

A Smarter Way to Learn JavaScript

The new approach that uses technology to cut your effort in half



1 Read a 10-minute chapter of this book to get each concept.

2 Code for 20 minutes at ASmarterWayToLearn.com to own the skill. (It's free.)

Mark Myers

A Smarter Way to Learn JavaScript

The new approach that uses technology to cut your effort in half

Mark Myers

copyright © 2013 by Mark Myers

Chapters

1. [Alerts](#)
2. [Variables for Strings](#)
3. [Variables for Numbers](#)
4. [Variable Names Legal and Illegal](#)
5. [Math Expressions: familiar operators](#)
6. [Math Expressions: unfamiliar operators](#)
7. [Math Expressions: eliminating ambiguity](#)
8. [Concatenating text strings](#)
9. [Prompts](#)
10. [if statements](#)
11. [Comparison operators](#)
12. [if...else and else if statements](#)
13. [Testing sets of conditions](#)
14. [if statements nested](#)
15. [Arrays](#)
16. [Arrays: adding and removing elements](#)
17. [Arrays: removing, inserting, and extracting elements](#)
18. [for loops](#)
19. [for loops: flags, Booleans, array length, and breaks](#)
20. [for loops nested](#)
21. [Changing case](#)
22. [Strings: measuring length and extracting parts](#)
23. [Strings: finding segments](#)
24. [Strings: finding a character at a location](#)
25. [Strings: replacing characters](#)
26. [Rounding numbers](#)
27. [Generating random numbers](#)
28. [Converting strings to integers and decimals](#)
29. [Converting strings to numbers, numbers to strings](#)
30. [Controlling the length of decimals](#)
31. [Getting the current date and time](#)
32. [Extracting parts of the date and time](#)
33. [Specifying a date and time](#)
34. [Changing elements of a date and time](#)
35. [Functions](#)
36. [Functions: passing them data](#)
37. [Functions: passing data back from them](#)
38. [Functions: local vs. global variables](#)
39. [switch statements: how to start them](#)
40. [switch statements: how to complete them](#)

41. [while loops](#)
42. [do...while loops](#)
43. [Placing scripts](#)
44. [Commenting](#)
45. [Events: link](#)
46. [Events: button](#)
47. [Events: mouse](#)
48. [Events: fields](#)
49. [Reading field values](#)
50. [Setting field values](#)
51. [Reading and setting paragraph text](#)
52. [Manipulating images and text](#)
53. [Swapping images](#)
54. [Swapping images and setting classes](#)
55. [Setting styles](#)
56. [Target all elements by tag name](#)
57. [Target some elements by tag name](#)
58. [The DOM](#)
59. [The DOM: Parents and children](#)
60. [The DOM: Finding children](#)
61. [The DOM: Junk artifacts and nodeType](#)
62. [The DOM: More ways to target elements](#)
63. [The DOM: Getting a target's name](#)
64. [The DOM: Counting elements](#)
65. [The DOM: Attributes](#)
66. [The DOM: Attribute names and values](#)
67. [The DOM: Adding nodes](#)
68. [The DOM: Inserting nodes](#)
69. [Objects](#)
70. [Objects: Properties](#)
71. [Objects: Methods](#)
72. [Objects: Constructors](#)
73. [Objects: Constructors for methods](#)
74. [Objects: Prototypes](#)
75. [Objects: Checking for properties and methods](#)
76. [Browser control: Getting and setting the URL](#)
77. [Browser control: Getting and setting the URL another way](#)
78. [Browser control: Forward and reverse](#)
79. [Browser control: Filling the window with content](#)
80. [Browser control: Controlling the window's size and location](#)
81. [Browser control: Testing for popup blockers](#)
82. [Form validation: text fields](#)

83. [Form validation: drop-downs](#)
84. [Form validation: radio buttons](#)
85. [Form validation: ZIP codes](#)
86. [Form validation: email](#)
87. [Exceptions: try and catch](#)
88. [Exceptions: throw](#)
89. [Handling events within JavaScript](#)

How I propose to cut your effort in half by using technology.

