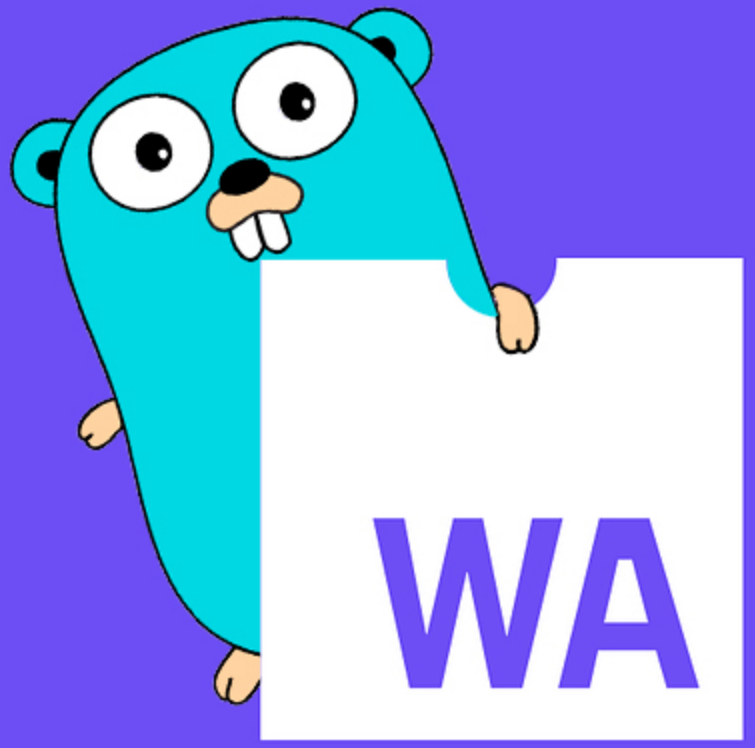
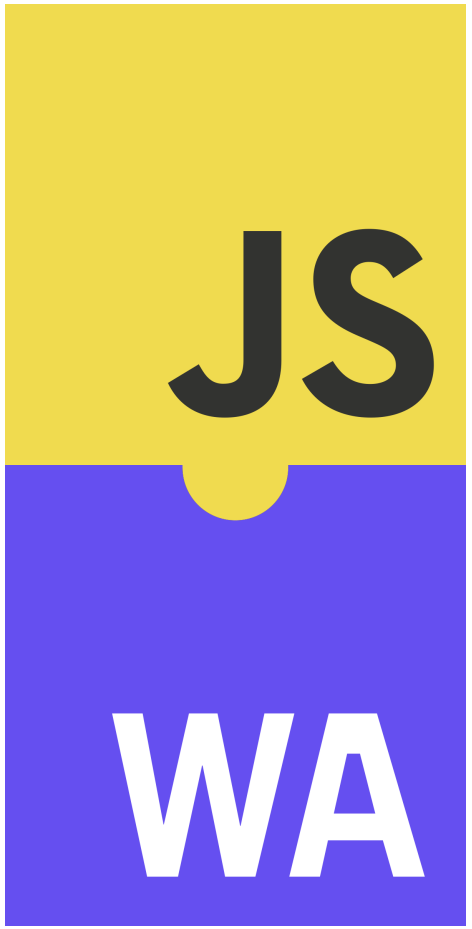
LET'S GO WEBASSEMBLY



