goto; copenhagen

GOTO Copenhagen 2019
Conference Nov. 18 - 20

WebAssembly
Beyond the Browser

Dan Callahan
@callahad

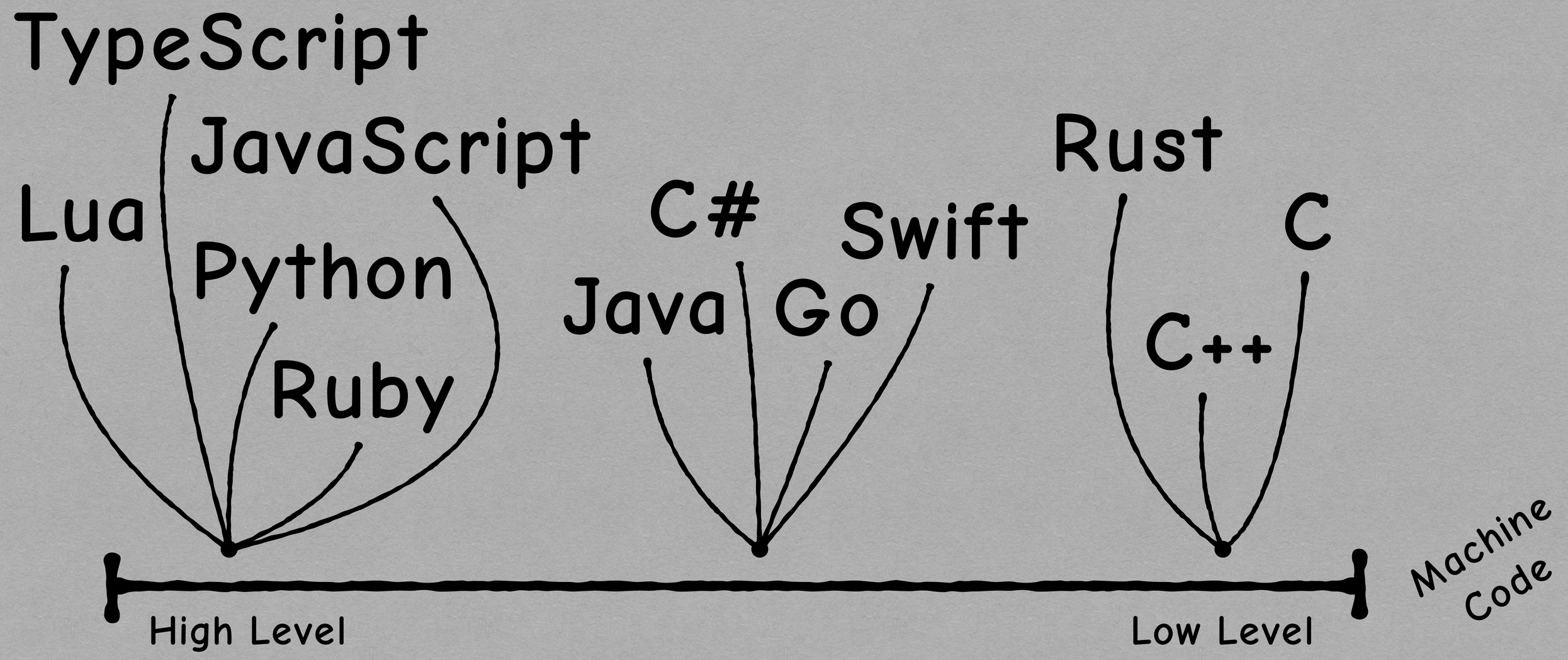
Follow us @gotocph

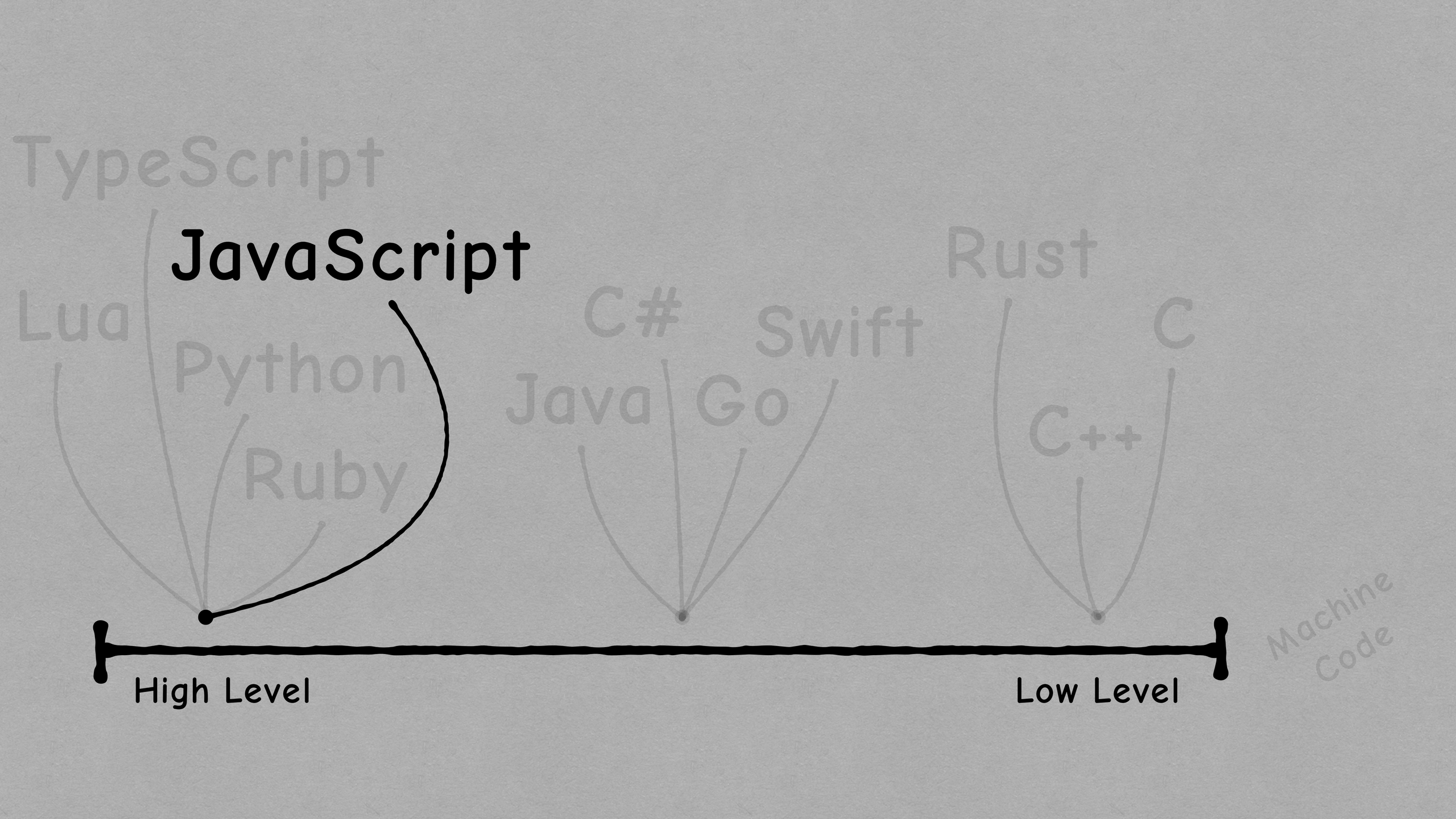
What *is* WebAssembly ...outside the browser?

HTML

HTML / CSS / JS

HTML / CSS / JS / ?


TypeScript
JavaScript
Lua
Python
Ruby
C#
Swift
Java
Go
Rust
C
C++
High Level
Low Level
Machine Code


TypeScript
JavaScript
Lua
Python
Ruby
C#
Swift
Java
Go
Rust
C
C++
High Level
Low Level
Machine Code

Sass: Syntactically Awesome St

https://sass-lang.com

Sass

Install Learn Sass Blog Documentation Get Involved

CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.

Current Releases: Dart Sass 1.23.6 LibSass 3.6.3 Ruby Sass Implementation Guide

CSS Compatible

Sass is completely compatible with all versions of CSS. We take this compatibility seriously, so that you can

Feature Rich

Sass boasts more features and abilities than any other CSS extension language out there. The Sass Core Team has

Mature

Sass has been actively supported for almost 13 years by its loving Core Team.

Sass: LibSass

https://sass-lang.com/libsass

Install Learn Sass Blog Documentation Get Involved

LibSass

We want everyone to enjoy Sass, no matter what language they use. Sass was originally written in Ruby. LibSass is a C/C++ port of the Sass engine. The point is to be simple, faster, and easy to integrate. Find out more about the project over at GitHub.

