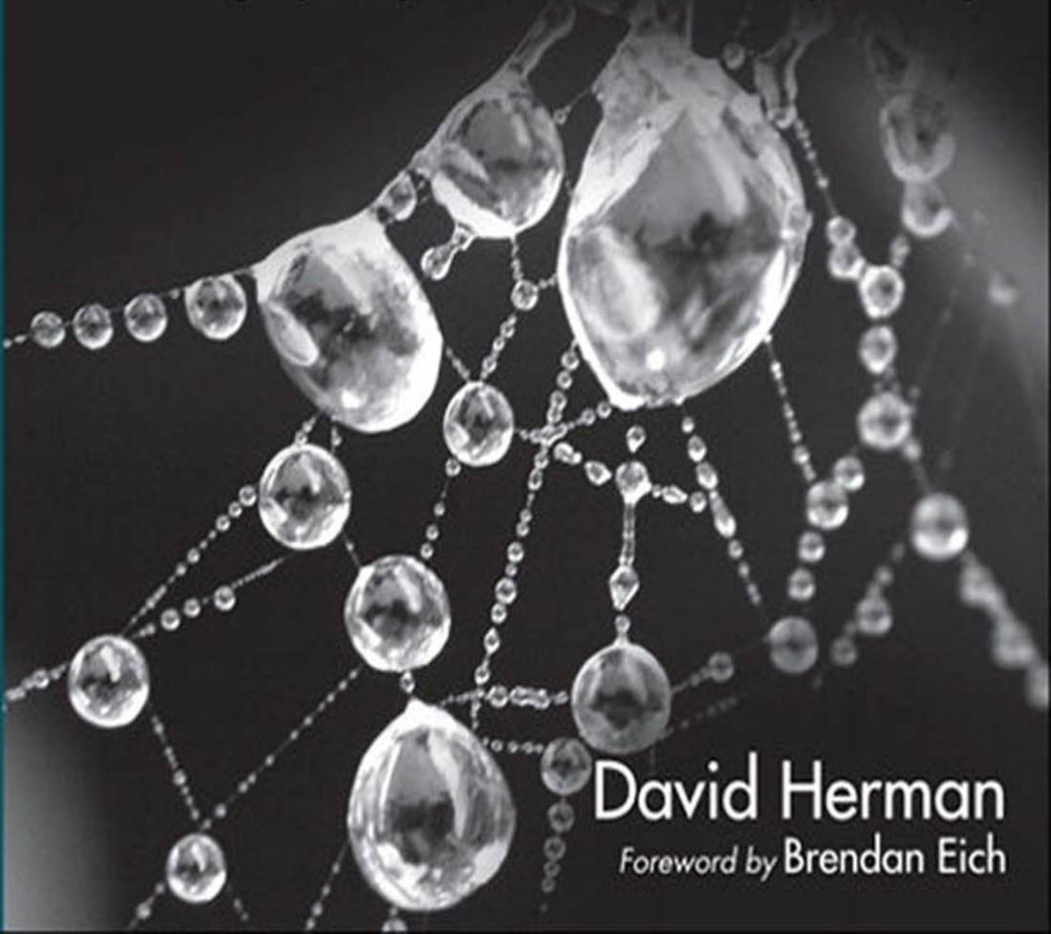
Effective Software Development Series

Scott Meyers, Consulting Editor

Effective JavaScript

68 Specific Ways to Harness the Power of JavaScript



David Herman

Foreword by Brendan Eich

Praise for *Effective JavaScript*

"Living up to the expectation of an Effective Software Development Series programming book, *Effective JavaScript* by Dave Herman is a must-read for anyone who wants to do serious JavaScript programming. The book provides detailed explanations of the inner workings of JavaScript, which helps readers take better advantage of the language."

—Erik Arvidsson, senior software engineer

"It's uncommon to have a programming language wonk who can speak in such comfortable and friendly language as David does. His walk through the syntax and semantics of JavaScript is both charming and hugely insightful; reminders of gotchas complement realistic use cases, paced at a comfortable curve. You'll find when you finish the book that you've gained a strong and comprehensive sense of mastery."

—Paul Irish, developer advocate, Google Chrome

"Before reading *Effective JavaScript*, I thought it would be just another book on how to write better JavaScript. But this book delivers that and so much more—it gives you a deep understanding of the language. And this is crucial. Without that understanding you'll know absolutely nothing whatever about the language itself. You'll only know how other programmers write their code.

"Read this book if you want to become a really good JavaScript developer. I, for one, wish I had it when I first started writing JavaScript."

—Anton Kovalyov, developer of JSHint

"If you're looking for a book that gives you formal but highly readable insights into the JavaScript language, look no further. Intermediate JavaScript developers will find a treasure trove of knowledge inside, and even highly skilled JavaScripters are almost guaranteed to learn a thing or ten. For experienced practitioners of other languages looking to dive headfirst into JavaScript, this book is a must-read for quickly getting up to speed. No matter what your background, though, author Dave Herman does a fantastic job of exploring JavaScript—its beautiful parts, its warts, and everything in between."