LET'S GO WEBASSEMBLY

- A little bit of history
- WebAssembly
- WASM and Go

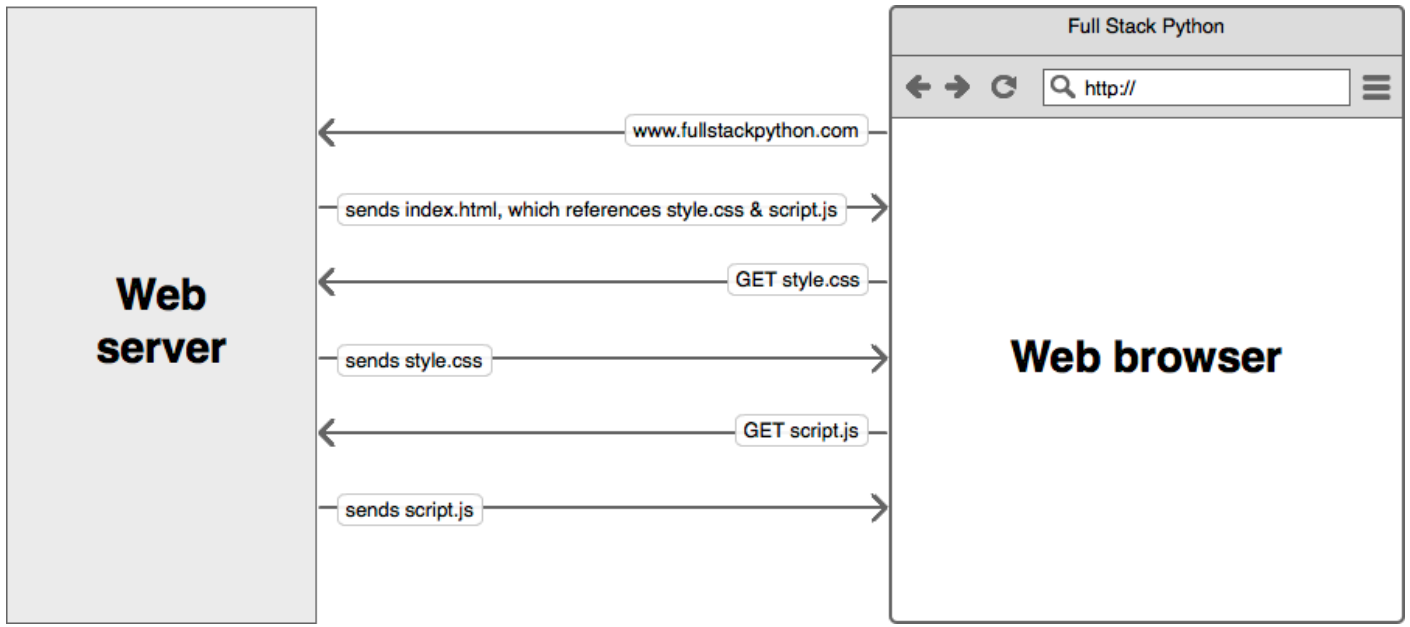


- Fast
- Zero configuration
- Secure
- Cross-platform
- Dev friendly

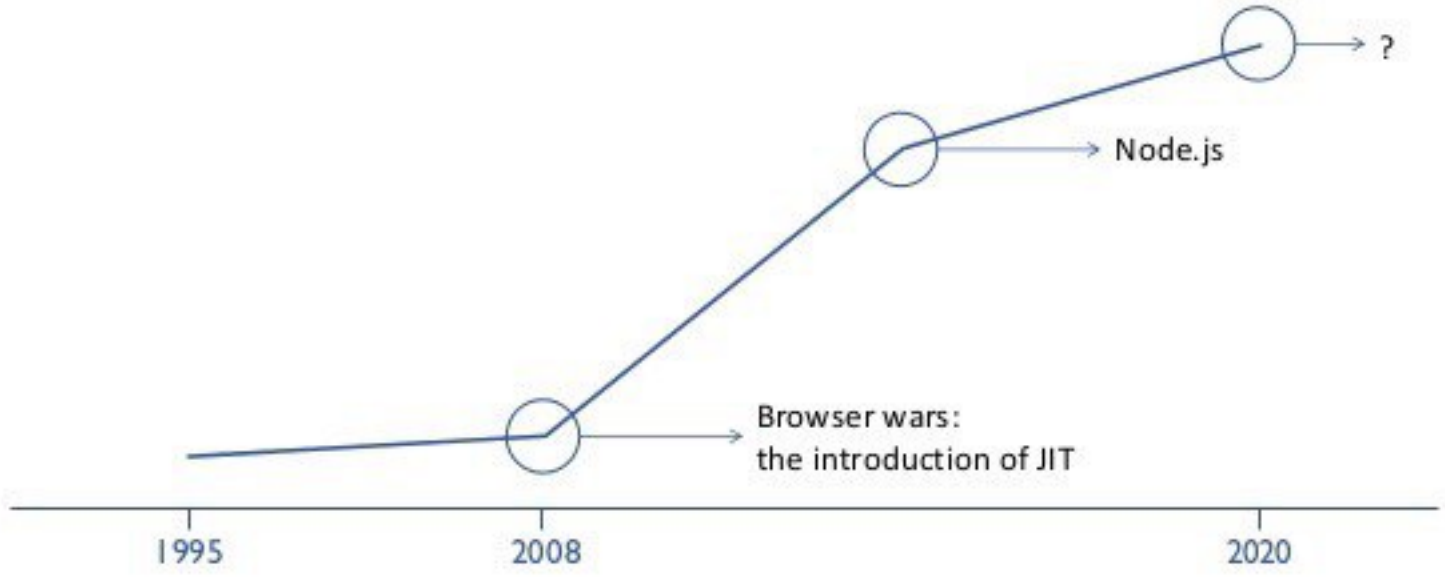
http://

HTML

CSS

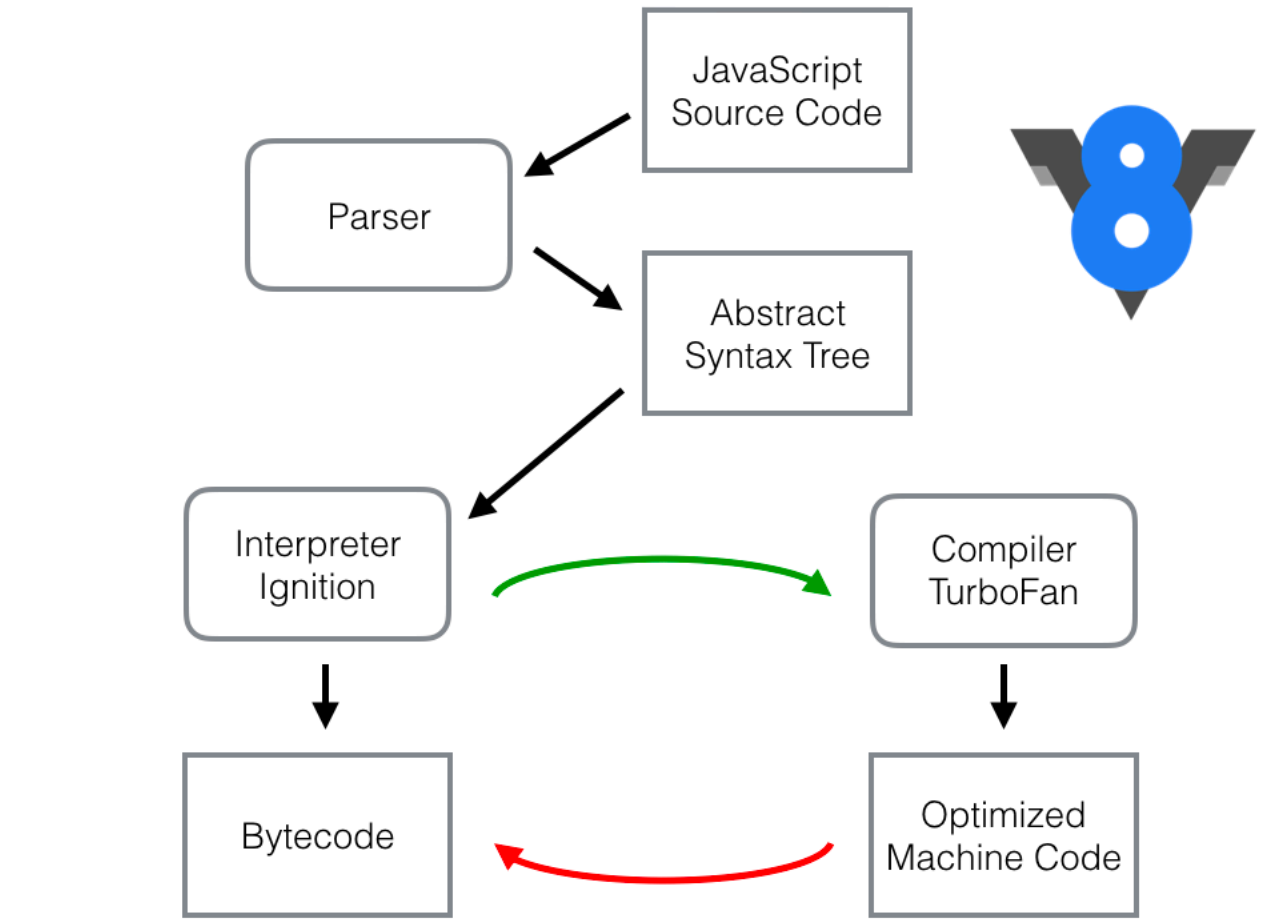


JavaScript performance over the years



@bmihaylov


npm
TS
REQUIRE·JS
BABEL
JS
A
npm
TS
REQUIRE·JS



Google

@fhinkel

ActiveX

is a software framework created by Microsoft that adapts its earlier Component Object Model (COM) and Object Linking and Embedding (OLE) technologies for content downloaded from a network, particularly from the World Wide Web.

Microsoft

Flash

Adobe Flash is a multimedia software platform used for production of animations, rich Internet applications, desktop applications, mobile applications, mobile games and embedded web browser video players.

https://www.apple.com/hotnews/thoughts-on-flash/

Adobe

Enjoy our **free online games** and **free games**. Play hundreds of **addicting games**, **funny games** and much more.

SET AS HOMEPAGE
ADD TO FAVORITES

SmashinGames

Play Free Online Games

Search

LATEST | TOP RATED | RANDOM | A-Z LIST | DRIVING | GIRLS

Latest Games

NaCl
PNaCl

Google Native Client (**NaCl**) is a sandboxing technology for running either a subset of Intel x86, ARM, or MIPS native code, or a portable executable, in a sandbox. It allows safely running native code from a web browser, independent of the user operating system, allowing web apps to run at near-native speeds,

Portable Native Client (PNaCl) is an architecture-independent version. PNaCl apps are compiled ahead-of-time. PNaCl is recommended over NaCl for most use cases.[1]

asm.js

asm.js is a subset of JavaScript designed to allow computer software written in languages such as C to be run as web applications while maintaining performance characteristics considerably better than standard JavaScript, which is the typical language used for such applications.