Wrappers

LibSass is just a library. To run the code locally (i.e. to compile your stylesheets), you need an implementer, or "wrapper". There are a number of other wrappers for LibSass. We encourage you to write your own wrapper — the whole point of LibSass is that we want to bring Sass to many other languages,

Wrappers

- SassC
- Crystal
- Go
- Java
- JavaScript
- Lua
- .NET
- Node
- Perl
- PHP
- Python
- Ruby
- Scala

node-sass - npm

https://www.npmjs.com/package/node-sass

Node Package Manager

npm Enterprise Products Solutions Resources Docs Support

npm

Search packages

Search

Join

Log In

node-sass

4.13.0 • Public • Published a month ago

Readme | 17 Dependencies | 7,307 Dependents | 136 Versions

node-sass

Supported Node.js versions vary by release, please consult the releases page. Bel
quick guide for minimum support:

NodeJS	Minimum node-sass version	Node Module
Node 13	4.13+	79
Node 12	4.12+	72
Node 11	4.10+	67
Node 10	4.9+	64
Node 8	4.5.3+	57

install

weekly downloads

4,275,413

version
4.13.0

license
MIT

open issues
93

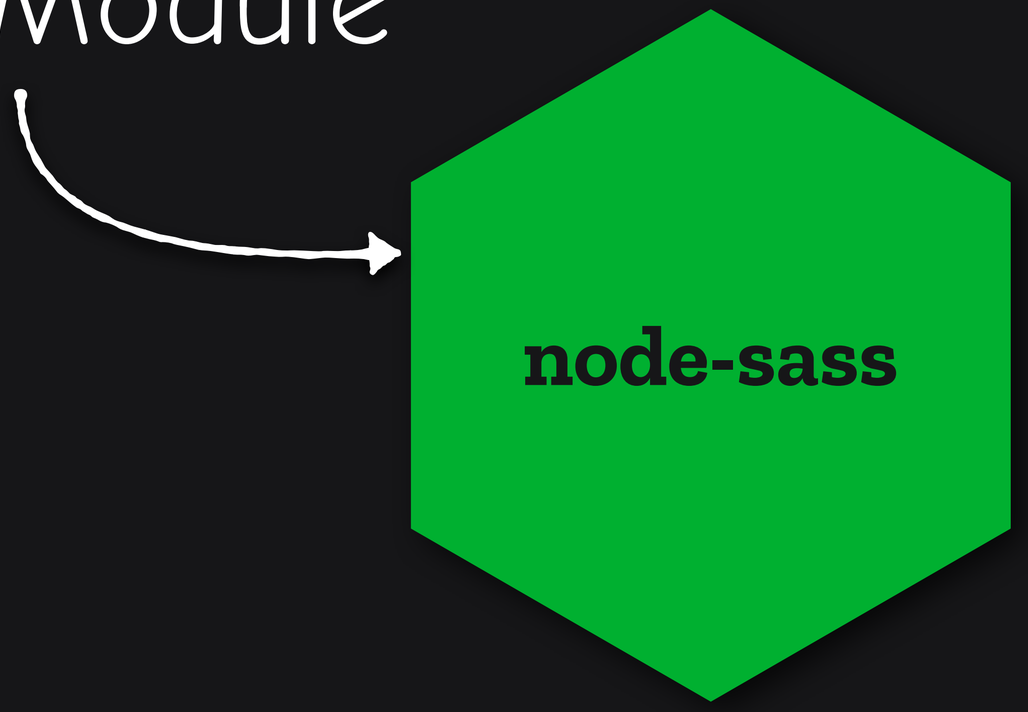
pull requests
33

homepage

repository

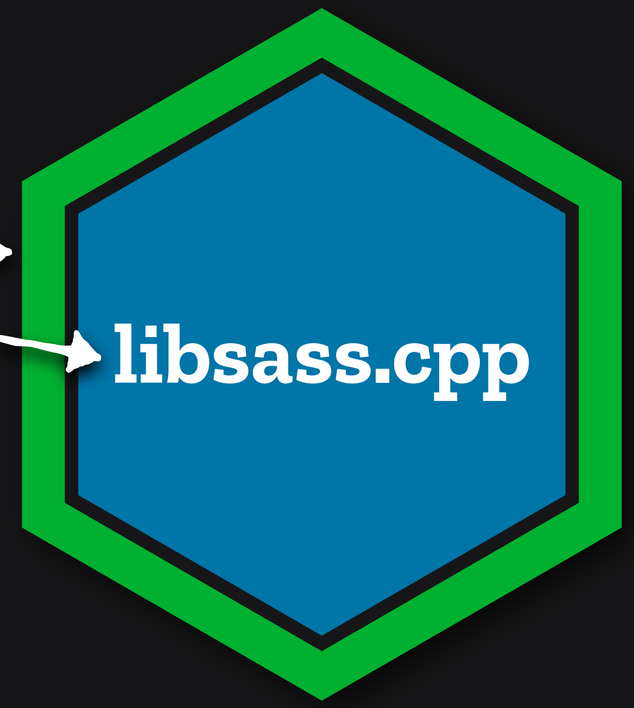
```
"dependencies" : {
  "node-sass" : "4.13.0"
}
```

Node Module

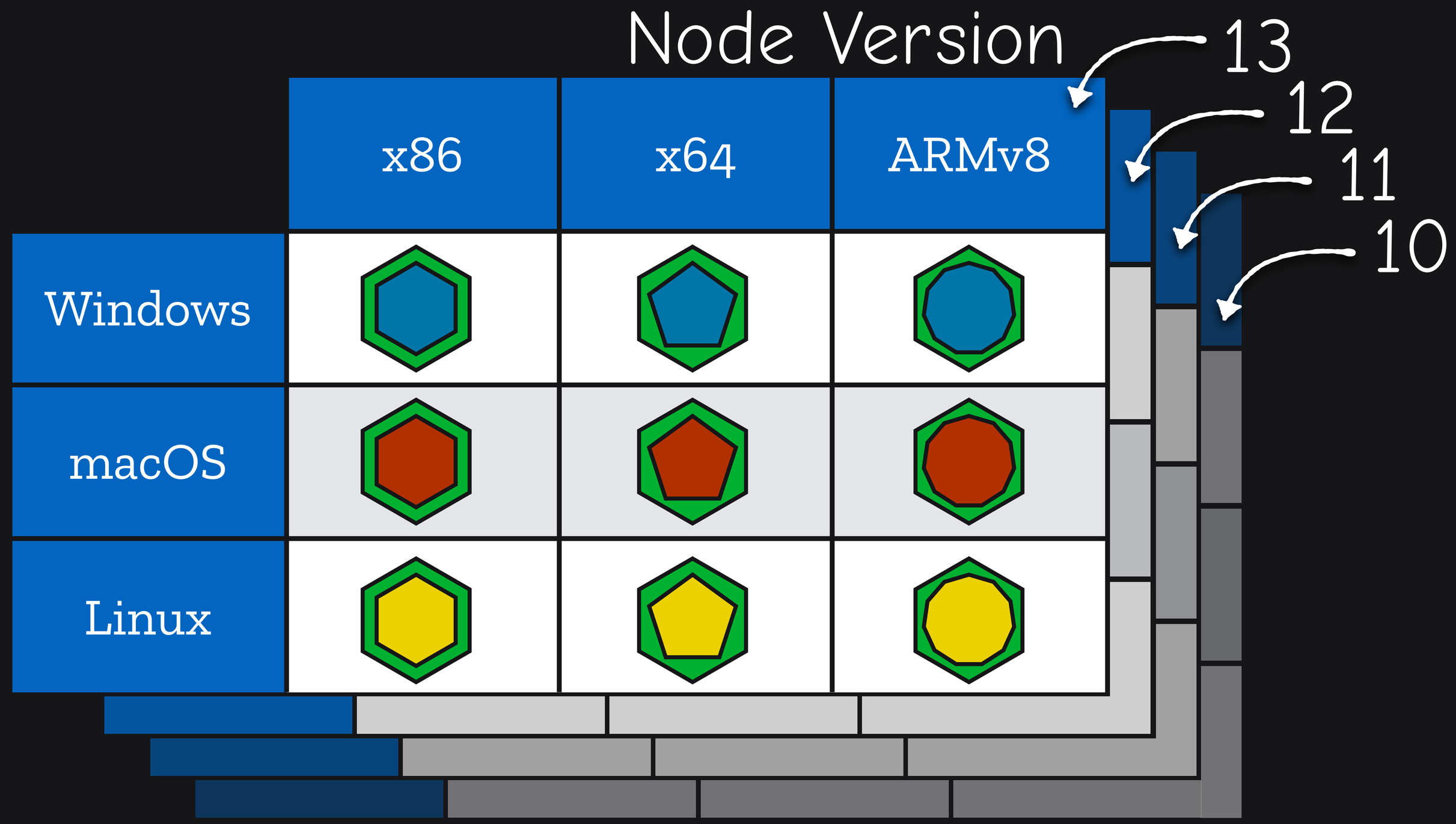


```
"dependencies" : {
  "node-sass" : "4.13.0"
}
```

Node Module



C++ Library


Node Version
13
12
11
10
x86
x64
ARMv8
Windows
macOS
Linux

Releases · sass/node-sass

https://github.com/sass/node-sass/releases

Search or jump to... / Pull requests Issues Marketplace Explore

sass / node-sass

Used by 1.9m | Watch 197 | Star 6.8k | Fork 990

Code | Issues 93 | Pull requests 33 | Actions | Projects 3 | Security | Insights

Releases | Tags

Latest release

v4.13.0

60fad5f

v4.13.0

xzyfer released this 27 days ago · 3 commits to master since this release

Features

- Node 13 support (@saper, #2766 #2767)

Community

- Fix broken link to NodeJS docs in README.md (@schwigri, #2753)
- Assorted typo fixes (@XhmikosR , #2726)
- Remove PR template (@nschonni)
- Remove sudo settings from .travis.yml (@abetomo, #2673)
- Add note in PR template about node-gyp 4.0 (@nschonni)
- Change note about Node 12 support (@nschonni)

Releases · sass/node-sass

https://github.com/sass/node-sass/releases

- Change note about Node 12 support (**@nschonni**)

Dependencies

- lodash@^4.17.15 (**@kessenich**, #2574)

