International Journal of People-Oriented Programming
Volume 5 • Issue 1 • January-June 2016

Book Review

Speaking JavaScript

Reviewed by Steve Goschnick, School of Design, Swinburne University of Technology, Melbourne, Australia

Speaking JavaScript
Axel Rauschmayer
© 2014 by O'Reilly Media
437 pp.
$49.99
ISBN:978-1-449-36503-5

There are two distinct markets for books about JavaScript: those people new to coding who are learning to program in a computer language for the first time (or maybe the second, after a block-based language); and those very experienced in other programming languages who have been steadily drifting in large numbers across to JavaScript for over a decade or more. Let me explain the reasons behind the large difference in those two cohorts of language learners, and hence why this book is organised the way that it is and why it is a valuable asset for it:

1. Back in the days soon after the birth of the world wide web (1993), new JavaScript (1995) was meant to be the light-weight scripting language of the web, while the Java language proper, cast in the form of applets (via a Java plug-in added to the web browser) was meant to do the heavy lifting regarding any serious coding. Back then JavaScript, embedded in HTML files, was promoted as a way to merely pick up the browser events (e.g. mouse and keyboard clicks) that HTML knew little about, and hook them up with Java applets and other rich Internet content. However, in 1997 JavaScript brought dynamic web pages to the small screen. JavaScript turned out way more powerful than many had anticipated, and Java applets along with the Java plug-in have withered on the vine and are now destined to go the way of the dinosaur in the next release of Java 9 (when the Java plugin becomes 'deprecated'). It has been a slow drawn-out death for Java applets, while Javascript has been the preeminent language of the web for some long time. However, JavaScript cemented its place high up the programmer food chain in 2008, with the official inclusion of JavaScript as a necessary companion to the HTML markup language within the HTML5 Standard (HTML + CSS + JavaScript). That year several of the web browser makers

(now collectively represented as the WHATWG Community, see: https://whatwg.org) decided to go it alone with HTML V5, since the custodians of HTML, the W3C (w3c.org), had frozen the official HTML standard at V4. At the time the W3C were moving forward with the now defunct XHTML language as a replacement for straight HTML. As a compromise, W3C and WHATWG got together for a while and forged the current HTML V5 standard, which W3C finally released in 2014. In addition to that draw-out timeline, several other major players took strategic actions that catapulted JavaScript even higher: Google produced a high-performance JavaScript engine called V8 in 2008; in 2009 the innovative Node.js backend runtime environment conscripted V8 to allow JavaScript programmers to code servers and related services, clear of the browser; in 2010 Steve Jobs banished Flash from the iPhone and iPad making it clear that any video and related interactivity that happened in iOS devices in the browser, would need to be via HTML and JavaScript; Google and many others promoted Single Page Applications (SPAs) that run in the browser (or in Chrome OS) that don't need to be connected to the Internet at all to do meaningful work. I.e. From its humble lightweight scripting language beginning, JavaScript became a mainstream programming language. And across that whole period of its evolution, large numbers of professional programmers realised that JavaScript was not only here to stay, but has been improved immensely, and that they should not ignore it any longer;

2. Meanwhile, a different cohort of would-be JavaScripters emerged: a large number of people learning to code for the first time were choosing JavaScript to program interactive web pages. A milestone for this movement came early in 2012 (the 'Code Year' initiative), when then New York Mayor, Michael Bloomberg, declared his New Years resolution for 2012 was to learn to code, via CodeCademy's free online course for JavaScript programming. He encouraged many of his NY citizens and others to follow his lead. 180,000 people did. That tradition has perhaps culminated with Code.org's 'Hour of Code' event held each year for a week in December, right around the globe, but spearheaded from their web-site (code.org). Its where new coders can learn a Scratch-like block language, and then see the generated JavaScript code underneath, should they want to take up coding with a more industry-relevant language in their future.

Those are clearly two very disparate groups of would-be JavaScript programmers. This book is aimed at the first cohort, and it hits the target very well indeed. The book is divided into four parts ranging across 33 chapters. Part 1 is a single chapter of just 30 pages, which is designed to launch the experienced programmer (from C, C++, Java, C#, etc.) straight into JavaScript without further ado. It reaches right across the language syntax, while pointing out the foibles of JavaScript functions, types, arrays, scoping, objects, inheritance and more. It does so in a very concise manner, drawing your attention to how this is probably different to what you'd expect, coming from another language with somewhat similar basic syntax. The terse descriptions are interspersed with page references which point to much fuller descriptions of the issue at hand, further on into the book, should you be struggling to comprehend the current brief note at hand. I.e. this chapter is a jumpstart to JavaScript for the experienced coder, so one can roll up their sleeves and get right into it, without distraction. Those referenced fuller descriptions occupy Section 3 of the book, ranging across Chapters 7 though 25, pretty much in the form of a regular reference manual, but always with suitable and succinct code fragments interspersed throughout.

