

# AssemblyScript

**AssemblyScript** is a TypeScript-based programming language (essentially JavaScript with static types) that is optimized for WebAssembly and compiled to WebAssembly using asc, the reference AssemblyScript compiler. It is developed by the AssemblyScript Project<sup>[6]</sup> and the AssemblyScript community.

## Contents


- Overview
- Compatibility with JavaScript
- Usage
- Reception
- See also
- References
- External links

## Overview

In 2017, the availability of support for WebAssembly, a standard definition for a low-level bytecode and an associated virtual machine, became widespread among major Web browsers, providing Web developers a lower-level and potentially higher-performance compilation target for client-side programs and applications to execute within Web browsers, in addition to the interpreted (and also typically dynamically compiled) JavaScript Web scripting language.<sup>[7]</sup> WebAssembly allows programs and code to be statically compiled ahead of time in order to run at potentially native-level or “bare-metal” performance within Web browsers, without the overhead of interpretation or the initial latency of dynamic compilation.<sup>[8]</sup>

With the adoption of WebAssembly in major Web browsers, Alon Zakai, creator of Emscripten, an LLVM/Clang-based C and C++ compiler that targeted a subset of JavaScript called asm.js, added support for WebAssembly as a compilation target in Emscripten, allowing C and/or C++ programs and code to be compiled directly to WebAssembly.<sup>[9]</sup>

While Emscripten and similar compilers allow Web developers to write new code, or port existing code, written in a high-level language such as C, C++, Go, and Rust to WebAssembly to

AssemblyScript	
	
<b>Paradigm</b>	Multi-paradigm: <u>functional</u> , <u>generic</u> , <u>imperative</u> , <u>object-oriented</u>
<b>Family</b>	<u>JavaScript</u> , <u>TypeScript</u>
<b>Designed by</b>	Daniel Wirtz, <sup>[1]</sup> Max Graey <sup>[2]</sup>
<b>Developer</b>	The AssemblyScript Project <sup>[1]</sup>
<b>First appeared</b>	2017 <sup>[3]</sup>
<b>Stable release</b>	0.18.29 / April 28, 2021 <sup>[4]</sup>
<b>Typing discipline</b>	<u>Static</u>
<b>Scope</b>	<u>Lexical</u>
<b>License</b>	<u>Apache License 2.0</u> <sup>[5]</sup>
<b>Filename extensions</b>	<u>.ts</u>
<b>File formats</b>	<u>Text</u> , <u>Unicode</u> (source) <u>.wasm</u> binary format (object)
<b>Website</b>	<u>www</u> <u>.assemblyscript</u> <u>.org</u> ( <u>https://www.a</u>

achieve potentially higher, native-level execution performance in Web browsers, this forces Web developers accustomed to developing client-side Web scripts and applications in JavaScript (the *de facto* client-side programming language in Web browsers) to use a different language for targeting WebAssembly than JavaScript. AssemblyScript, as a variant of TypeScript that is syntactically nearly identical to JavaScript, allows developers accustomed to JavaScript to use a familiar language for targeting WebAssembly, potentially reducing the learning curve of a separate language that can be compiled to WebAssembly.

Furthermore, because AssemblyScript was designed to be an optimal source language for WebAssembly, the language's type system closely reflects that of WebAssembly,<sup>[10]</sup> and the language provides standard low-level functions (typically implemented as macros) that map directly to WebAssembly instructions that mirror specialized hardware instructions provided by modern processors such as SIMD and vector instructions and more specialized instructions such as clz (count leading zero bits), ctz (count trailing zero bits), and popcnt (population count), used in applications such as encryption and cryptographic libraries.<sup>[11]</sup>

The AssemblyScript compiler `asc` is based on Binaryen, a back-end compiler toolchain developed by Alon Zakai that compiles to WebAssembly and is a component of Emscripten (which Zakai also developed). The `asc` compiler and other tooling are available via the npm package manager.

While WebAssembly was originally designed for execution within Web browsers, the development of WASI (WebAssembly System Interface), a community specification for a standard API that allows WebAssembly programs access to system calls and other operating system functions,<sup>[12]</sup> has led to the development of WebAssembly runtime environments from projects such as Wasmtime<sup>[13]</sup> and Wasmer<sup>[14]</sup> that allow WebAssembly, and code written in languages such as AssemblyScript that can compile to it, to run in non-Web environments as well.

## Compatibility with JavaScript

---

AssemblyScript is compiled to WebAssembly modules, which can then be instantiated into client-side Web pages using standard JavaScript methods such as `WebAssembly.compileStreaming` and `WebAssembly.instantiateStreaming` just like standard WebAssembly binaries.<sup>[15]</sup> Data passing between JavaScript and the compiled WebAssembly modules, as well as function calls between JavaScript and WebAssembly, are then the same as for any WebAssembly module.<sup>[16]</sup>

Because the AssemblyScript language is mostly a subset of TypeScript (it also has a small superset of TypeScript), it is possible to write an AssemblyScript program using the subset and compile it to both plain JavaScript and WebAssembly, using both the TypeScript compiler and AssemblyScript compiler, respectively. This allows for portable code that runs in both JavaScript and WebAssembly environments.