C code

```c
int f(int i) {
    return i + 1;
}
```

JS code

```js
function f(i) {
    i = i|0;
    return (i + 1)|0;
}
```

moz://a

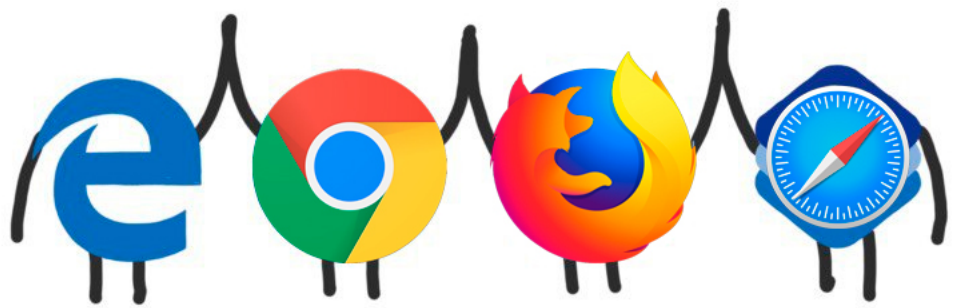

Google

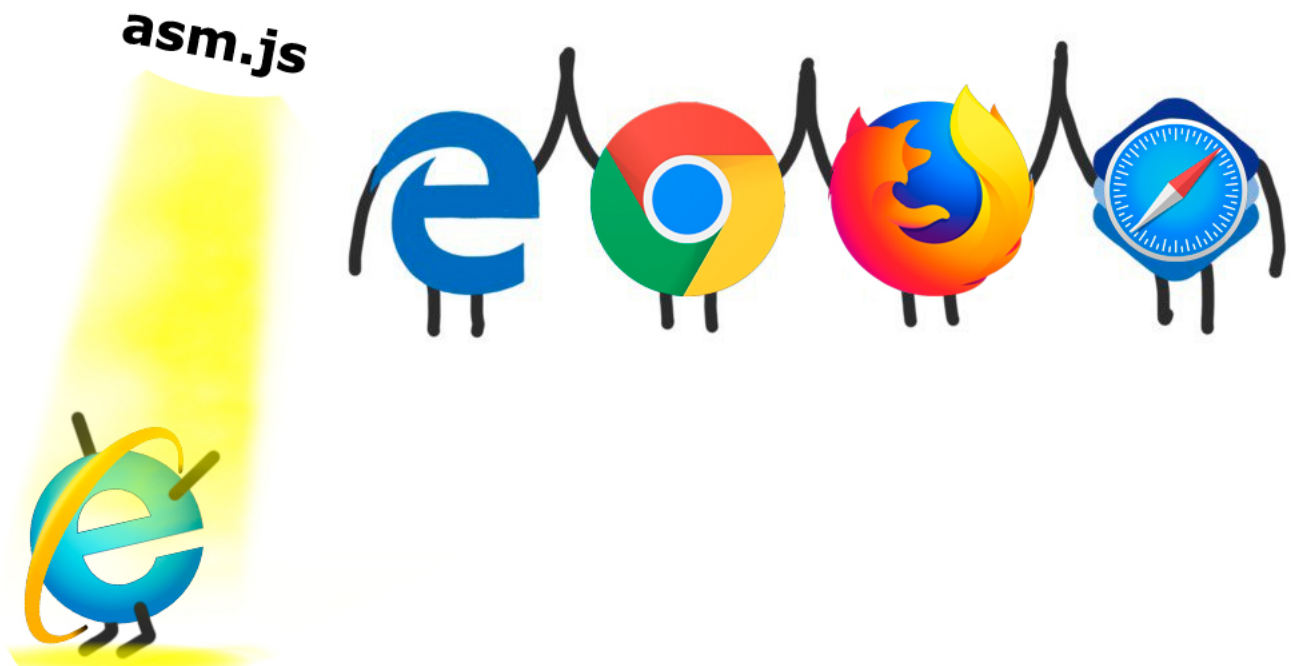

WA


Microsoft


moz://a


WebKit
Open Source Web Browser Engine



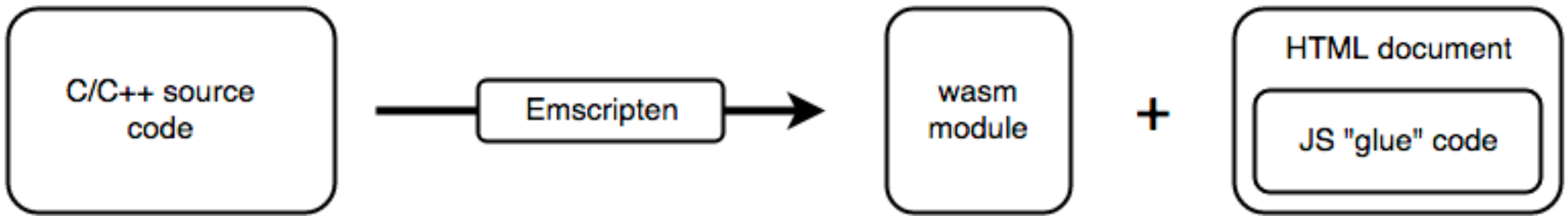

asm.js

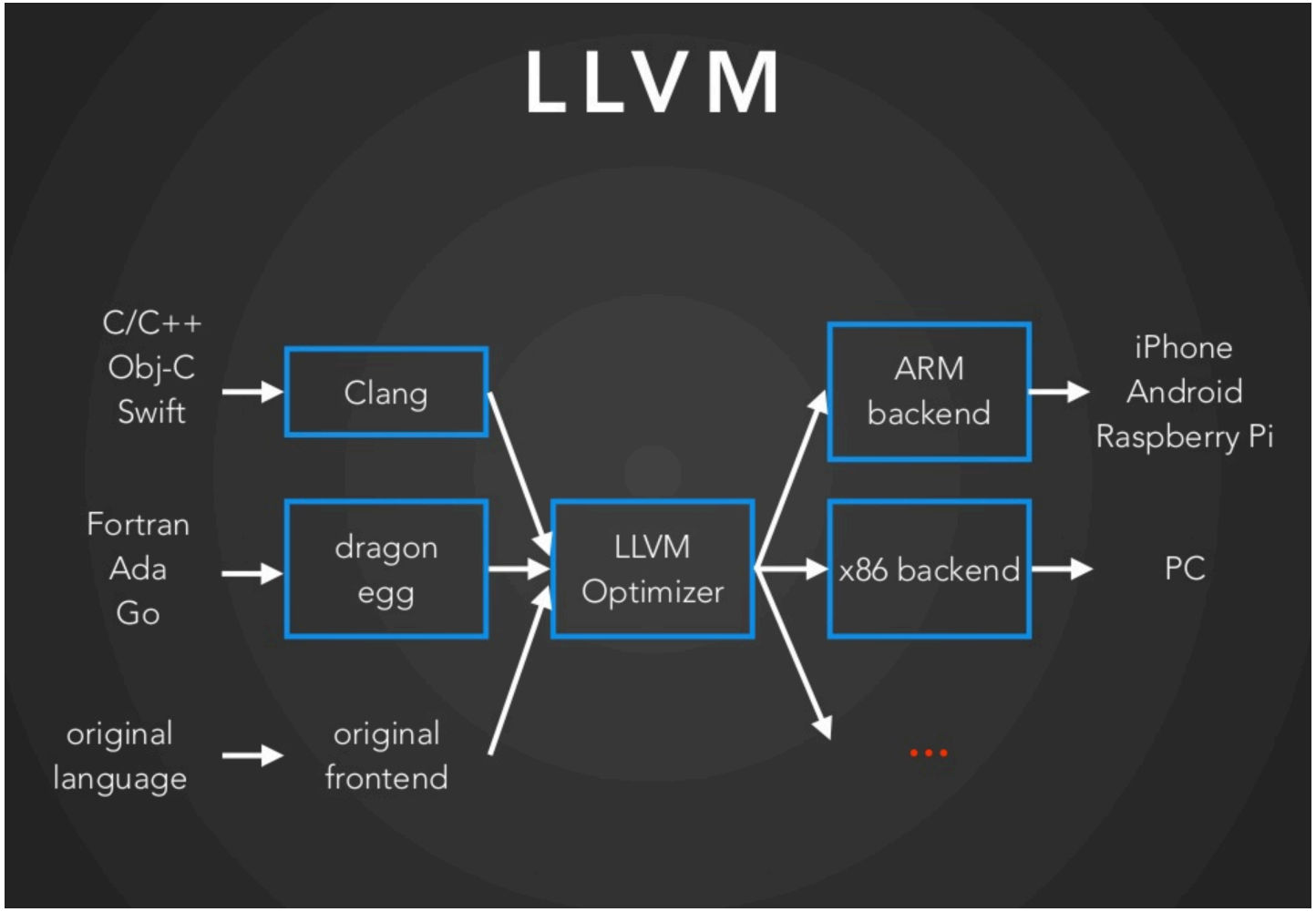
Browser compatibility

Update compatibility data on GitHub

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet	Node.js
`WebAssembly`	57	16	52 *	No	44	11	57	57	52 *	Yes	11	7.0	8.0.0
`CompileError`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`Global`	69	No	62	No	No	No	69	69	62	No	No	10.0	No
`Instance`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`LinkError`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`Memory`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`Module`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`RuntimeError`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`Table`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`compile`	57	16	52 *	No	44	11	57	57	52 *	43	11	7.0	8.0.0