Section 2 of the book, simply named 'Background', contains much of the information that many other book authors would have put up front, keeping you from getting started on the programming right away. The titles of the enclosed chapters of this Section are descriptively accurate enough for our purposes here: Why JavaScript?; The Nature of JavaScript; How JavaScript was Created; Standardisation: ECMAScript; and, Historical JavaScript Milestones. This last mentioned chapter includes several of the milestones outlined above in my second paragraph, but with many more important entries, such as: The introduction of the JSON notation in 2001, that uses JavaScript syntax

to store data in succinctly structured text files, effectively becoming a lightweight alternative to the XML language, and which has since become the native format for many NoSQL database systems.

That leaves us with only Section 4 to cover, titled: '*Tips, Tools, and Libraries*' - which encompasses 8 chapters: The author tells us that the JavaScript language has functional gaps, but these are compensated for with techniques, tools and libraries, allowing the coder to stand on the shoulders of those that came before them. As the author states early on "In other languages you learn language features. In JavaScript, you often learn patterns instead." There is good coverage of several dominant style guides, with recommended 'best of' styles to take on as your own ("if you are not in a team that already adheres to a style"). There is discussion of the API Documentation generating tool, JSDoc, which produces HTML documentation of your code via the comments you embed, much like JavaDoc does for Java - to help keep your Project Manager happy. Some coverage of useful libraries, and links to many more directories of such resources. Given that JavaScript doesn't have language support of code *modules*, there is a chapter on available module systems and package managers, including the venerable *npm* - Node Packaged Modules (https://www.npmjs.com). Links to tools that: pickup style violations; Unit testing; and code minification (useful for an interpreted language such that JavaScript is, where your source code is often visible to all and sundry). And more links if those weren't enough to satisfy a growing thirst for all-things JavaScript. And the last link but far from the least, in the last sentence on the last page of the book, as a reward for reading this far: http://speakingjs.com - where the author makes available a free HTML version of this book, for your online reading pleasure.

The big second paragraph above is necessary for this review in my opinion, because of the contorted history of JavaScript which has left its mark upon it. Modern JavaScript is a language that has been extruded between the tectonic plates of the commercial and open source worlds. A place where open standards have persisted, despite much commercial competition and pressure, to undo them. But a language that has also advanced dramatically with support through innovation and investment from many commercial players. JavaScript is a language that has been updated and refined to remain essential in the web browser, increasingly so on the server and the mobile device, and useful elsewhere too. The broad-ranging uses of JavaScript highlight its success. For example: beyond the browser, Web apps can be used as native apps on several platforms including Chrome OS, Firefox OS and Android; JavaScript can be added to digital books, giving them interactivity within EPUB3, the standard that lies beneath several ebook formats including those in Apple's iBook store; JavaScript can be used to code in the Unity 3D game engine; there are cross-platform development environments such as PhoneGap and Cordova, that allow programs coded in JavaScript to run natively on iOS, Android, Windows, Linux and so on; and more recently React Native, the Facebook developed technology, used to define user interfaces for both iOS and Android devices, for apps coded in JavaScript. Not to mention the ability to put the supporting technology of your cross-platform JavaScript app, in a Docker container (Docker.com), and run it from out there in the cloud somewhere.

Besides the web programmer, there is no doubt that JavaScript has become a useful and even necessary language to the toolkit of the modern professional programmer, full-stack or not. However, I do have very serious doubts as to whether it should be promoted as the first programming language a person should learn. This book is a very good choice for that first cohort of JavaScript learners: the coder with pre-existing programming language skills. Meanwhile, I'm left wondering whether Michael Bloomberg ever accomplished his 2012 New Year's resolution, and if so, what he did with his new found JavaScript skill? I am sure he gave it a good try. And if nothing else I'm sure it helped him understand somewhat better, what is really involved in programming the web, the interactive book and the mobile app, how it came about, and where it is all heading in the midterm future. And getting clear glimpses of the mid-term future in ICT, is a rare thing. Beyond learning JavaScript, this book will also help you do that.