Supported Environments

OS	Architecture	Node
Windows	x86 & x64	0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
OSX	x64	0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
Linux*	x86 & x64	0.10, 0.12, 1, 2, 3, 4, 5, 6, 7, 8**, 9**, 10**^, 11**^, 12**^, 13**^
Alpine Linux	x64	6, 8, 10, 11, 12, 13
FreeBSD	i386 amd64	6, 8, 10, 12, 13

Assets 119

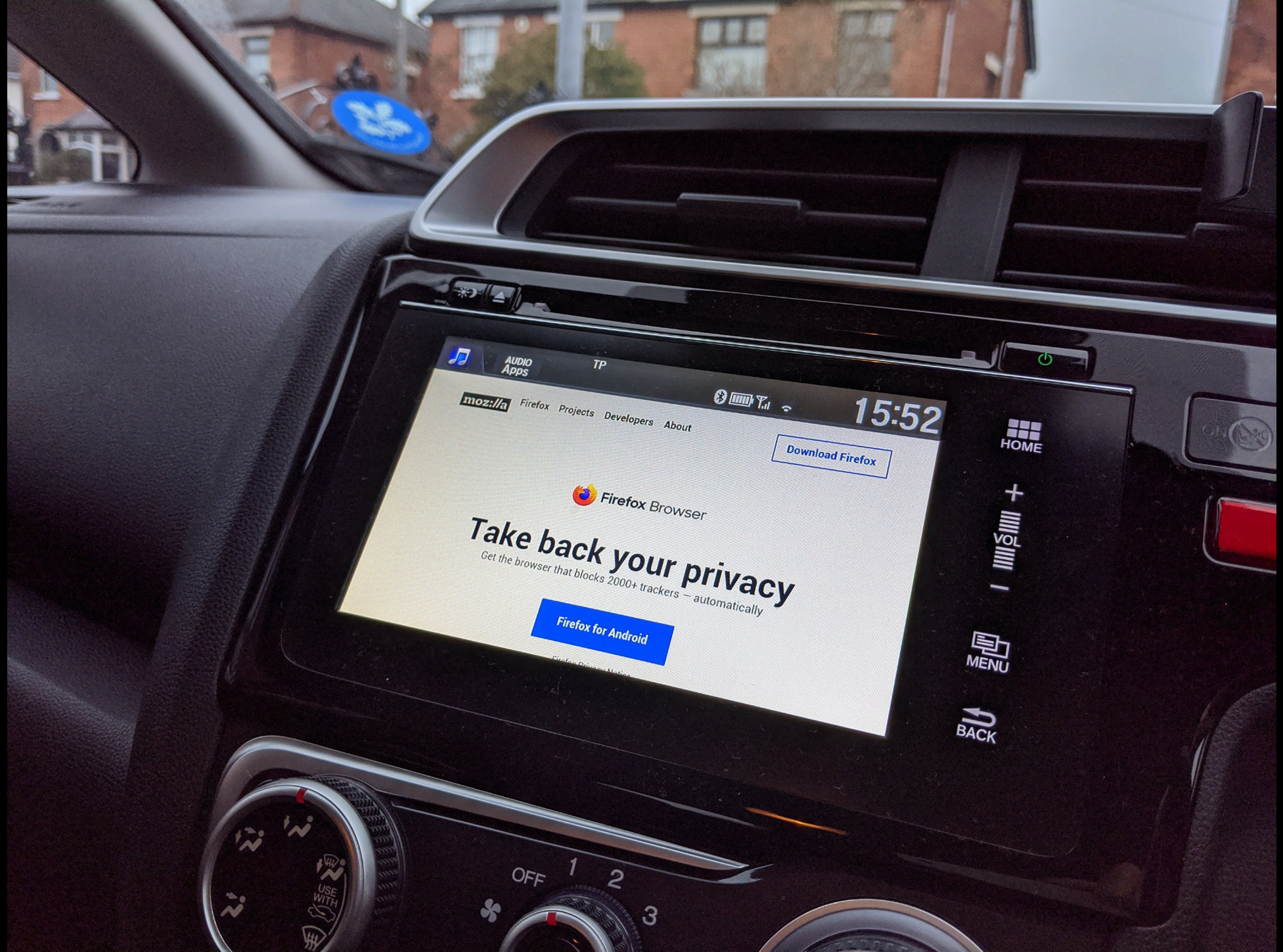
darwin-x64-11_binding.node	2.7 MB
darwin-x64-14_binding.node	2.71 MB
darwin-x64-42_binding.node	2.71 MB

What about the Web?

