Extracted from:

Programming WebAssembly with Rust

Unified Development for Web, Mobile, and Embedded Applications

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Programming WebAssembly with Rust

Unified Development for Web, Mobile, and Embedded Applications

Kevin Hoffman
edited by Andrea Stewart

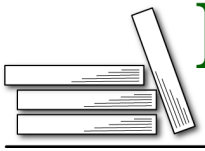
Programming WebAssembly with Rust

Unified Development for Web, Mobile, and Embedded Applications

Kevin Hoffman

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: Jasmine Kwityn
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: 978-1-68050-636-5
Book version: P1.0—March 2019

For my grandfather—Walter K. MacAdam—
inventor, tinkerer, and IEEE president. He
quietly supported my exposure to computers
and programming throughout my childhood,
often in ways I didn't know until after his death.
I always wanted to grow up to be like him, and
I only wish he could've seen this book.

Introduction

I'm old enough to have lived through quite a few seismic changes in the way developers build software and the kinds of products we can build. I was just starting my career when DPMI gave us native access to 32-bit integers, allowed unfettered access to a heap greater than 640k, and enabled the creation of ground-breaking games like *DOOM*. I remember the potential behind Java's promise of *write once, run anywhere*. I was there when small, local communities built around dial-up bulletin board systems (BBSes) faded as the world became a single, digital community riding the wave of the Internet's surge toward ubiquity. I experienced the shift in solution design from client/server to fat server to fat client and back again, today landing on *cloud native* applications, microservices, and independent functions where everything including our infrastructure is a service.

I remember the web's growth from a billion archipelagos of text (often blinking!) and Under Construction signs where the coolest places were the ones with the most intricate full-page background images, to the vast, sprawling engine of commerce, communication, lifestyle, and social connection that it is today. The web has gone from a place where only an elite few dabbled in that strange new world to a place where millions of people spend their days coding some of the most powerful and complex applications of the modern era.

I firmly believe that we stand on the precipice of another seismic change—*WebAssembly*. This new technology holds within it the potential to radically change how developers build applications for the web. Moreover, as you'll see throughout this book, WebAssembly is more than just a new pebble thrown into the ocean of web technologies. It's a tsunami that can change not only how consumers interact with and how developers build applications but also fundamentally alter the kinds of applications we can create. It may even transform our core definition of the word *application*.

Today's Web Technology

Today's web is a veritable playground for developers. You have the luxury of easy access to broadband speeds (with exceptions). Browsers are faster and more powerful than they've ever been, and the workstations people use to run those browsers have oodles of RAM, storage, and multiple cores—even the mobile devices.

Today's JavaScript is nothing like the primordial 1995 JavaScript that drew so much ire from the developer community. It dominates the web development landscape so much that its own ubiquity has become something of a joke or a meme. Competition between browser vendors (they didn't call it the *browser wars* for nothing) has spurred years of refining the way their products execute JavaScript, which will be a key discussion point when you get into the details of browser-hosted WebAssembly.

Modern browsers have virtual machines responsible for JavaScript execution. Internally they optimize and produce a form of bytecode from processed JavaScript. This, coupled with more memory and processing power, means that JavaScript is actually *fast*. Not just a little bit fast, but it's so fast you can play full, grade-A video games in the browser. Applications can perform complex calculations, run machine learning models, process vast amounts of data, and otherwise treat the browser like an operating system.

Frameworks like *React*, *Angular*, *Backbone*, *VueJS*, and countless others have made a dramatic impact on how web applications are built. Modern web applications can render incredibly dense user interfaces like what you see on Facebook or YouTube, all while receiving real-time events published from servers in the cloud to provide a level of interaction that's now such a ubiquitous feature that web sites that don't provide this new level of real-time interaction are often publicly shamed and doomed to fail.

Nothing in the rest of this book that extols the virtues of WebAssembly should take away from the fact that the modern, programmable web is a giant virtual toy store, ripe for the plundering by eager developers. For any avid learner of technology, it's a great time to be alive (and probably learn some JavaScript).

