Introduction

The JavaScript core language features are defined in a standard called ECMA-262. The language defined in this standard is called ECMAScript. What you know as JavaScript in browsers and Node.js is actually a superset of ECMAScript. Browsers and Node.js add more functionality through additional objects and methods, but the core of the language remains as defined in ECMAScript. The ongoing development of ECMA-262 is vital to the success of JavaScript as a whole, and this book covers the changes brought about by the most recent major update to the language: ECMAScript 6.

The Road to ECMAScript 6

In 2007, JavaScript was at a crossroads. The popularity of Ajax was ushering in a new age of dynamic web applications, while JavaScript hadn't changed since the third edition of ECMA-262 was published in 1999. TC-39, the committee responsible for driving the ECMAScript development process, put together a large draft specification for ECMAScript 4. ECMAScript 4 was massive in scope, introducing changes both small and large to the language. Updated features included new syntax, modules, classes, classical inheritance, private object members, optional type annotations, and more.

The scope of the ECMAScript 4 changes caused a rift to form in TC-39, with some members feeling that the fourth edition was trying to accomplish too much. A group of leaders from Yahoo, Google, and Microsoft created an alternate proposal for the next version of ECMAScript that they initially called ECMAScript 3.1. The "3.1" was intended to show that this was an incremental change to the existing standard.

ECMAScript 3.1 introduced very few syntax changes, instead focusing on property attributes, native JSON support, and adding methods to already-existing objects. Although there was an early attempt to reconcile ECMAScript 3.1 and ECMAScript 4, this ultimately failed as the two camps had difficulty with the very different perspectives on how the language should grow.

In 2008, Brendan Eich, the creator of JavaScript, announced that TC-39 would focus its efforts on standardizing ECMAScript 3.1. They would table the major syntax and feature changes of ECMAScript 4 until after the next version of ECMAScript was standardized, and all members of the committee would work to bring the best pieces of ECMAScript 3.1 and 4 together after that point into an effort initially nicknamed ECMAScript Harmony.

ECMAScript 3.1 was eventually standardized as the fifth edition of ECMA-262, also described as ECMAScript 5. The committee never released an ECMAScript 4 standard to avoid confusion with the now-defunct effort of the same name. Work then began on ECMAScript Harmony, with ECMAScript 6 being the first standard released in this new "harmonious" spirit.

ECMAScript 6 reached feature complete status in 2015 and was formally dubbed "ECMAScript 2015." (But this text still refers to it as ECMAScript 6, the name most familiar to developers.) The features vary widely from completely new objects and patterns to syntax changes to new methods on existing objects. The exciting thing about ECMAScript 6 is that all of its changes are geared toward solving problems that developers actually face.

About This Book

A good understanding of ECMAScript 6 features is key for all JavaScript developers going forward. The language features introduced in ECMAScript 6 represent the foundation upon which JavaScript applications will be built for the foreseeable future. That's where this book comes in. My hope is that you'll read this book to learn about ECMAScript 6 features so that you'll be ready to start using them as soon as you need to.

Browser and Node.js Compatibility

Many JavaScript environments, such as web browsers and Node.js, are actively working on implementing ECMAScript 6. This book doesn't attempt to address the inconsistencies between implementations and instead focuses on what the specification defines as the correct behavior. As such, it's possible that your JavaScript environment may not conform to the behavior described in this book.

Who This Book is For

This book is intended as a guide for those who are already familiar with JavaScript and ECMAScript 5. While a deep understanding of the language isn't necessary to use this book, it will help you understand the differences between ECMAScript 5 and 6. In particular, this book is aimed at intermediate-to-advanced JavaScript developers programming for a browser or Node.js environment who want to learn about the latest developments in the language.

This book is not for beginners who have never written JavaScript. You will need to have a good basic understanding of the language to make use of this book.

Overview

Each of this book's thirteen chapters covers a different aspect of ECMAScript 6. Many chapters start by discussing problems that ECMAScript 6 changes were made to solve, to give you a broader context for those changes, and all chapters include code examples to help you learn new syntax and concepts.

Chapter 1: How Block Bindings Work talks about `let` and `const`, the block-level replacement for `var`.

Chapter 2: Strings and Regular Expressions covers additional functionality for string manipulation and inspection as well as the introduction of template strings.

Chapter 3: Functions in ECMAScript 6 discusses the various changes to functions. This includes the arrow function form, default parameters, rest parameters, and more.

Chapter 4: Expanded Object Functionality explains the changes to how objects are created, modified, and used. Topics include changes to object literal syntax, and new reflection methods.

Chapter 5: Destructuring for Easier Data Access introduces object and array destructuring, which allow you to decompose objects and arrays using a concise syntax.