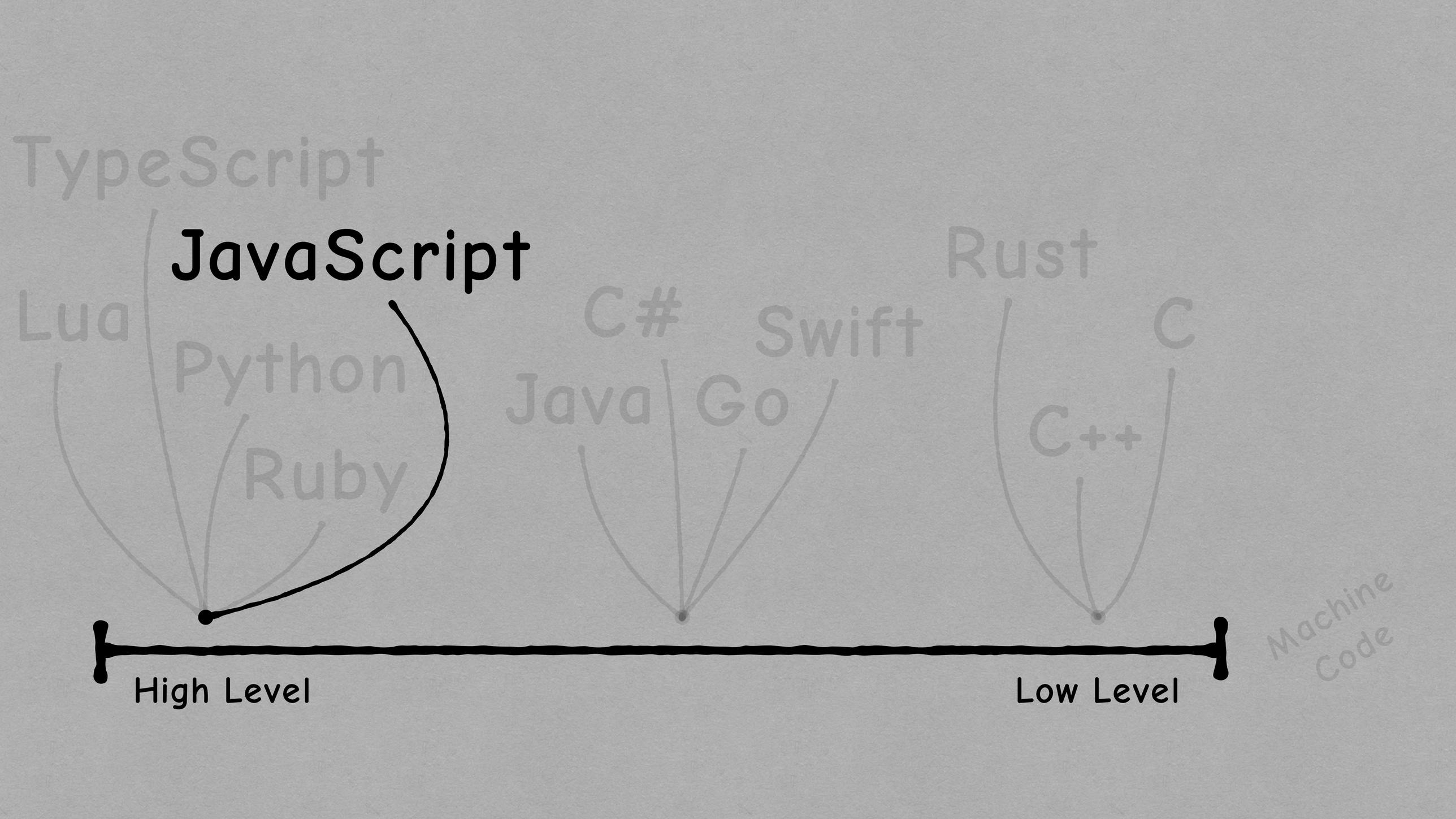
Assets 119

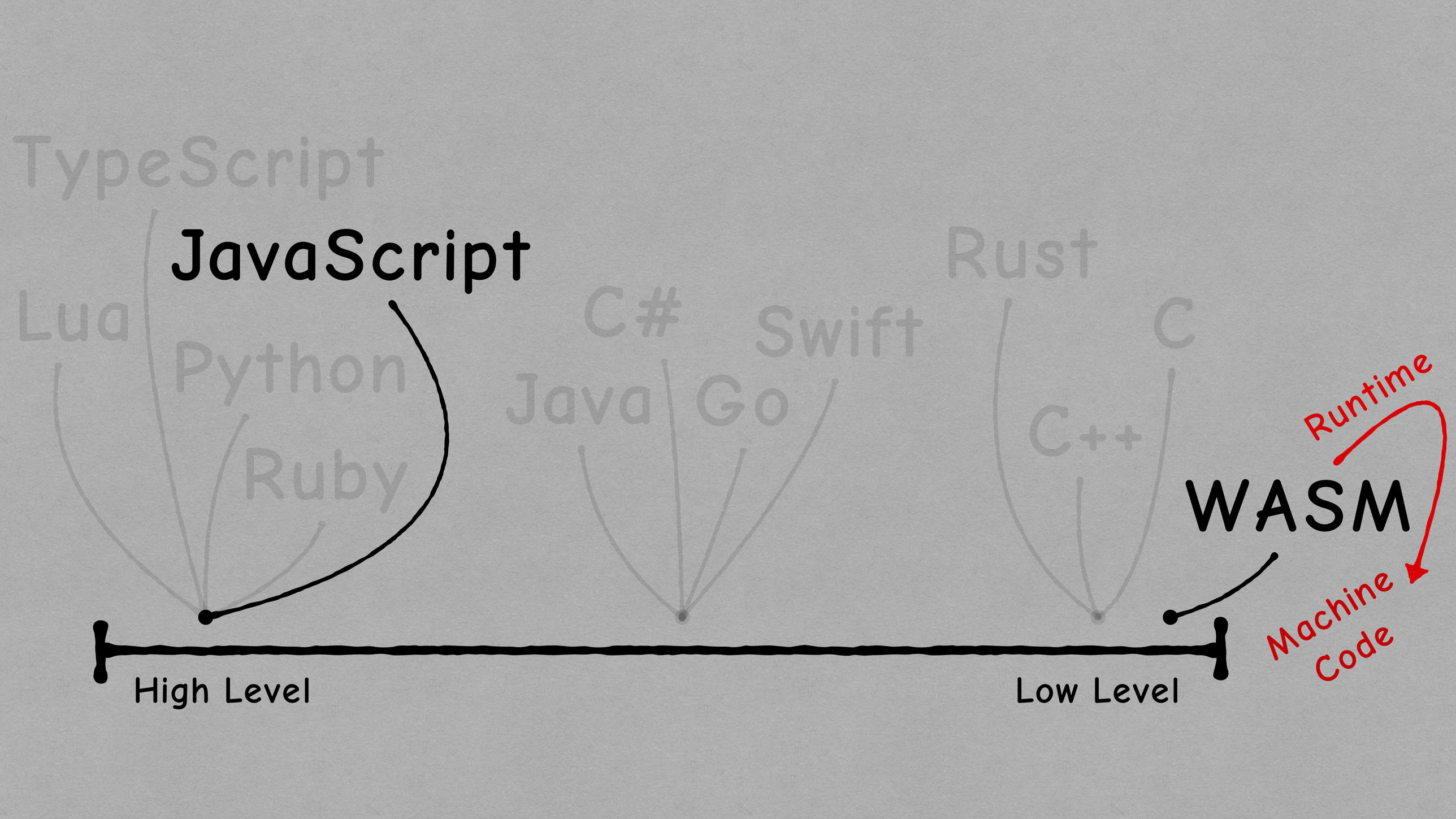
darwin-x64-11_binding.node	2.7 MB
darwin-x64-14_binding.node	2.71 MB
darwin-x64-42_binding.node	2.71 MB
darwin-x64-43_binding.node	2.71 MB
darwin-x64-44_binding.node	2.71 MB
darwin-x64-45_binding.node	2.71 MB
darwin-x64-46_binding.node	2.71 MB
darwin-x64-47_binding.node	2.71 MB
darwin-x64-48_binding.node	2.71 MB
darwin-x64-51_binding.node	3.12 MB
darwin-x64-57_binding.node	3.12 MB
darwin-x64-59_binding.node	3.12 MB
darwin-x64-64_binding.node	3.12 MB
darwin-x64-67_binding.node	
darwin-x64-72_binding.node	
darwin-x64-79_binding.node	
freebsd-ia32-48_binding.node	
freebsd-ia32-57_binding.node	
freebsd-ia32-64_binding.node	
freebsd-ia32-72_binding.node	2.5 MB
freebsd-ia32-79_binding.node	2.41 MB
freebsd-x64-48_binding.node	2.58 MB
freebsd-x64-57_binding.node	2.68 MB
freebsd-x64-64_binding.node	2.68 MB
freebsd-x64-72_binding.node	2.68 MB
freebsd-x64-79_binding.node	2.59 MB
linux-ia32-11_binding.node	3.45 MB
linux-ia32-14_binding.node	3.48 MB
linux-ia32-42_binding.node	3.48 MB
linux-ia32-43_binding.node	3.48 MB
linux-ia32-46_binding.node	3.5 MB
linux-ia32-47_binding.node	3.51 MB
linux-ia32-48_binding.node	3.51 MB
linux-ia32-51_binding.node	3.51 MB
linux-ia32-57_binding.node	3.49 MB
linux-ia32-59_binding.node	3.49 MB
linux-x64-11_binding.node	3.4 MB
linux-x64-14_binding.node	3.39 MB
linux-x64-42_binding.node	3.38 MB
linux-x64-43_binding.node	3.38 MB
linux-x64-45_binding.node	3.4 MB
linux-x64-46_binding.node	3.4 MB
linux-x64-47_binding.node	3.4 MB
linux-x64-48_binding.node	3.4 MB
linux-x64-51_binding.node	3.4 MB
linux-x64-57_binding.node	3.38 MB
linux-x64-59_binding.node	3.38 MB
linux-x64-64_binding.node	3.38 MB
linux-x64-67_binding.node	3.38 MB
linux_musl-x64-72_binding.node	3.35 MB
linux_musl-x64-79_binding.node	3.35 MB
win32-ia32-11_binding.node	1.89 MB
win32-ia32-11_binding.pdb	12.1 MB
win32-ia32-14_binding.node	1.89 MB
win32-ia32-14_binding.pdb	12.1 MB
win32-ia32-43_binding.node	1.86 MB
win32-ia32-43_binding.pdb	11.9 MB
win32-ia32-44_binding.node	1.86 MB
win32-ia32-44_binding.pdb	11.9 MB
win32-ia32-45_binding.node	1.87 MB
win32-ia32-45_binding.pdb	11.9 MB
win32-ia32-46_binding.node	1.87 MB
win32-ia32-46_binding.pdb	11.9 MB
win32-ia32-47_binding.node	1.87 MB
win32-ia32-48_binding.node	1.8 MB
win32-ia32-48_binding.pdb	18.2 MB
win32-ia32-51_binding.node	1.8 MB
win32-ia32-51_binding.pdb	18.2 MB
win32-ia32-57_binding.node	1.8 MB
win32-ia32-57_binding.pdb	18.2 MB
win32-ia32-59_binding.node	1.75 MB
win32-ia32-59_binding.pdb	17.9 MB
win32-ia32-64_binding.node	1.83 MB
win32-ia32-64_binding.pdb	18.5 MB
win32-ia32-67_binding.node	1.83 MB
win32-ia32-67_binding.pdb	18.5 MB
win32-ia32-72_binding.node	2.02 MB
win32-ia32-72_binding.pdb	22.6 MB
win32-ia32-79_binding.node	2.02 MB
win32-x64-43_binding.pdb	14 MB
win32-x64-44_binding.node	2.46 MB
win32-x64-44_binding.pdb	14 MB
win32-x64-45_binding.node	2.47 MB
win32-x64-45_binding.pdb	14 MB
win32-x64-46_binding.node	2.47 MB
win32-x64-46_binding.pdb	14 MB
win32-x64-47_binding.node	2.47 MB
win32-x64-47_binding.pdb	14 MB
win32-x64-48_binding.node	2.44 MB
win32-x64-48_binding.pdb	19 MB
win32-x64-51_binding.node	2.44 MB
win32-x64-51_binding.pdb	19 MB
win32-x64-57_binding.node	2.44 MB
win32-x64-57_binding.pdb	19 MB
win32-x64-59_binding.node	2.36 MB
win32-x64-59_binding.pdb	18.7 MB
win32-x64-64_binding.node	2.29 MB
win32-x64-64_binding.pdb	19.1 MB
win32-x64-67_binding.node	2.29 MB
win32-x64-67_binding.pdb	19.1 MB
win32-x64-72_binding.node	2.76 MB
win32-x64-72_binding.pdb	24 MB
win32-x64-79_binding.node	2.76 MB
win32-x64-79_binding.pdb	24 MB
Source code (zip)	
Source code (tar.gz)	