`compileStreaming`	61	16	58	No	47	No	61	61	58	?	No	8.0	No
`instantiate`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0
`instantiateStreaming`	61	16	58	No	47	No	61	61	58	?	No	8.0	No
`validate`	57	16	52 *	No	44	11	57	57	52 *	?	11	7.0	8.0.0


WA
Ah shit, here we go again.


C/C++ source code
Emscripten
wasm module
+
HTML document
JS "glue" code


emscripten
LLVM
C/C++
Obj-C
Swift
Clang
Fortran
Ada
Go
dragon
egg
original
language
original
frontend
LLVM
Optimizer
ARM
backend
x86 backend
...
iPhone
Android
Raspberry Pi
PC

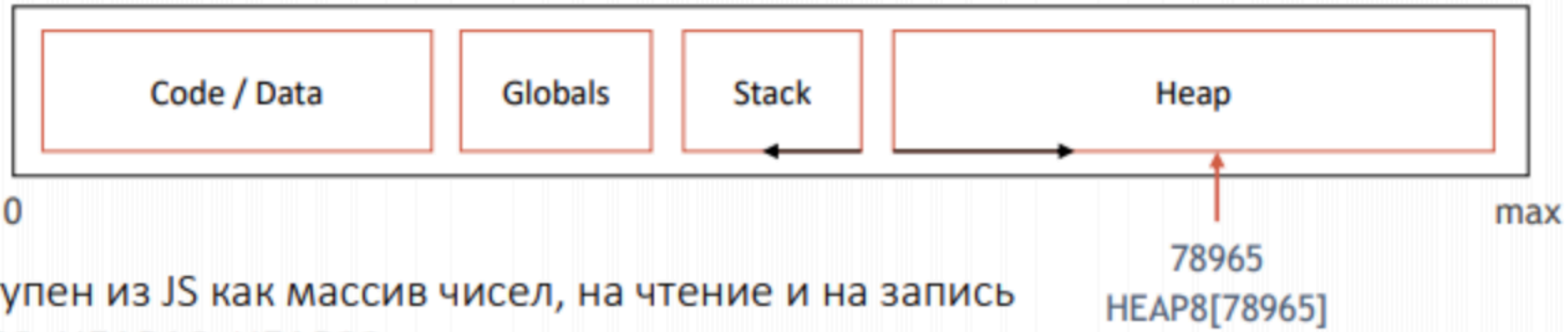
https://wasdk.github.io/WasmFiddle/

C++	Binary	Text

`int factorial(int n) {`
`  if (n == 0)`
`  return 1;`
`  else`
`  return n * factorial(n-1);`
`}`
	`20 00`
	`42 00`
	`51`
	`04 7e`
	`42 01`
	`05`
	`20 00`
	`20 00`
	`42 01`
	`7d`
	`10 00`
	`7e`
	`0b`
		`get_local 0`
		`i64.const 0`
		`i64.eq`
		`if i64`
		`  i64.const 1`
		`else`
		`  get_local 0`
		`  get_local 0`
		`  i64.const 1`
		`  i64.sub`
		`  call 0`
		`  i64.mul`
		`end`