Chapter 6: Symbols and Symbol Properties introduces the concept of symbols, a new way to define properties. Symbols are a new primitive type that can be used to obscure (but not hide) object properties and methods.

Chapter 7: Sets and Maps details the new collection types of `Set`, `WeakSet`, `Map`, and `WeakMap`. These types expand on the usefulness of arrays by adding semantics, de-duping, and memory management designed specifically for JavaScript.

Chapter 8: Iterators and Generators discusses the addition of iterators and generators to the language. These features allow you to work with collections of data in powerful ways that were not possible in previous versions of JavaScript.

Chapter 9: Introducing JavaScript Classes introduces the first formal concept of classes in JavaScript. Often a point of confusion for those coming from other languages, the addition of class syntax in JavaScript makes the language more approachable to others and more concise for enthusiasts.

Chapter 10: Improved Array Capabilities details the changes to native arrays and the interesting new ways they can be used in JavaScript.

Chapter 11: Promises and Asynchronous Programming introduces promises as a new part of the language. Promises were a grassroots effort that eventually took off and gained in popularity due to extensive library support. ECMAScript 6 formalizes promises and makes them available by default.

Chapter 12: Proxies and the Reflection API introduces the formalized reflection API for JavaScript and the new proxy object that allows you to intercept every operation performed on an object. Proxies give developers unprecedented control over objects and, as such, unlimited possibilities for defining new interaction patterns.

Chapter 13: Encapsulating Code with Modules details the official module format for JavaScript. The intent is that these modules can replace the numerous ad-hoc module definition formats that have appeared over the years.

Appendix A: Smaller ECMAScript 6 Changes covers other changes implemented in ECMAScript 6 that you'll use less frequently or that didn't quite fit into the broader major topics covered in each chapter.

Appendix B: Understanding ECMAScript 7 (2016) describes the two additions to the standard that were implemented for ECMAScript 7, which didn't impact JavaScript nearly as much as ECMAScript 6.

Conventions Used

The following typographical conventions are used in this book:

- *Italics* introduces new terms
- `Constant width` indicates a piece of code or filename

Additionally, longer code examples are contained in constant width code blocks such as:

```
function doSomething() {
    // empty
}
```

Within a code block, comments to the right of a `console.log()` statement indicate the output you'll see in the browser or Node.js console when the code is executed, for example:

```
console.log("Hi");      // "Hi"
```

If a line of code in a code block throws an error, this is also indicated to the right of the code:

```
doSomething();          // error!
```

Help and Support

You can file issues, suggest changes, and open pull requests against this book by visiting: https://github.com/nzakas/understandinges6

If you have questions as you read this book, please send a message to my mailing list: http://groups.google.com/group/zakasbooks.

Acknowledgments

Thanks to Jennifer Griffith-Delgado, Alison Law, and everyone at No Starch Press for their support and help with this book. Their understanding and patience as my productivity slowed to a crawl during my extended illness is something I will never forget.

I'm grateful for the watchful eye of Juriy Zaytsev as tech editor and to Dr. Axel Rauschmayer for his feedback and several conversations that helped to clarify some of the concepts discussed in this book.

Thanks to everyone who submitted fixes to the version of this book that is hosted on GitHub: ShMcK, Ronen Elster, Rick Waldron, blacktail, Paul Salaets, Lonniebiz, Igor Skuhar, jakub-g, David Chang, Kevin Sweeney, Kyle Simpson, Peter Bakondy, Philip Borisov, Shaun Hickson, Steven Foote, kavun, Dan Kielp, Darren Huskie, Jakub Narębski, Jamund Ferguson, Josh Lubaway, Marián Rusnák, Nikolas Poniros, Robin Pokorný, Roman Lo, Yang Su, alexyans, robertd, 404, Aaron Dandy, AbdulFattah Popoola, Adam Richeimer, Ahmad Ali, Aleksandar Djindjic, Arjunkumar, Ben Regenspan, Carlo Costantini, Dmitri Suvorov, Kyle Pollock, Mallory, Erik Sundahl, Ethan Brown, Eugene Zubarev, Francesco Pongiluppi, Jake Champion, Jeremy Caney, Joe Eames, Juriy Zaytsev, Kale Worsley, Kevin Lozandier, Lewis Ellis, Mohsen Azimi, Navaneeth Kesavan, Nick Bottomley, Niels Dequeker, Pahlevi Fikri Auliya, Prayag Verma, Raj Anand, Ross Gerbasi, Roy Ling, Sarbbottam Bandyopadhyay, and Shidhin.

Also, thanks to everyone who supported this book on Patreon: Casey Visco.