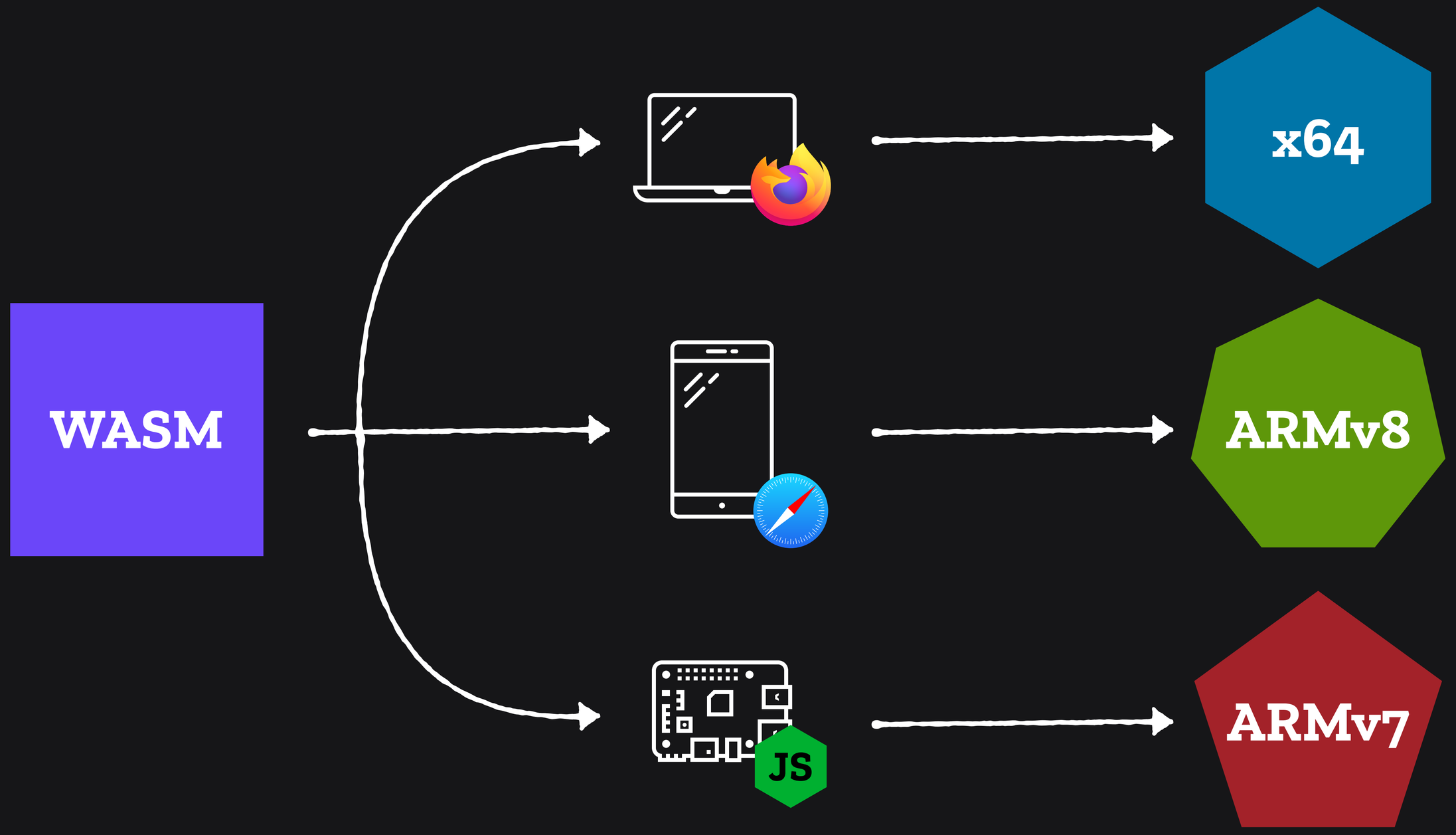
engadget.com/2016/09/02/lg-s-29-inch-windows-10-tablet-comes-stuck-to-a-smart-fridge

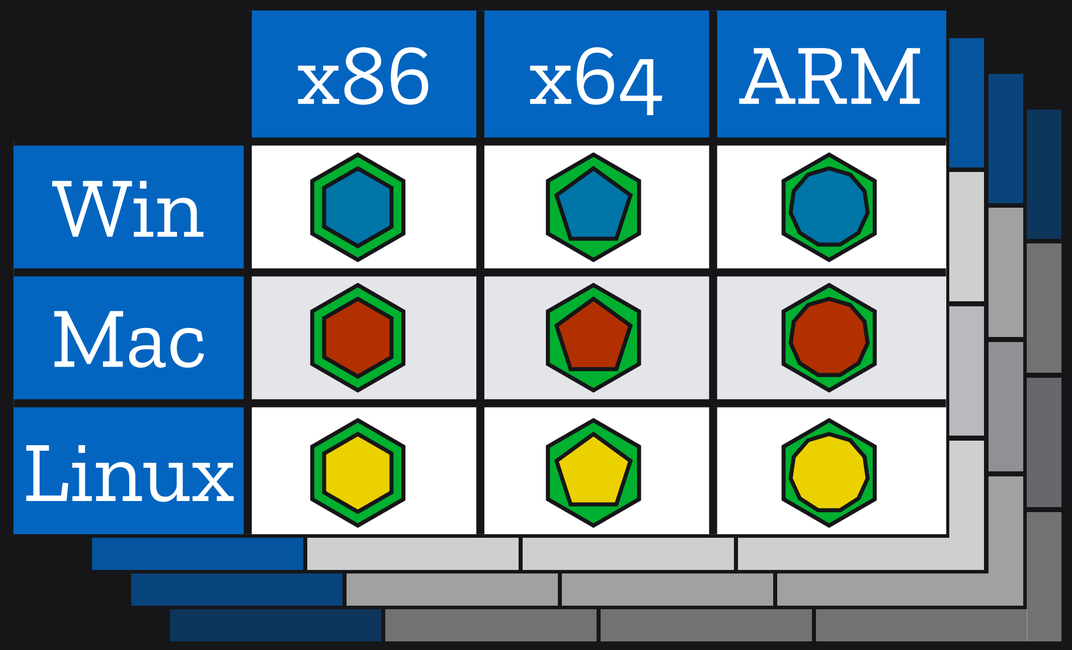

AUDIO
Apps
TP
15:52
mozilla
Firefox
Projects
Developers
About
Download Firefox
Firefox Browser
Take back your privacy
Get the browser that blocks 2000+ trackers — automatically
Firefox for Android
HOME
VOL
MENU
BACK
OFF
1
2
3
USE WITH

macrumors.com/how-to/browse-the-internet-watchos-5


TypeScript
JavaScript
Lua
Python
Ruby
C#
Swift
Java
Go
Rust
C
C++
Machine Code
High Level
Low Level


TypeScript
JavaScript
Lua
Python
Ruby
C#
Swift
Java
Go
Rust
C
C++
Runtime
WASM
Machine Code
High Level
Low Level




x86
x64
ARM
Win
Mac
Linux


x86
x64
ARM
Win
Mac
Linux

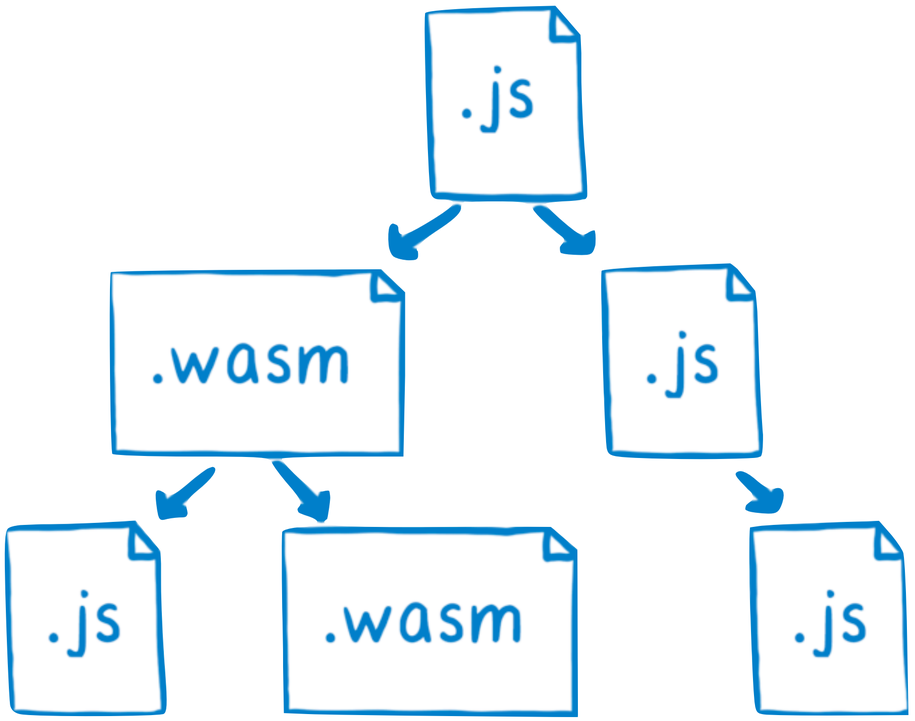

libsass
.wasm

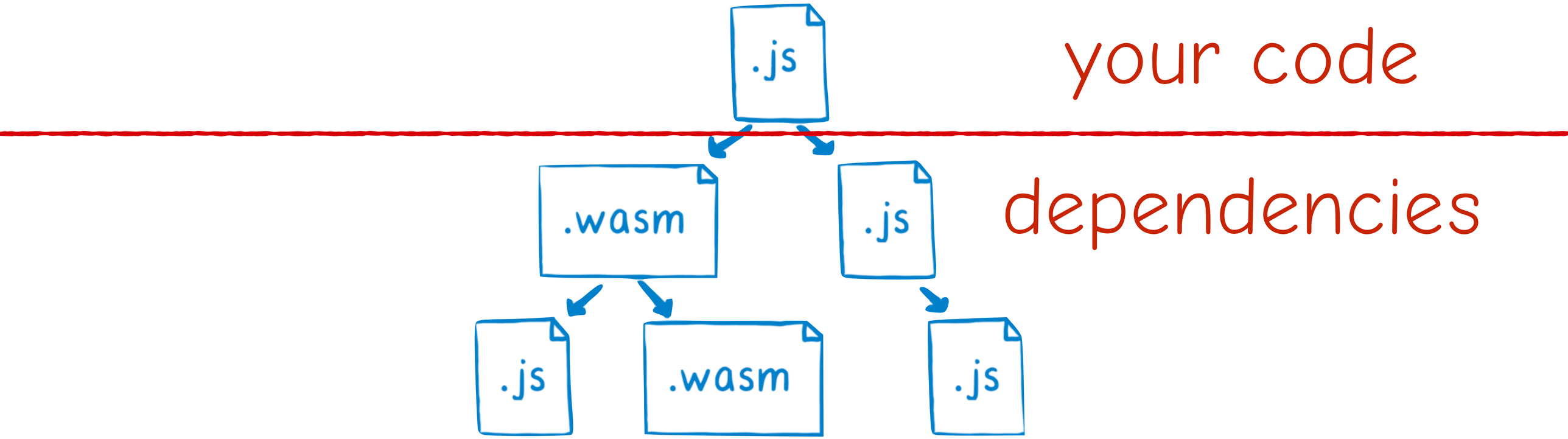

x86
x64
ARM
Win
Mac
Linux


libsass
.wasm


```
import { render } from "./libsass.wasm";
```

...soon?