## Usage

---

As of March 2021, approximately 3000 projects hosted on GitHub are written, either wholly or partially, in AssemblyScript,<sup>[17]</sup> with roughly 5000 downloads of the AssemblyScript compiler per week via npm.<sup>[18]</sup>

## Reception

---

<u>ssemblyscript.org/</u>
Major implementations
<code>asc</code> (AssemblyScript compiler)
Influenced by
<u>JavaScript</u> , <u>TypeScript</u> , <u>WebAssembly</u>

Lead Emscripten developer Alon Zakai has characterized AssemblyScript as being “designed with WebAssembly and code size in mind. It’s not an existing language that we are using for a new purpose, but it’s a language designed for WebAssembly. It has great `wasm-opt` integration—in fact, it’s built with it—and it’s very easy to get good code size.”<sup>[19]</sup>

Norwegian musician Peter Salomonsen, in a 2020 WebAssembly Summit talk titled, “WebAssembly Music,” demonstrated the use of AssemblyScript for real-time compilation to WebAssembly in live electronic music synthesis, saying, “I chose AssemblyScript because it has high-level readability and low-level control; it’s like a high-level language, but you get that low-level feeling, and you can even write direct WebAssembly intrinsics if you want to.”<sup>[20]</sup>

Aaron Turner, a senior engineer at Fastly, a cloud computing services provider that uses WebAssembly for the company’s Compute@Edge serverless compute environment, in a review of AssemblyScript wrote:<sup>[21]</sup>

While AssemblyScript requires stricter typing than TypeScript does, it sticks as close as possible to TypeScript syntax and semantics—which means that most JavaScript developers will find AssemblyScript comfortable to use—and it enables great support for the modern JavaScript ecosystem. For instance, the AssemblyScript compiler is available on `npm`, as well as common AssemblyScript tools and libraries like `as-pect`. AssemblyScript files also use TypeScript’s ‘`.ts`’ file extension, and it includes proper typings for allowing AssemblyScript to piggy-back on TypeScript tooling, such as the TypeScript linter. With the right small tweaks, AssemblyScript can even be used with the TypeScript compiler.

This is very exciting, as AssemblyScript offers a low-overhead entry-point for JavaScript developers to pick up a language to output WebAssembly—both in terms of learning to read and write AssemblyScript, as well as using a lot of the pre-existing tooling that may already be in a JavaScript developer’s workflow. AssemblyScript is often referred to in the WebAssembly community as a great gateway to picking up WebAssembly. It offers a large group of developers who already write applications for the web a path to pick up and learn WebAssembly. Even if you are starting from scratch and are not particularly familiar with JavaScript or TypeScript, AssemblyScript is a solid choice when picking a language to start outputting WebAssembly.

However, Turner went on to cite the language’s relative newness and thus its lack of some features available in larger, more complex and established programming languages as potential shortcomings of the language.

## See also

---

- Emscripten
- JavaScript
- TypeScript
- WebAssembly

## References

---

1. The AssemblyScript Project (April 24, 2020). “AssemblyScript Working Group” (<https://github.com/AssemblyScript/working-group>). *GitHub.com*. AssemblyScript Project. Retrieved February 10, 2021. “Daniel Wirtz (@dcodeIO) - Author of AssemblyScript”

