DOCUMENTATION

FAQ

WebAssembly High-Level Goals

Use Cases

Portability

Security

Web Embedding

Non-Web Embeddings

Tooling support

Non-Web Embeddings

While WebAssembly is designed to run on the Web, it is also desirable for it to be able to execute well in other environments, including everything from minimal shells for testing to full-blown application environments e.g. on servers in datacenters, on IoT devices, or mobile/desktop apps. It may even be desirable to execute WebAssembly embedded within larger programs.

Non-Web environments may provide different APIs than Web environments, which feature testing and dynamic linking will make discoverable and usable.

Non-Web environments may include JavaScript VMs (e.g. node.js), however WebAssembly is also being designed to be capable of being executed without a JavaScript VM present.

The WebAssembly spec itself will not try to define any large portable libc-like library. However, certain features that are core to WebAssembly semantics that are similar to functions found in native libc *would* be part of the core WebAssembly spec as primitive operators (e.g., the `grow_memory` operator, which is similar to the `sbrk` function on many systems, and in the future, operators similar to `dlopen`).

Where there is overlap between the Web and popular non-Web environments, shared specs could be proposed, but these would be separate from the WebAssembly spec. A symmetric example in JavaScript would be the in-progress Loader spec, which is proposed for both Web and node.js environments and is distinct from the JavaScript spec.

However, for most cases it is expected that, to achieve portability at the source code level, communities would build libraries that mapped from a source-level interface to the host environment's builtin capabilities (either at build time or runtime). WebAssembly would provide the raw building blocks (feature testing, builtin modules and dynamic loading) to make these libraries possible. Two early expected examples are POSIX and SDL.

In general, by keeping the non-Web path such that it doesn't require Web APIs, WebAssembly could be used as a portable binary format on many platforms, bringing great benefits in portability, tooling and language-agnosticity (since it supports C/C++ level semantics).