.js
.wasm
.js
.js
.wasm
.js


.js
your code
.wasm
.js
dependencies
.js
.wasm
.js

Can I use WebAssembly ?

WebAssembly - OTHER

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			49						
			60			10.2			
	[3] 15	55	61	10.1	47	10.3		4.4	
11	16	56	62	11	48	11	all	56	61
		57	63	TP	49				
		58	64		50				
		59	65						

Notes | Known issues (0) | Resources (7) | Feedback

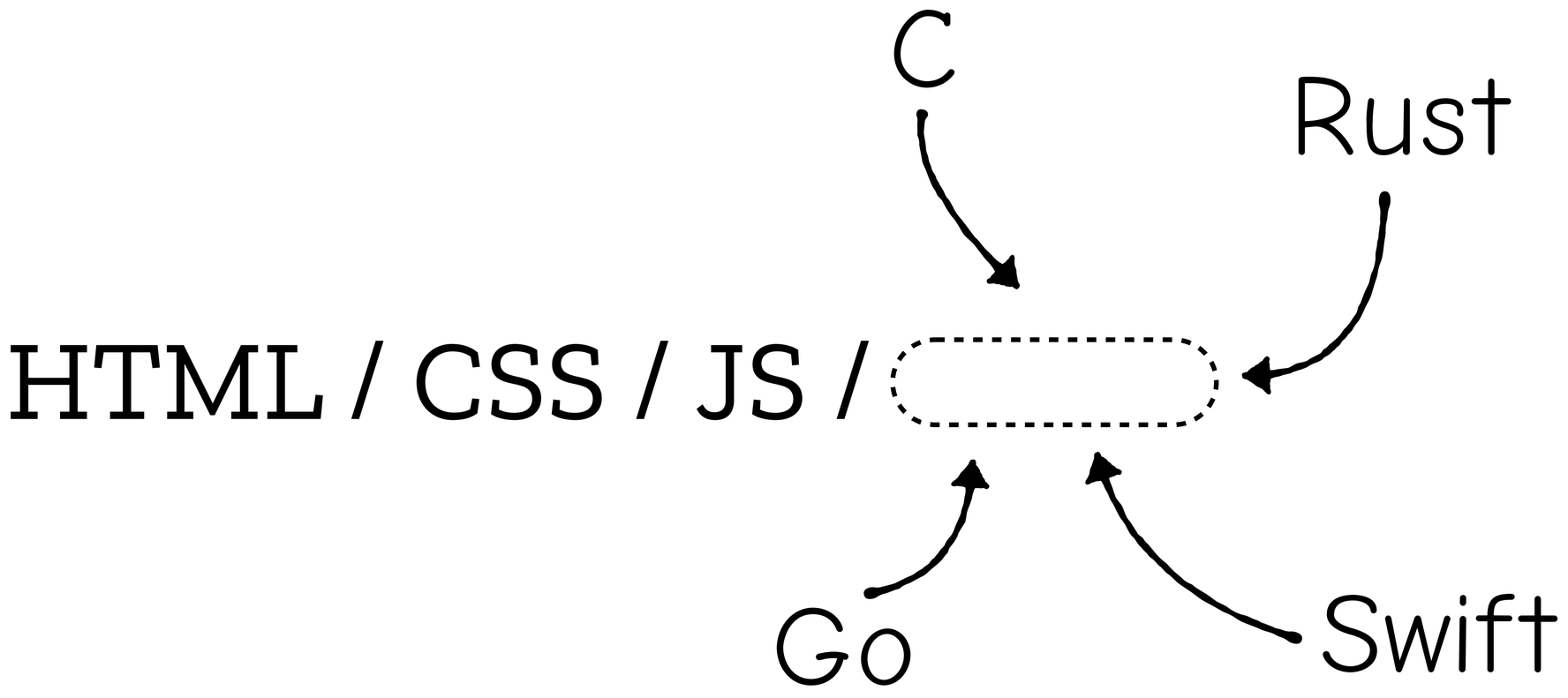
MS Edge status: Preview Release

[3] Can be enabled via the Experimental JavaScript Features flag

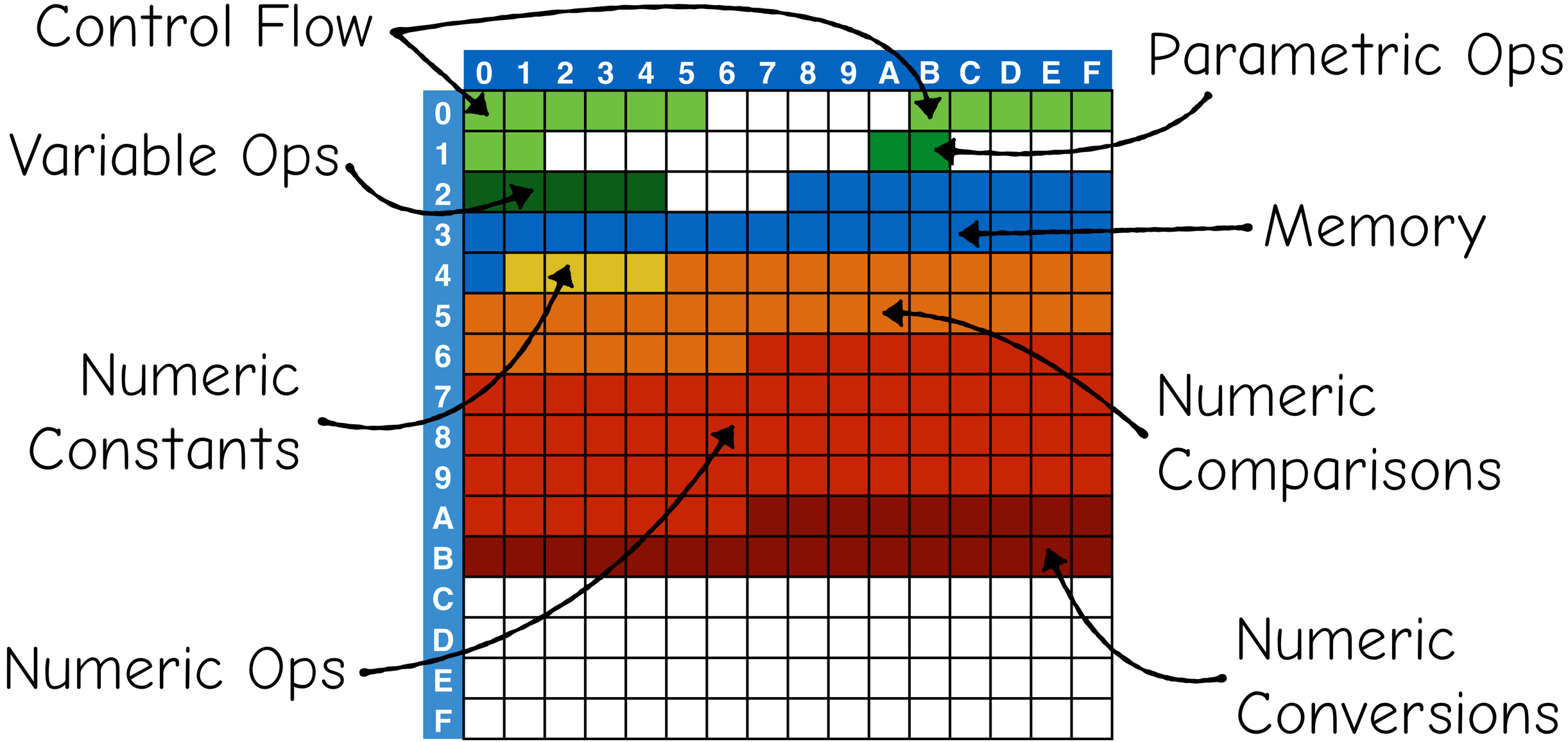
October 2017

HTML / CSS / JS / WASM

HTML / CSS / JS /


C
Rust
HTML / CSS / JS /
Go
Swift

WebAssembly Instructions



Control Instructions

Control instructions have varying encodings. For structured instructions, the instruction sequences forming nested blocks are terminated with explicit opcodes for end and else.