When you set out to learn anything as complicated as JavaScript, you sign up for some heavy cognitive lifting. If I had to guess, I'd say the whole project of teaching yourself a language burns at least a large garden-cart load of brain glucose. But here's what you may not realize: When you teach yourself, your cognitive load doubles.

Yes, all the information is right there in the book if the author has done a good job. But learning a language entails far more than reading some information. You need to commit the information to memory, which requires some kind of plan. You need to practice. How are you going to structure that? And you need some way to correct yourself when you go off-course. Since a book isn't the best way to help you with these tasks, most authors don't even try. Which means all the work of designing a learning path for yourself is left to you. And this do-it-yourself *meta*-learning, this struggle with the question of how to master what the book is telling you, takes more effort than the learning itself.

Traditionally, a live instructor bridges the gap between reading and learning. Taking a comprehensive course or working one-on-one with a mentor is still the best way to learn JavaScript if you have the time and can afford it. But, as long as many people prefer to learn on their own, why not use the latest technology as a substitute teacher? Let the book lay out the principles. Then use an interactive program for memorization, practice, and correction. When the computer gets into the act, you'll learn twice as fast, with half the effort. It's a smarter way to learn JavaScript. It's a smarter way to learn anything.

And as long as we're embracing new technology, why not use all the tech we can get our hands on to optimize the book? Old technology—i.e. the paper book—has severe limitations from an instructional point of view. New technology—i.e. the ebook—is the way to go, for many reasons. Here are a few:

Color is a marvelous information tool. That's why they use it for traffic lights. But printing color on paper multiplies the cost. Thanks to killer setup charges, printing this single word—**color**—in a print-on-demand book adds thirty dollars to the retail price. So color is usually out, or else the book is priced as a luxury item. With an ebook, color is free.

Paper itself is expensive, so there usually isn't room to do everything the author would like to do. A full discussion of fine points? Forget it. Extra help for the rough spots? Can't afford it. Hundreds of examples? Better delete some. But no such limitation applies to an ebook. What do an extra hundred digital pages cost? Usually nothing.

When a book is published traditionally, it may take up to a year for the manuscript to get into print. This means there isn't time for extensive testing on the target audience, or for the revisions that testing would inevitably suggest. And once the book is in print, it's a big,

expensive deal to issue revised editions. Publishers put it off as long as possible. Reader feedback usually doesn't lead to improvements for years. An ebook can go from manuscript to book in a day, leaving lots of time for testing and revision. After it's published, new editions with improvements based on reader feedback can come out as often as the author likes, at no cost.

With all this going for them, is there any doubt that all the best instructional books are going to be ebooks? And would anyone deny that the most helpful thing an author can do for you, in addition to publishing a good book electronically, is to take on the whole teaching job, not just part of it, by adding interactivity to help you with memorization, practice, and correction?

Here, then, is how I propose to use current technology to help you learn JavaScript in half the time, with half the effort.

- **Cognitive portion control.** Testing showed me that when they're doing hard-core learning, even strong-minded people get tired faster than I would have expected. You may be able to read a novel for two hours at a stretch, but when you're studying something new and complicated, it's a whole different ballgame. My testing revealed that studying new material for about ten minutes is the limit, before most learners start to fade. But here's the good news: Even when you've entered the fatigue zone after ten minutes of studying, you've still got the mental wherewithal to practice for up to thirty minutes. Practice that's designed correctly takes less effort than studying, yet teaches you more. Reading a little and practicing a lot is the fastest way to learn.
- **500 coding examples** that cover every aspect of what you're learning. Examples make concepts easy to grasp and focus your attention on the key material covered in each chapter. Color cues embedded in the code help you commit rules to memory. Did I go overboard and put in more examples than you need? Well, if things get too easy for you, just skip some them.
- **Tested on naive users.** The book includes many rounds of revisions based on feedback from programming beginners. It includes extra-help discussions to clarify concepts that proved to be stumbling blocks during testing. Among the testers: my technophobe wife, who discovered that, with good instruction, she could code—and was surprised to find that she enjoyed it. For that matter, I got a few surprises myself. Some things that are simple to me turned out not to be not so simple to some readers. Rewriting ensued.
- **Free interactive coding exercises paired with each chapter—1,750 of them in all.** They're the feature that testers say helps them the most. No surprise there. According to the New York Times, psychologists "have shown that taking a test—say, writing down all you can remember from a studied prose passage—can deepen the memory of that passage better than further study." I would venture that this goes double when you're learning to code. After reading each chapter, go online and practice everything you learned. Each chapter ends with a link to its accompanying online exercises. Find an index of all the exercises at <http://www.ASmarterWayToLearn.com/js/>.

