other Properties	495
9.3.4	Ramped (or Eased) Animation	495
9.3.5	Declarative CSS Animation	497
9.4	The <code>canvas</code> Element	499
9.4.1	Instantiating a <code>canvas</code>	499
9.4.2	The Rendering Context	500
9.4.3	Drawing Rectangles	502
9.4.4	Drawing Lines and Polygons	503
9.4.5	Drawing Arcs and Circles	505
9.4.6	Drawing Bézier and Quadratic Curves	507
9.4.7	Working with Images	509
9.4.8	Transformations	514
9.4.9	Animation	521
9.4.10	<code>canvas</code> by Example	523
9.5	SVG	529
9.5.1	Seeing SVG in a Web Browser	530
9.5.2	SVG Case Study: A Bézier Curve Editor	533
9.5.3	Objects in the Drawing	535
9.5.4	Reading and Writing Attributes	536
9.5.5	Interactivity (aka Event Handling Redux)	540
9.5.6	Other SVG Features	544
9.6	3D Graphics with WebGL	545
9.6.1	WebGL Is the 3D <code>canvas</code>	546
9.6.2	Case Study: The Sierpinski Gasket	546
9.6.3	Defining the 3D Data	549
9.6.4	Shader Code	551

9.6.5	Drawing the Scene	553
9.6.6	Interactivity and Events	554
9.7	Other Client-Side Graphics Technologies	556
9.7.1	Flash	556
9.7.2	Java	557
9.7.3	VML	558
	Exercises	559
10	Advanced Topics	577
10.1	Regular Expressions	578
10.1.1	Introducing Regular Expressions	578
10.1.2	Capture	581
10.1.3	Quantifiers	582
10.1.4	Backreferences	583
10.1.5	Regex Modifiers	584
10.1.6	The RegExp Constructor	585
10.1.7	More on Regular Expressions	586
10.2	Recursion	586
10.2.1	What Is Recursion?	587
10.2.2	Classic Examples of Recursion	588
10.2.3	Recursion and Family Trees	600
10.2.4	When Not to Use Recursion	603
10.3	Caching	605
10.4	MapReduce	609
10.4.1	Using map , filter , and reduce	609
10.4.2	Implementation	612
10.4.3	MapReduce in Large-Scale Data Processing	613
10.5	Dynamically Creating Event Handlers	614
	Exercises	619
A	JavaScript Language Reference	627
B	Numeric Encoding	655
C	Unicode	661