$$
\begin{array}{lcll}
\texttt{instr} & ::= & \texttt{0x00} & \Rightarrow \ \text{unreachable} \\
& | & \texttt{0x01} & \Rightarrow \ \text{nop} \\
& | & \texttt{0x02}\ rt{:}\texttt{blocktype}\ (in{:}\texttt{instr})^*\ \texttt{0x0B} & \Rightarrow \ \text{block}\ rt\ in^*\ \text{end} \\
& | & \texttt{0x03}\ rt{:}\texttt{blocktype}\ (in{:}\texttt{instr})^*\ \texttt{0x0B} & \Rightarrow \ \text{loop}\ rt\ in^*\ \text{end} \\
& | & \texttt{0x04}\ rt{:}\texttt{blocktype}\ (in{:}\texttt{instr})^*\ \texttt{0x0B} & \Rightarrow \ \text{if}\ rt\ in^*\ \text{else}\ \epsilon\ \text{end} \\
& | & \texttt{0x04}\ rt{:}\texttt{blocktype}\ (in_1{:}\texttt{instr})^*\ \texttt{0x05}\ (in_2{:}\texttt{instr})^*\ \texttt{0x0B} & \Rightarrow \ \text{if}\ rt\ in_1^*\ \text{else}\ in_2^*\ \text{end} \\
& | & \texttt{0x0C}\ l{:}\texttt{labelidx} & \Rightarrow \ \text{br}\ l \\
& | & \texttt{0x0D}\ l{:}\texttt{labelidx} & \Rightarrow \ \text{br\_if}\ l \\
& | & \texttt{0x0E}\ l^*{:}\texttt{vec}(\texttt{labelidx})\ l_N{:}\texttt{labelidx} & \Rightarrow \ \text{br\_table}\ l^*\ l_N \\
& | & \texttt{0x0F} & \Rightarrow \ \text{return} \\
& | & \texttt{0x10}\ x{:}\texttt{funcidx} & \Rightarrow \ \text{call}\ x \\
& | & \texttt{0x11}\ x{:}\texttt{typeidx}\ \texttt{0x00} & \Rightarrow \ \text{call\_indirect}\ x
\end{array}
$$

Expressions

Expressions are encoded by their instruction sequence terminated with an explicit 0x0B opcode for end.

$$\texttt{expr} \quad ::= \quad (in{:}\texttt{instr})^*\ \texttt{0x0B} \quad \Rightarrow \quad in^*\ \text{end}$$

Parametric Instructions

Parametric instructions are represented by single byte codes.

$$
\begin{array}{lcll}
\texttt{instr} & ::= & \dots & \\
& | & \texttt{0x1A} & \Rightarrow \quad \text{drop} \\
& | & \texttt{0x1B} & \Rightarrow \quad \text{select}
\end{array}
$$

Variable Instructions

Variable instructions are represented by byte codes followed by the encoding of the respective index.

$$
\begin{array}{lcllll}
\texttt{instr} & ::= & \dots & & & \\
& | & \texttt{0x20} & x{:}\texttt{localidx} & \Rightarrow & \text{local. get } x \\
& | & \texttt{0x21} & x{:}\texttt{localidx} & \Rightarrow & \text{local. set } x \\
& | & \texttt{0x22} & x{:}\texttt{localidx} & \Rightarrow & \text{local. tee } x \\
& | & \texttt{0x23} & x{:}\texttt{globalidx} & \Rightarrow & \text{global. get } x \\
& | & \texttt{0x24} & x{:}\texttt{globalidx} & \Rightarrow & \text{global. set } x
\end{array}
$$

Memory Instructions

Each variant of memory instruction is encoded with a different byte code. Loads and stores are followed by the encoding of their *memarg* immediate.

memarg	::=	a:u32 o:u32	⇒	{align a, offset o}							
instr	::=	…									
	|	0x28 m:memarg	⇒	i32. load m	|	0x36 m:memarg	⇒	i32. store m			
	|	0x29 m:memarg	⇒	i64. load m	|	0x37 m:memarg	⇒	i64. store m			
	|	0x2A m:memarg	⇒	f32. load m	|	0x38 m:memarg	⇒	f32. store m			
	|	0x2B m:memarg	⇒	f64. load m	|	0x39 m:memarg	⇒	f64. store m			
	|	0x2C m:memarg	⇒	i32. load8_s m	|	0x3A m:memarg	⇒	i32. store8 m			
	|	0x2D m:memarg	⇒	i32. load8_u m	|	0x3B m:memarg	⇒	i32. store16 m			
	|	0x2E m:memarg	⇒	i32. load16_s m	|	0x3C m:memarg	⇒	i64. store8 m			
	|	0x2F m:memarg	⇒	i32. load16_u m	|	0x3D m:memarg	⇒	i64. store16 m			
	|	0x30 m:memarg	⇒	i64. load8_s m	|	0x3E m:memarg	⇒	i64. store32 m			
	|	0x31 m:memarg	⇒	i64. load8_u m	|	0x3F 0x00	⇒	memory. size			
	|	0x32 m:memarg	⇒	i64. load16_s m	|	0x40 0x00	⇒	memory. grow			
	|	0x33 m:memarg	⇒	i64. load16_u m							
	|	0x34 m:memarg	⇒	i64. load32_s m							
	|	0x35 m:memarg	⇒	i64. load32_u m							

Numeric Instructions

All variants of numeric instructions are represented by separate byte codes.

The const instructions are followed by the respective literal.

instr	::=	…		
	|	0x41 n:i32	⇒	i32. const n
	|	0x42 n:i64	⇒	i64. const n
	|	0x43 z:f32	⇒	f32. const z
	|	0x44 z:f64	⇒	f64. const z

All other numeric instructions are plain opcodes without any immediates.

instr	::=	…															
	|	0x45	⇒	i32. eqz	|	0x50	⇒	i64. eqz	|	0x5B	⇒	f32. eq	|	0x61	⇒	f64. eq
	|	0x46	⇒	i32. eq	|	0x51	⇒	i64. eq	|	0x5C	⇒	f32. ne	|	0x62	⇒	f64. ne
	|	0x47	⇒	i32. ne	|	0x52	⇒	i64. ne	|	0x5D	⇒	f32. lt	|	0x63	⇒	f64. lt
	|	0x48	⇒	i32. lt_s	|	0x53	⇒	i64. lt_s	|	0x5E	⇒	f32. gt	|	0x64	⇒	f64. gt
	|	0x49	⇒	i32. lt_u	|	0x54	⇒	i64. lt_u	|	0x5F	⇒	f32. le	|	0x65	⇒	f64. le
	|	0x4A	⇒	i32. gt_s	|	0x55	⇒	i64. gt_s	|	0x60	⇒	f32. ge	|	0x66	⇒	f64. ge
	|	0x4B	⇒	i32. gt_u	|	0x56	⇒	i64. gt_u								
	|	0x4C	⇒	i32. le_s	|	0x57	⇒	i64. le_s								
	|	0x4D	⇒	i32. le_u	|	0x58	⇒	i64. le_u								
	|	0x4E	⇒	i32. ge_s	|	0x59	⇒	i64. ge_s								
	|	0x4F	⇒	i32. ge_u	|	0x5A	⇒	i64. ge_u								

0x67	⇒	i32.clz	0x79	⇒	i64.clz	0x8B	⇒	f32.abs	0x99	⇒	f64.abs
0x68	⇒	i32.ctz	0x7A	⇒	i64.ctz	0x8C	⇒	f32.neg	0x9A	⇒	f64.neg
0x69	⇒	i32.popcnt	0x7B	⇒	i64.popcnt	0x8D	⇒	f32.ceil	0x9B	⇒	f64.ceil
0x6A	⇒	i32.add	0x7C	⇒	i64.add	0x8E	⇒	f32.floor	0x9C	⇒	f64.floor
0x6B	⇒	i32.sub	0x7D	⇒	i64.sub	0x8F	⇒	f32.trunc	0x9D	⇒	f64.trunc
0x6C	⇒	i32.mul	0x7E	⇒	i64.mul	0x90	⇒	f32.nearest	0x9E	⇒	f64.nearest
0x6D	⇒	i32.div_s	0x7F	⇒	i64.div_s	0x91	⇒	f32.sqrt	0x9F	⇒	f64.sqrt
0x6E	⇒	i32.div_u	0x80	⇒	i64.div_u	0x92	⇒	f32.add	0xA0	⇒	f64.add
0x6F	⇒	i32.rem_s	0x81	⇒	i64.rem_s	0x93	⇒	f32.sub	0xA1	⇒	f64.sub
0x70	⇒	i32.rem_u	0x82	⇒	i64.rem_u	0x94	⇒	f32.mul	0xA2	⇒	f64.mul
0x71	⇒	i32.and	0x83	⇒	i64.and	0x95	⇒	f32.div	0xA3	⇒	f64.div
0x72	⇒	i32.or	0x84	⇒	i64.or	0x96	⇒	f32.min	0xA4	⇒	f64.min
0x73	⇒	i32.xor	0x85	⇒	i64.xor	0x97	⇒	f32.max	0xA5	⇒	f64.max
0x74	⇒	i32.shl	0x86	⇒	i64.shl	0x98	⇒	f32.copysign	0xA6	⇒	f64.copysign
0x75	⇒	i32.shr_s	0x87	⇒	i64.shr_s						
0x76	⇒	i32.shr_u	0x88	⇒	i64.shr_u						
0x77	⇒	i32.rotl	0x89	⇒	i64.rotl						
0x78	⇒	i32.rotr	0x8A	⇒	i64.rotr						