- **Live coding experience.** In scripting, the best reward is seeing your code run flawlessly. Most practice sessions include live coding exercises that let you see your scripts execute in the browser.

How to use this book

This isn't a book quite like any you've ever owned before, so a brief user manual might be helpful.

- **Study, practice, then rest.** If you're intent on mastering the fundamentals of JavaScript, as opposed to just getting a feel for the language, work with this book and the online exercises in a 15-to-30-minute session, then take a break. Study a chapter for 5 to 10 minutes. Immediately go online at <http://www.ASmarterWayToLearn.com/js> and code for 10 to 20 minutes, practicing the lesson until you've coded everything correctly. Then go for a walk.
- **Use the largest, most colorful screen available.** This book can be read on small phone screens and monochrome readers, but you'll be happier if things appear in color on a larger screen. I use color as an important teaching tool, so if you're reading in black-and-white, you're sacrificing some of the extra teaching value of a full-color ebook. Colored elements do show up as a lighter shade on some black-and-white screens, and on those devices the effect isn't entirely lost, but full color is better. As for reading on a larger screen—the book includes more than 2,000 lines of example code. Small screens break long lines of code into awkward, arbitrary segments, jumbling the formatting. While still decipherable, the code becomes harder to read. If you don't have a mobile device that's ideal for this book, consider installing the free Kindle reading app on your laptop.
- **If you're reading on a mobile device, go horizontal.** For some reason, I resist doing this on my iPad unless I'm watching a video. But even I, Vern Vertical, put my tablet into horizontal mode to proof this book. So please: starting with Chapter 1, do yourself a favor and rotate your tablet, reader, or phone to give yourself a longer line of text. It'll help prevent the unpleasant code jumble mentioned above.
- **Do the coding exercises on a physical keyboard.** A mobile device can be ideal for reading, but it's no way to code. Very, very few Web developers would attempt to do their work on a phone. The same thing goes for *learning* to code. Theoretically, most of the interactive exercises *could* be done on a mobile device. But the idea seems so perverse that I've disabled online practice on tablets, readers, and phones. Read the book on your mobile device if you like. But practice on your laptop.
- **If you have an authority problem, try to get over it.** When you start doing the exercises, you'll find that I can be a pain about insisting that you get every little detail right. For example, if you indent a line one space instead of two spaces, the program monitoring your work will tell you the code isn't correct, even though it would still run perfectly. Do I insist on having everything just so because I'm a control freak? No, it's because I have to place a limit on harmless maverick behavior in order to automate the

exercises. If I were to grant you as much freedom as you might like, creating the algorithms that check your work would be, for me, a project of driverless-car proportions. Besides, learning to write code with fastidious precision helps you learn to pay close attention to details, a fundamental requirement for coding in any language.

- **Subscribe, temporarily, to my formatting biases.** Current code formatting is like seventeenth-century spelling. Everyone does it his own way. There are no universally accepted standards. But the algorithms that check your work when you do the interactive exercises need standards. They can't grant you the latitude that a human teacher could, because, let's face it, they aren't that bright. So I've had to settle on certain conventions. All of the conventions I teach are embraced by a large segment of the coding community, so you'll be in good company. But that doesn't mean you'll be married to my formatting biases forever. When you begin coding projects, you'll soon develop your own opinions or join an organization that has a stylebook. Until then, I'll ask you to make your code look like my code.
- **Email me with any problems or questions.** The book and exercises have been tested on many learners, but haven't been tested on you. If you hit a snag, if you're just curious about something, or if I've found some way to give you fits, email me at mark@ASmarterWayToLearn.com. I'll be happy to hear from you. I'll reply promptly. And, with your help, I'll probably learn something that improves the next edition.