Extracted from:

Web Development with ReasonML

Type-Safe, Functional Programming for JavaScript Developers

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Web Development with ReasonML

Type-Safe, Functional Programming for JavaScript Developers

J. David Eisenberg

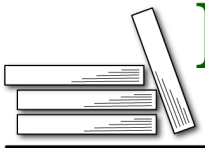
edited by Andrea Stewart

Web Development with ReasonML

Type-Safe, Functional Programming for JavaScript Developers

J. David Eisenberg

The Pragmatic Bookshelf
Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Susan Conant
Development Editor: Andrea Stewart
Copy Editor: Sean Dennis
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-633-4
Book version: P1.0—April 2019

Introduction

JavaScript has taken its place as a major programming language. It seems to be everywhere, has a large, flourishing ecosystem, and works both client- and server-side. It has effectively become a *lingua franca* for the Web. At the same time that JavaScript has grown, two trends in programming have grown in popularity for front-end web development: functional programming and static typing. Functional programming helps you avoid the problems that come with mutable data and mutable global state. Static typing moves many programming errors from runtime to compile time so your program's users never encounter them.

The result of this convergence has been the creation of functional programming libraries such as Lodash,[1] tools like Flow[2] that help with static typing, and functional, statically typed languages that compile to JavaScript, such as TypeScript,[3] PureScript,[4] Elm,[5] and ReasonML.[6]

What Makes ReasonML Special?

With all these choices, why choose ReasonML? First, ReasonML is not a new language that compiles to JavaScript. Rather, it is a new syntax for OCaml, an established language used in industry for over 20 years. With ReasonML, you get OCaml's strong static-type system with an excellent type inference engine as well as its features for functional programming with immutable variables.

For those times when you need to break away from the functional world, ReasonML allows "opt-in side-effect, mutation and object[s] for familiarity &

1. lodash.com/
2. flow.org/
3. www.typescriptlang.org/
4. www.purescript.org/
5. elm-lang.org/
6. reasonml.github.io/

interop[eration with JavaScript], while keeping the rest of the language pure, immutable and functional."[7] ReasonML hits the sweet spot between the purely theoretical world and the laissez-faire approach of JavaScript, but always with the emphasis on getting things done.

ReasonML's syntax has been designed to be familiar to JavaScript programmers, with features that let you create bindings to existing JavaScript libraries. The compilation is incredibly fast, and the code ReasonML produces is highly readable. You aren't restricted to compiling to JavaScript—because ReasonML is OCaml, you can also compile to native code.

There are combinations of languages and libraries that give you all these capabilities, but ReasonML has them all—it's a "one-stop shop" for your programming needs, and that's why it's special.

What Should You Know?

We're going to presume that you have some experience programming in JavaScript. On the client side, you are familiar with HTML and CSS and have some knowledge of programming the Document Object Model (DOM). It's useful but not necessary to have experience with a framework like React or Vue.

What's in This Book?

In Chapter 1, Make Your First ReasonML Project, on page ?, you'll learn to set up your system to compile and run short programs using variables, built-in data types, and operations on numbers and strings.

Chapter 2, Writing Functions, on page ? shows you how to write functions as you learn more about ReasonML's type inference. You'll also learn about currying, which allows you to apply only some of a function's arguments to create a new function.

In Chapter 3, Creating Your Own Data Types, on page ?, you'll be able to create your own data types which automatically take advantage of ReasonML's type inference engine. You'll also learn about the `option` data type, which ensures that your code handles both cases of an operation that may or may not succeed. This helps you avoid the dreaded null or undefined errors at run time.

Chapter 4, Interacting with Web Pages, on page ? will have you creating web pages that call on ReasonML code. You'll use the `bs-webapi` library to directly manipulate the DOM.

7. reasonml.github.io/docs/en/what-and-why.html

In Chapter 5, Using Collections, on page ?, you'll work with tuples, lists, and arrays, and find out how to use the map(), keep(), and reduce() functions to transform the data in these collections without needing loops.

While map(), keep(), and reduce() are powerful and useful, sometimes you need to customize the iteration through a collection. In Chapter 6, Repeating with Recursion, on page ?, you will see how recursion lets you do this customization quite elegantly. You will also find out how the use of a technique called tail recursion lets ReasonML optimize the JavaScript it produces for best performance.

In Chapter 7, Structuring Data with Records and Modules, on page ?, you'll learn to construct data types with multiple fields and values, much like (but not identical to) JavaScript objects. You'll also explore how the use of modules lets you avoid name collisions between fields of different record types.

Chapter 8, Connecting to JavaScript, on page ? returns to the world of JavaScript, showing you how to write bindings to existing JavaScript libraries so that they can be used in your ReasonML code. This is where you'll see how to deal with JavaScript objects.

Continuing further into JavaScript, Chapter 9, Making Applications with Reason/React, on page ? shows you how to use the React framework with ReasonML to build a basic single-page web application. You'll see that ReasonML has been designed to integrate well with React.

Acknowledgments

Thanks to Andrea Stewart, a great editor.

Thanks also to the people who contributed to OpenClipArt.org:[8]

- Cell phone in Chapter 2, Writing Functions, on page ? by user lifeincolour, keypad by user Startright
- Icons in Chapter 9, Making Applications with Reason/React, on page ? by user sixsixfive
- Animal drawings by users scout, deb53, juhele, and davidblyons
- Graphics for recursive drawing in Chapter 6, Repeating with Recursion, on page ? by users Arvin61r58, DooFi, Firkin, gnokii, and tomas_arad

Special thanks to the technical reviewers who made many useful and insightful suggestions: Arno Bastenhof, Massimiliano Bertinetti, Eduard Bondarenko,

8. openclipart.org/

Lewis Chung, Sam Elliott, Riza Fahmi, Avi Goel, Peter Hampton, Luca Mezzalira, Nick McGinness, António Monteiro, Khoa Nguyen, Carmelo Piccione, Kristof Semjen, and Gianluigi Spagnuolo. Thanks go to all the people who noted errata in the beta versions of this book, especially Peer Reynders for his many excellent comments. Also, thanks to the people in the Discord chat rooms who answered my many naïve questions.

Online Resources

You can download all the example source code for this book from its Pragmatic Bookshelf website.[9] You can also provide feedback by submitting errata entries.

If you're reading the book in PDF form, you can click the link above a code listing to view or download the specific examples.

Ready to get started? Great—let's begin.

9. pragprog.com/book/reasonml/