2. Aaron Turner (March 28, 2019). "WebAssembly for Javascript Developers" (<https://www.youtube.com/watch?v=ZIL1nduatZQ>). *YouTube.com*. WebAssembly SF. Retrieved February 10, 2021. "@dcodeIO [Daniel Wirtz] and @MaxGraey [Max Graey]—they're the main two developers of AssemblyScript"
3. The AssemblyScript Project (2017). "assemblyscript 0.1.0" (<https://www.npmjs.com/package/assemblyscript/v/0.1.0>). *npmjs.com*. AssemblyScript Project. Retrieved February 10, 2021.
4. The AssemblyScript Project (April 28, 2021). "assemblyscript" (<https://www.npmjs.com/package/assemblyscript>). *npmjs.com*. The AssemblyScript Project. Retrieved April 28, 2021.
5. The AssemblyScript Project (September 29, 2017). "LICENSE" (<https://github.com/AssemblyScript/assemblyscript/blob/master/LICENSE>). *GitHub.com*. The AssemblyScript Project. Retrieved February 10, 2021. "AssemblyScript/assemblyscript is licensed under the Apache License 2.0"
6. The AssemblyScript Project. "The AssemblyScript Project" (<https://github.com/AssemblyScript>). *GitHub.com*. The AssemblyScript Project. Retrieved February 10, 2021.
7. WebAssembly Community Group (November 2017). "Roadmap" (<https://webassembly.org/roadmap/>). *WebAssembly.org*. WebAssembly Community Group. Retrieved February 10, 2021.
8. WebAssembly Working Group. "WebAssembly" (<https://webassembly.org/>). *WebAssembly.org*. WebAssembly Working Group. Retrieved February 10, 2021.
9. Alon Zakai and Emscripten Contributors. "Emscripten" (<https://emscripten.org/>). *Emscripten.org*. Emscripten project. Retrieved February 10, 2021.
10. The AssemblyScript Project. "Types" (<https://www.assemblyscript.org/types.html>). *AssemblyScript.org*. The AssemblyScript Project. Retrieved February 10, 2021.
11. The AssemblyScript Project. "Environment" (<https://www.assemblyscript.org/environment.html>). *AssemblyScript.org*. The AssemblyScript Project. Retrieved February 10, 2021.
12. The Wasmtime Project. "WASI: The WebAssembly System Interface" (<https://wasi.dev/>). *WASI.dev*. The Wasmtime Project. Retrieved February 10, 2021.
13. The Wasmtime Project. "Wasmtime: A small and efficient runtime for WebAssembly & WASI" (<https://wasmtime.dev/>). *Wasmtime.dev*. The Wasmtime Project. Retrieved February 10, 2021.
14. Wasmer, Inc. "Wasmer: The Universal WebAssembly Runtime" (<https://wasmer.io/>). *Wasmer.io*. Wasmer, Inc. Retrieved February 10, 2021.
15. Mozilla Developer Network. "Loading and running WebAssembly code" ([https://developer.mozilla.org/en-US/docs/WebAssembly/Loading\\_and\\_running](https://developer.mozilla.org/en-US/docs/WebAssembly/Loading_and_running)). *developer.mozilla.org*. Mozilla Developer Network. Retrieved February 10, 2021.
16. Mozilla Developer Network. "Using the WebAssembly JavaScript API" ([https://developer.mozilla.org/en-US/docs/WebAssembly/Using\\_the\\_JavaScript\\_API](https://developer.mozilla.org/en-US/docs/WebAssembly/Using_the_JavaScript_API)). *developer.mozilla.org*. Mozilla Developer Network. Retrieved February 10, 2021.
17. GitHub. "AssemblyScript/assemblyscript Dependency graph" ([https://github.com/AssemblyScript/assemblyscript/network/dependents?package\\_id=UGFja2FnZS0xODY0NzM4NQ%3D%3D](https://github.com/AssemblyScript/assemblyscript/network/dependents?package_id=UGFja2FnZS0xODY0NzM4NQ%3D%3D)). *GitHub.com*. *GitHub*. Retrieved March 11, 2021.
18. npmjs. "assemblyscript Weekly Downloads" (<https://www.npmjs.com/package/assemblyscript>). *npmjs.com*. *npmjs.com*. Retrieved March 11, 2021.
19. Alon Zakai (February 19, 2020). "Shipping Tiny WebAssembly Builds" ([https://www.youtube.com/watch?v=\\_ILqZR4ufSI](https://www.youtube.com/watch?v=_ILqZR4ufSI)). *YouTube.com*. WebAssembly-Summit.org. Retrieved February 10, 2021.
20. Peter Salomonsen (February 19, 2020). "WebAssembly Music" ([https://www.youtube.com/watch?v=C8j\\_ieOm4vE](https://www.youtube.com/watch?v=C8j_ieOm4vE)). *YouTube.com*. WebAssembly-Summit.org. Retrieved February 10, 2021.

21. Aaron Turner (October 29, 2020). "Meet AssemblyScript: your next computing language" (<https://www.fastly.com/blog/meet-assemblyscript-your-next-computing-language>). *Fastly.com*. *Fastly*. Retrieved February 10, 2021.

## External links

---

- [AssemblyScript.org \(https://www.assemblyscript.org/\)](https://www.assemblyscript.org/) (official site)
  - [AssemblyScript documentation \(https://www.assemblyscript.org/introduction.html\)](https://www.assemblyscript.org/introduction.html) (project page)
  - [The AssemblyScript Project \(https://github.com/AssemblyScript/\)](https://github.com/AssemblyScript/) (on GitHub)
  - [assemblyscript \(https://www.npmjs.com/package/assemblyscript\)](https://www.npmjs.com/package/assemblyscript) (on npm)
- 

Retrieved from "<https://en.wikipedia.org/w/index.php?title=AssemblyScript&oldid=1020302854>"

---

**This page was last edited on 28 April 2021, at 10:30 (UTC).**

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.