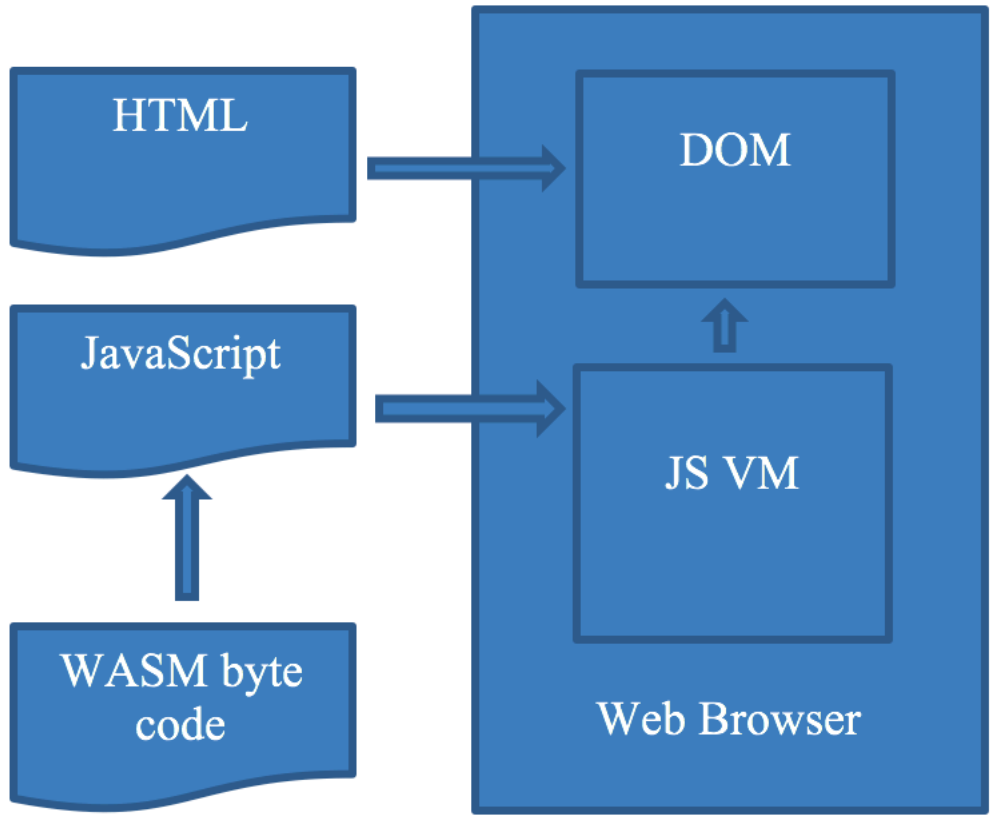
ВИРТУАЛЬНАЯ МАШИНА WASM: ПАМЯТЬ

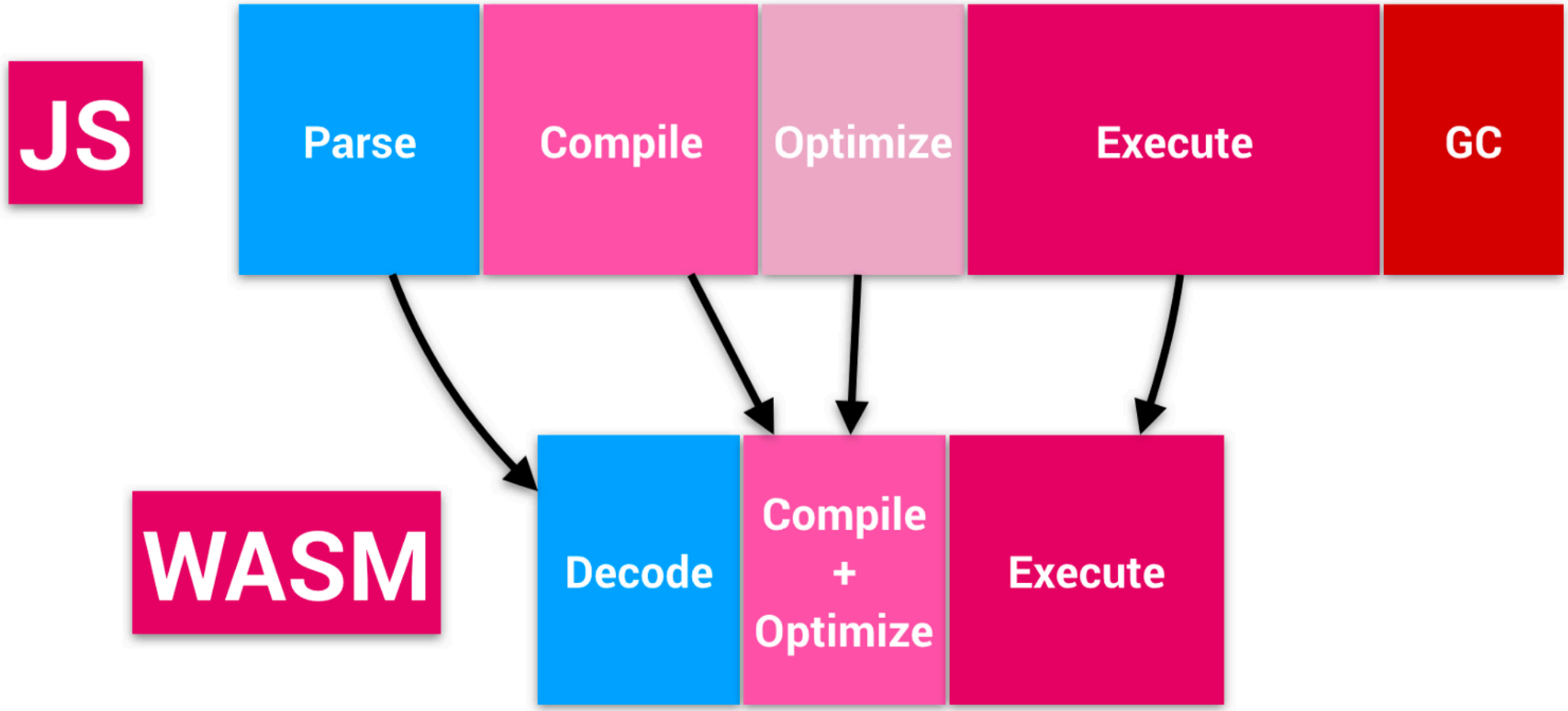
Плоская модель памяти, выделяется по 64КБ
Один блок памяти – программа, глобальные переменные, стек, куча
Расширяемый блок (ALLOW_MEMORY_GROW)

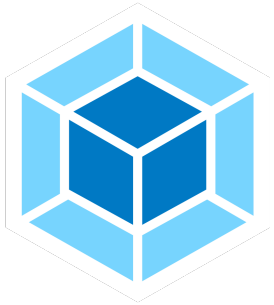


Доступен из JS как массив чисел, на чтение и на запись
HEAP8, HEAP16, HEAP32
HEAPU8, HEAPU16, HEAPU32
HEAPF32, HEAPF64
Нет указателей, только индексы, адресуется до 4ГБ

@nzeemin


HTML
JavaScript
WASM byte code
DOM
JS VM
Web Browser


JS
Parse
Compile
Optimize
Execute
GC
WASM
Decode
Compile + Optimize
Execute



webpack

_entry.js Copy Raw

```
1 import("./abc.js").then(abc => abc.doIt());
```

abc.js Copy Raw

```
1 import { add } from "./addition.wat";
2
3 export function doIt() {
4   console.log(add(1, 2));
5 }
```

addition.wat Copy Raw

```
1 (module
2  (func
3   (export "add")
4   (param i32 i32)
5   (result i32)
6   (i32.add (get_local 0) (get_local 1))
7  )
8 )
```

WebAssembly Package Manager

The wapm ecosystem makes WebAssembly more accessible to developers.

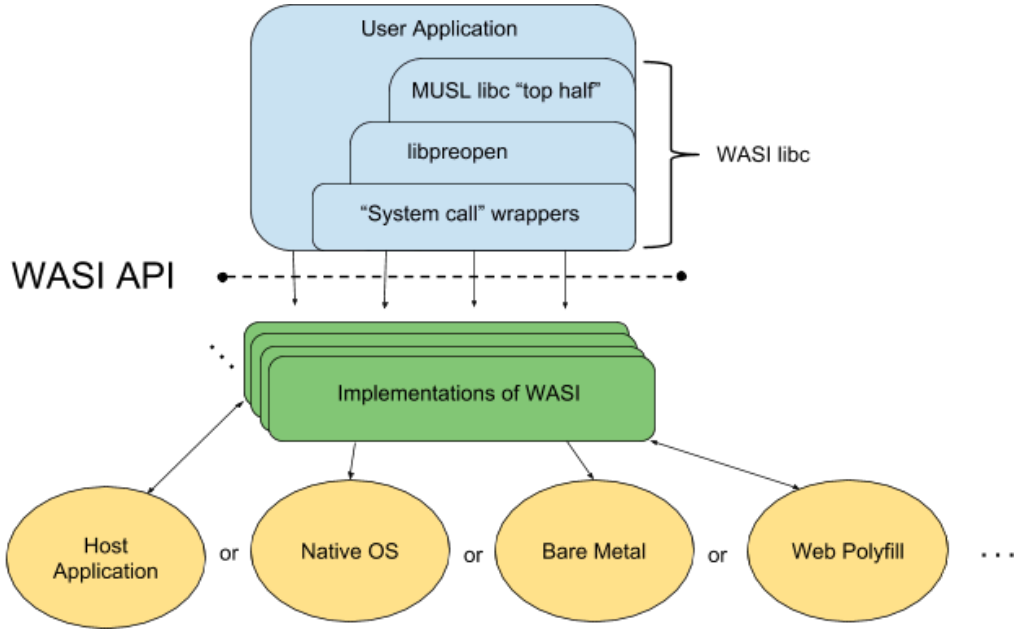
- wapm package registry for storing and serving packages
- wapm package client (called *wapm-cli*) for installing and managing packages