The Tech of Tomorrow

WebAssembly is currently a 1.0 product, having just reached its first MVP (Minimum Viable Product). As with most 1.0 products, it's bound to experience

some growing pains and points of friction, and we'll go over those in depth as they come up in this book. As you look at the current state of WebAssembly and its limitations, you might get discouraged and feel the urge to give up and wait for things to get more mature. But I think the time is right to start learning and developing with this incredible new technology, and there are already many WebAssembly 1.0 products deployed and running in the wild and more appearing every day. In the span between two edits of this chapter, someone released a virtual machine built in WebAssembly that runs Windows 95 in a browser.

The good news is that the experience will only improve over time. The tooling will get better, the interface between the browser and WebAssembly modules will get better, support for non-browser hosts will get better, and the number of tested and proven use cases will grow. In short, as time goes on, every aspect of the development of WebAssembly modules will improve.

I am convinced that WebAssembly is at the tip of the next wave of truly paradigm-shifting changes in the programmable Internet. I did indeed mean to say *Internet* and not just *web*. The distinction might seem subtle, but I'm also thoroughly convinced that the browser as a host for WebAssembly modules is just the tip of the iceberg. WebAssembly is going to join the long line of game-changing innovations in the history of the Internet and fundamentally alter our concept of applications.

Who This Book Is For

This book is for anyone who wants to build web applications. Whether you have had just a little bit of JavaScript exposure or whether you are a seasoned professional with dozens of React and Redux applications under your belt, WebAssembly has the potential to change the way that you build apps and the power of those applications in a way that few technologies before ever have.

Whether you consider yourself a front-end, back-end, embedded, or any other kind of developer—this book is for you. Compiling other languages to WebAssembly means you get to use familiar development life cycles and toolchains and build and test strongly-typed, powerful code.

Finally, if you think that there is more to this WebAssembly thing than just the web applications, then you will enjoy this book as well as we build WebAssembly interpreters in Rust and run them on Raspberry Pis to control

hardware via GPIO. WebAssembly holds a lot of promise for many different types of developers, including the promise of unifying back- and front-end coding experiences.

Why Rust?

Rust is a systems language that compiles to native binaries on any number of operating systems and hardware architectures. It is fast, its binaries take up very little space and have a small memory footprint, and is designed from the ground up to avoid accidental mutation, null referencing, and data races. In fact, the compiler will check your code and prevent you from making those mistakes.

But I chose Rust for this book for reasons beyond just the language syntax and its powerful compiler. What excited me about Rust was how quickly it embraced WebAssembly. While other languages right now can compile code to WebAssembly, the sheer number of libraries and tools available within the Rust community for WebAssembly is staggering. It is the enthusiasm, support, and rapid pace of advancement in the Rust WebAssembly community that influenced my decision to use Rust for this book.

What You'll Learn

This book is divided up into three main parts:

Building a Foundation

As you build a foundation, you'll learn the fundamentals and the core architecture of WebAssembly, including what it can and cannot do and how you can develop basic applications. By the time you reach the end of this section, you'll be able to create a Checkers engine written entirely in raw WebAssembly.

Interacting with JavaScript

Building on your solid WebAssembly foundation, you'll move on to using Rust to create your WebAssembly modules. You'll start with the basics of creating a Rust version of your Checkers engine, and then you'll move on to using code generation, advanced tooling, and macros to build powerful web applications that interact with JavaScript. By the end of this section, you'll be able to write a multiuser, real-time chat application in Rust that compiles to WebAssembly.

Working with Non-Web Hosts

Once you've had your first taste of the power of WebAssembly, it's time to take it to the next level and start working with non-web hosts.

WebAssembly is about far more than just building things for the web, and you'll see this firsthand as you create modules that control LED patterns for lights attached to a Raspberry Pi and, as your final project, you create a fully multiplayer arena battle game that lets developers pit their WebAssembly code against each other in a battle to the death.

Now it's time to get coding!