—Rebecca Murphey, senior JavaScript developer, Bocoup

"*Effective JavaScript* is essential reading for anyone who understands that JavaScript is no mere toy and wants to fully grasp the power it has to offer. Dave Herman brings users a deep, studied, and practical understanding of the language, guiding them through example after example to help them come to the same conclusions he has. This is not a book for those looking for shortcuts; rather, it is hard-won experience distilled into a guided tour. It's one of the few books on JavaScript that I'll recommend without hesitation."

—Alex Russell, TC39 member, software engineer, Google

"Rarely does anyone have the opportunity to study alongside a master in their craft. This book is just that—the JavaScript equivalent of a time-traveling philosopher visiting fifth century BC to study with Plato."

—Rick Waldron, JavaScript evangelist, Bocoup

Effective JavaScript: 68 Specific Ways to Harness the Power of JavaScript

Table of Contents

Contents
Foreword
Preface
Acknowledgments
About the Author
Chapter 1: Accustoming Yourself to JavaScript
Item 1: Know Which JavaScript You Are Using
Item 2: Understand JavaScripts Floating-Point Numbers
Item 3: Beware of Implicit Coercions
Item 4: Prefer Primitives to Object Wrappers
Item 5: Avoid using == with Mixed Types
Item 6: Learn the Limits of Semicolon Insertion
Item 7: Think of Strings As Sequences of 16-Bit Code Units
Chapter 2: Variable Scope
Item 8: Minimize Use of the Global Object
Item 9: Always Declare Local Variables
Item 10: Avoid with
Item 11: Get Comfortable with Closures
Item 12: Understand Variable Hoisting
Item 13: Use Immediately Invoked Function Expressions to

Table of Contents

Create Local Scopes

Item 14: Beware of Unportable Scoping of Named Function Expressions

Item 15: Beware of Unportable Scoping of Block-Local Function Declarations

Item 16: Avoid Creating Local Variables with eval

Item 17: Prefer Indirect eval to Direct eval

Chapter 3: Working with Functions

Item 18: Understand the Difference between Function, Method, and Constructor Calls

Item 19: Get Comfortable Using Higher-Order Functions

Item 20: Use call to Call Methods with a Custom Receiver

Item 21: Use apply to Call Functions with Different Numbers of Arguments

Item 22: Use arguments to Create Variadic Functions

Item 23: Never Modify the arguments Object

Item 24: Use a Variable to Save a Reference to arguments

Item 25: Use bind to Extract Methods with a Fixed Receiver

Item 26: Use bind to Curry Functions

Item 27: Prefer Closures to Strings for Encapsulating Code

Item 28: Avoid Relying on the toString Method of Functions

Item 29: Avoid Nonstandard Stack Inspection Properties

Chapter 4: Objects and Prototypes

Item 30: Understand the Difference between prototype, getPrototypeOf, and__proto__

Item 31: Prefer Object.getPrototypeOf to __proto__

Item 32: Never Modify __proto__

Table of Contents

Item 33: Make Your Constructors new-Agnostic

Item 34: Store Methods on Prototypes

Item 35: Use Closures to Store Private Data

Item 36: Store Instance State Only on Instance Objects

Item 37: Recognize the Implicit Binding of this

Item 38: Call Superclass Constructors from Subclass Constructors

Item 39: Never Reuse Superclass Property Names

Item 40: Avoid Inheriting from Standard Classes

Item 41: Treat Prototypes As an Implementation Detail

Item 42: Avoid Reckless Monkey-Patching

Chapter 5: Arrays and Dictionaries

Item 43: Build Lightweight Dictionaries from Direct Instances of Object

Item 44: Use null Prototypes to Prevent Prototype Pollution

Item 45: Use hasOwnProperty to Protect Against Prototype Pollution

Item 46: Prefer Arrays to Dictionaries for Ordered Collections

Item 47: Never Add Enumerable Properties to Object.prototype

Item 48: Avoid Modifying an Object during Enumeration

Item 49: Prefer for Loops to for...in Loops for Array Iteration

Item 50: Prefer Iteration Methods to Loops

Item 51: Reuse Generic Array Methods on Array-Like Objects

Item 52: Prefer Array Literals to the Array Constructor

Chapter 6: Library and API Design

Table of Contents

Item 53: Maintain Consistent Conventions

Item 54: Treat undefined As No Value

Item 55: Accept Options Objects for Keyword Arguments

Item 56: Avoid Unnecessary State

Item 57: Use Structural Typing for Flexible Interfaces

Item 58: Distinguish between Array and Array-Like

Item 59: Avoid Excessive Coercion

Item 60: Support Method Chaining

Chapter 7: Concurrency

Item 61: Dont Block the Event Queue on I/O

Item 62: Use Nested or Named Callbacks for Asynchronous Sequencing

Item 63: Be Aware of Dropped Errors

Item 64: Use Recursion for Asynchronous Loops

Item 65: Dont Block the Event Queue on Computation

Item 66: Use a Counter to Perform Concurrent Operations

Item 67: Never Call Asynchronous Callbacks Synchronously

Item 68: Use Promises for Cleaner Asynchronous Logic

Index

Pearson