https://wapm.io/

WebAssembly System Interface

WASI is a modular system interface for WebAssembly. It's an API designed by the Wasmtime project that provides access to several operating-system-like features, including files and filesystems, Berkeley sockets, clocks, and random numbers

https://github.com/bytecodealliance/wasmtime/blob/master/docs/WASI-overview.md



Awesome WebAssembly Languages

This repo contains a list of languages that currently compile to or have their VMs in WebAssembly(wasm)

https://github.com/appcypher/awesome-wasm-langs

Legend

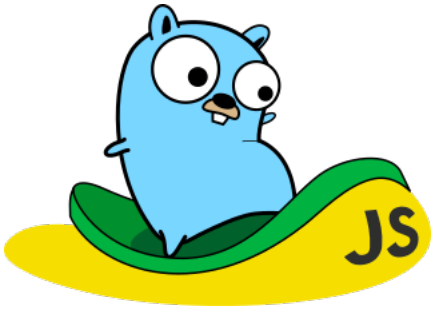
 - Work in progress.

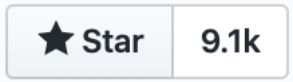
 - Unstable but usable.

 - Stable for production usage.

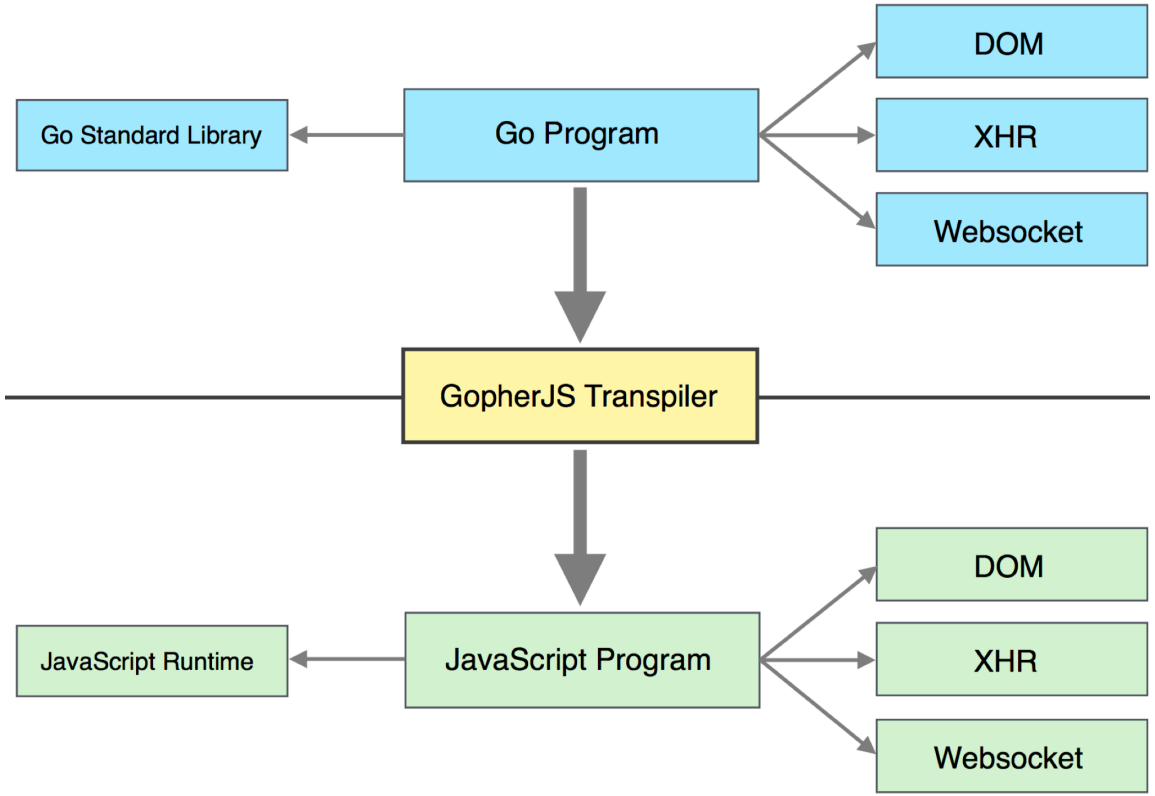
Let's

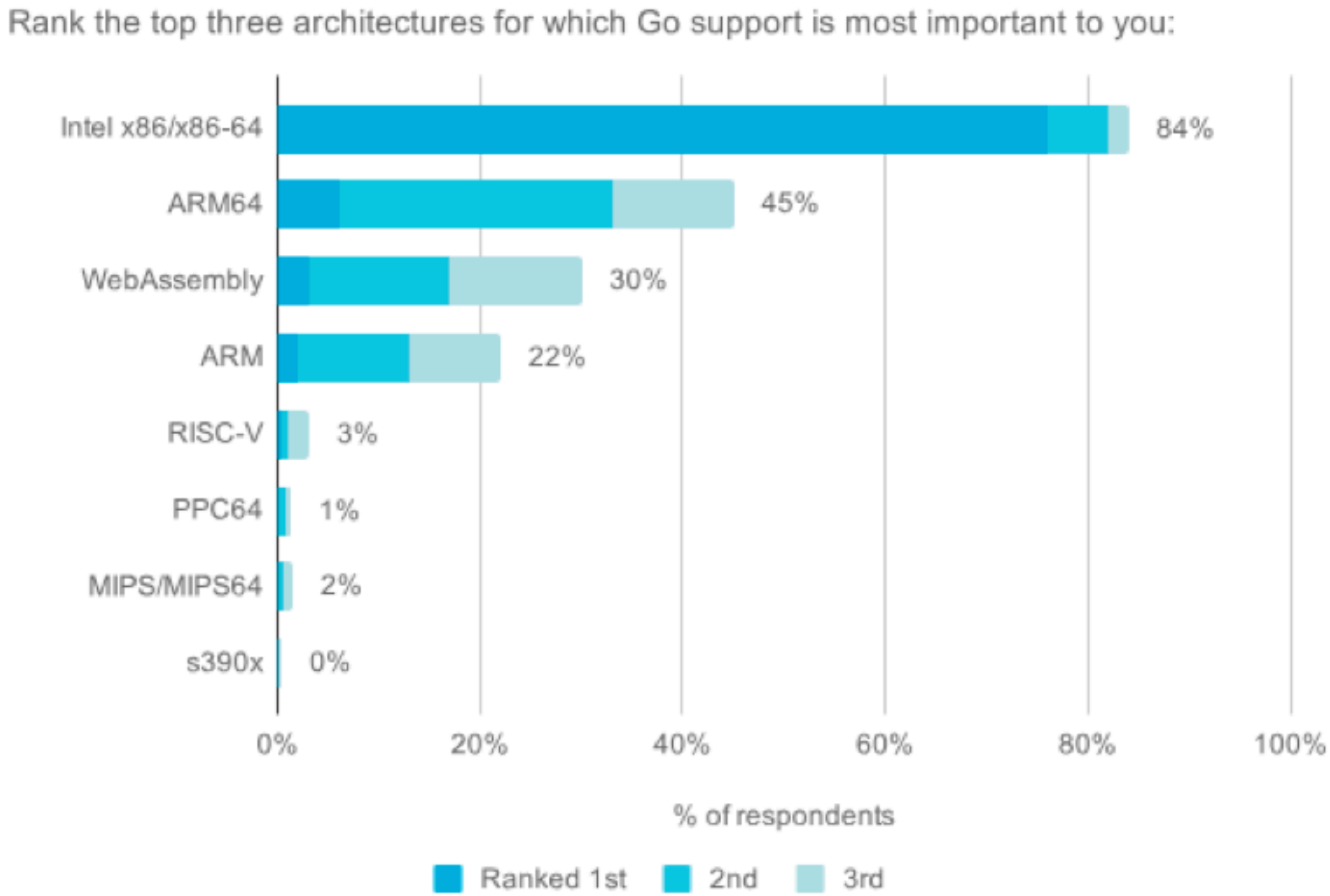
GopherJS





GopherJS compiles Go code (golang.org) to pure JavaScript code. Its main purpose is to give you the opportunity to write front-end code in Go which will still run in all browsers.




Rank the top three architectures for which Go support is most important to you:
Intel x86/x86-64
ARM64
WebAssembly
ARM
RISC-V
PPC64
MIPS/MIPS64
s390x
84%
45%
30%
22%
3%
1%
2%
0%
0%
20%
40%
60%
80%
100%
% of respondents
Ranked 1st
2nd
3rd

all: WebAssembly ("wasm") support #18892

Closed **bradfitz** opened this issue on Feb 2, 2017 · 147 comments

bradfitz commented on Feb 2, 2017 • edited ▾ Member •••

WebAssembly ("wasm") is similar to Native Client, but different notably in that other browsers plan to implement it.

http://webassembly.org/

This has been asked about a few times, so this is a tracking bug for it.

Whether we get it via cmd/compile, gccgo, or llvm-go, we can post updates here.

👍 642 🎉 146 ❤️ 221

Go 1.11 Release Notes (Aug 24, 2018)

WebAssembly

Go 1.11 adds an experimental port to WebAssembly (js/wasm).

Go programs currently compile to one WebAssembly module that includes the Go runtime for goroutine scheduling, garbage collection, maps, etc. As a result, the resulting size is at minimum around 2 MB, or 500 KB compressed. Go programs can call into JavaScript using the new experimental syscall/js package. Binary size and interop with other languages has not yet been a priority but may be addressed in future releases.

As a result of the addition of the new GOOS value "js" and GOARCH value "wasm", Go files named *_js.go or *_wasm.go will now be ignored by Go tools except when those GOOS/GOARCH values are being used. If you have existing filenames matching those patterns, you will need to rename them.

Go 1.12 Release Notes

syscall/js

The Callback type and NewCallback function have been renamed; they are now called Func and FuncOf, respectively. This is a breaking change, but WebAssembly support is still experimental and not yet subject to the Go 1 compatibility promise. Any code using the old names will need to be updated.

If a type implements the new Wrapper interface, ValueOf will use it to return the JavaScript value for that type.

The meaning of the zero Value has changed. It now represents the JavaScript undefined value instead of the number zero. This is a breaking change, but WebAssembly support is still experimental and not yet subject to the Go 1 compatibility promise. Any code relying on the zero Value to mean the number zero will need to be updated.
The new Value.Truthy method reports the JavaScript "truthiness" of a given value.

Go 1.13 Release Notes

Ports

For GOARCH=wasm, the new environment variable GOWASM takes a comma-separated list of experimental features that the binary gets compiled with. The valid values are documented here.

syscall/js

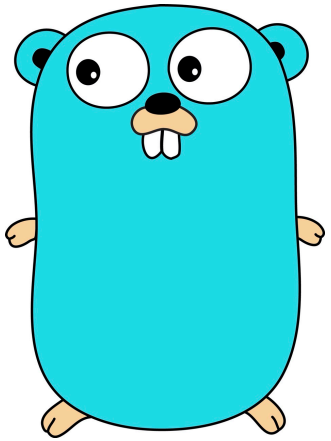
TypedArrayOf has been replaced
by CopyBytesToGo and CopyBytesToJS for copying bytes between
a byte slice and a Uint8Array.

Go 1.14 Release Notes

WebAssembly

JavaScript values referenced from Go via js.Value objects can now be garbage collected.
js.Value values can no longer be compared using the == operator, and instead must be compared using their Equal method.
js.Value now has IsUndefined, IsNull, and IsNaN methods.

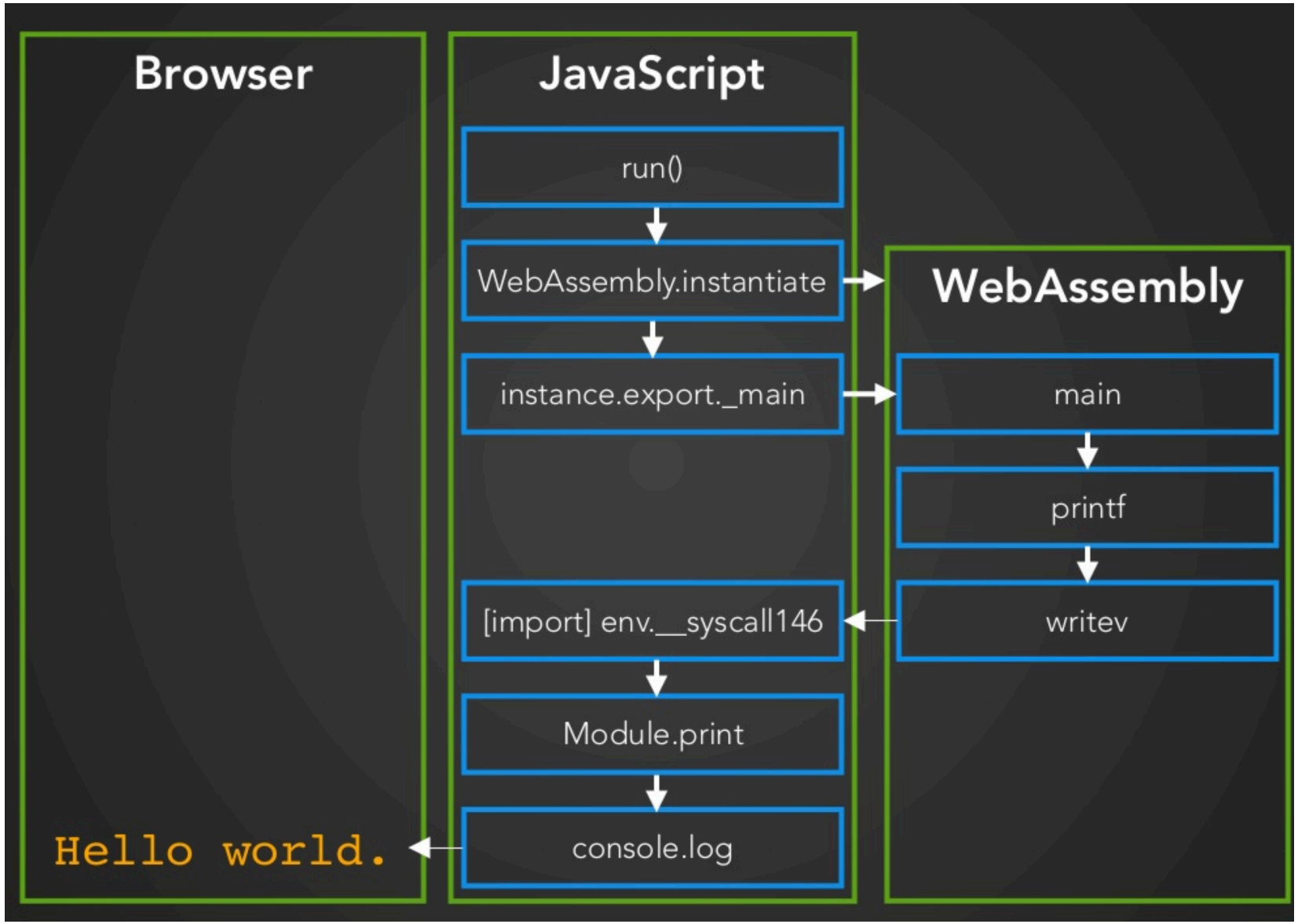
Hello GO World

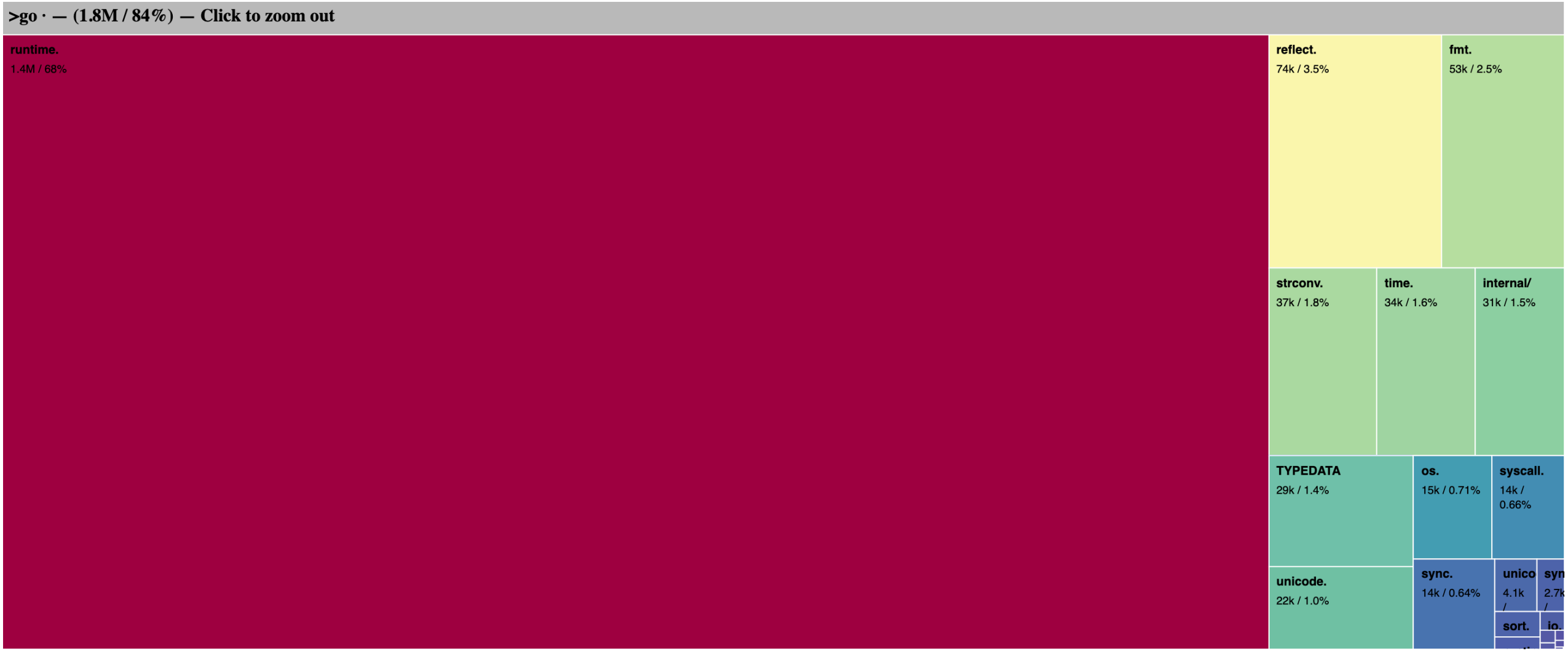


```
$ GOOS=js GOARCH=wasm go build
-o main.wasm
```

main.go

main.wasm

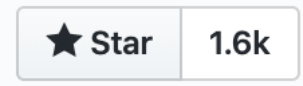

Browser
JavaScript
run()
WebAssembly.instantiate
WebAssembly
instance.export._main
main
printf
writev
[import] env.__syscall146
Module.print
console.log
Hello world.



https://github.com/knz/go-binsize-viz

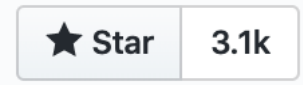
VECTY

YOUR FRONTEND, IN GO



Vecty is a library for building responsive and dynamic web frontends in Go instead of in JavaScript, HTML & CSS.

https://github.com/gopherjs/vecty



Vugu: A modern UI library for Go+WebAssembly (experimental)

https://github.com/vugu/vugu

Tiny go

Go compiler for small places. Microcontrollers, WebAssembly, and command-line tools. Based on LLVM.

https://tinygo.org/

What’s next in go?

- Next browser APIs
- Threading
- Debugging
- Garbage Collector

Used materials:

Golang WebAssembly

https://github.com/golang/go/wiki/WebAssembly

A cartoon intro to WebAssembly

https://hacks.mozilla.org/2017/02/a-cartoon-intro-to-webassembly/

Знакомство с WebAssembly @nzeemin

https://habr.com/ru/post/342180/

WebAssembly docs

https://webassembly.org/docs/

WebAssembly|MDN

https://developer.mozilla.org/en-US/docs/WebAssembly

GopherCon 2019: Johan Brandhorst - Get Going with WebAssembly

https://www.youtube.com/watch?v=Dxs4LGjmEL4

WASI docs

https://github.com/bytecodealliance/wasmtime/blob/master/docs/WASI-overview.md

詳説WebAssembly

https://pt.slideshare.net/llamerada-jp/webassembly-75175349


WA
TS

THE END

Author:
Max @ipriver

Thanks for the help
with the images:
Euhen Shumra