0xA7	⇒	i32. wrap_i64
0xA8	⇒	i32. trunc_f32_s
0xA9	⇒	i32. trunc_f32_u
0xAA	⇒	i32. trunc_f64_s
0xAB	⇒	i32. trunc_f64_u
0xAC	⇒	i64. extend_i32_s
0xAD	⇒	i64. extend_i32_u
0xAE	⇒	i64. trunc_f32_s
0xAF	⇒	i64. trunc_f32_u
0xB0	⇒	i64. trunc_f64_s
0xB1	⇒	i64. trunc_f64_u
0xB2	⇒	f32. convert_i32_s
0xB3	⇒	f32. convert_i32_u
0xB4	⇒	f32. convert_i64_s
0xB5	⇒	f32. convert_i64_u
0xB6	⇒	f32. demote_f64
0xB7	⇒	f64. convert_i32_s
0xB8	⇒	f64. convert_i32_u
0xB9	⇒	f64. convert_i64_s
0xBA	⇒	f64. convert_i64_u
0xBB	⇒	f64. promote_f32
0xBC	⇒	i32. reinterpret_f32
0xBD	⇒	i64. reinterpret_f64
0xBE	⇒	f32. reinterpret_i32
0xBF	⇒	f64. reinterpret_i64

WebAssembly Explorer

http://localhost:8080 170%

C99 -Os

COMPILE

```c
int foo(int x) {
  return x / 4;
}
```

WebAssembly (Text)

```
(module
  (export "foo" $func0)
  (func $func0
      (param $var0 i32)
      (result i32)
    get_local $var0
    i32.const 4
    i32.div_s
  )
)
```

x86 Assembly

```
wasm-function[0]:
  sub rsp, 8
  mov eax, edi
  sar eax, 0x1f
  shr eax, 0x1e
  add eax, edi
  sar eax, 2
  nop
  add rsp, 8
  ret
```

WebAssembly Explorer

http://localhost:8080 170%

C99 -Os

COMPILE

```c
int foo(unsigned int x) {
  return x / 4;
}
```

WebAssembly (Text)

```
(module
  (export "foo" $func0)
  (func $func0
      (param $var0 i32)
      (result i32)
    get_local $var0
    i32.const 2
    i32.shr_u
  )
)
```

x86 Assembly

```
wasm-function[0]:
  sub rsp, 8
  mov ecx, edi
  mov eax, ecx
  shr eax, 2
  nop
  add rsp, 8
  ret
```

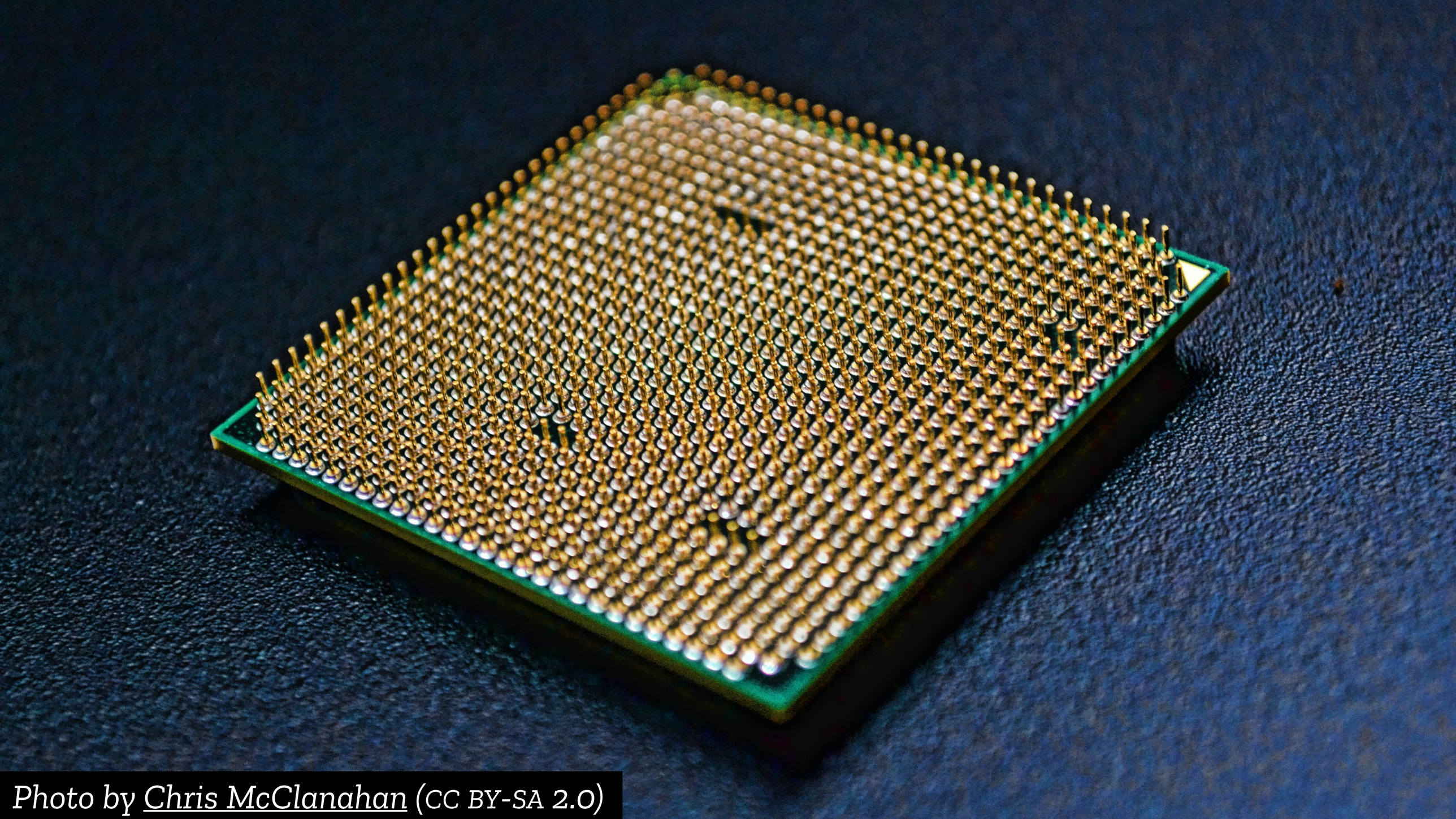
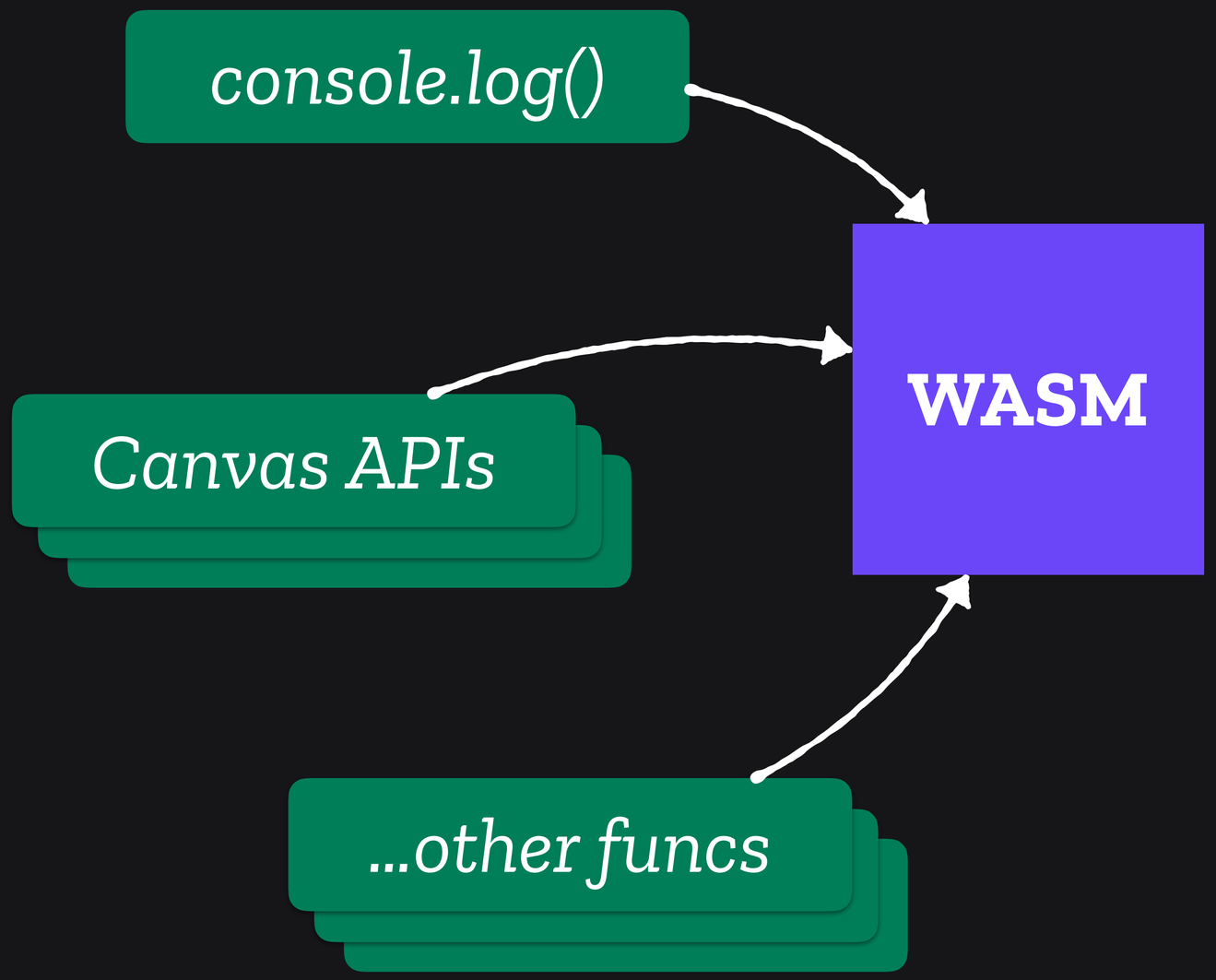
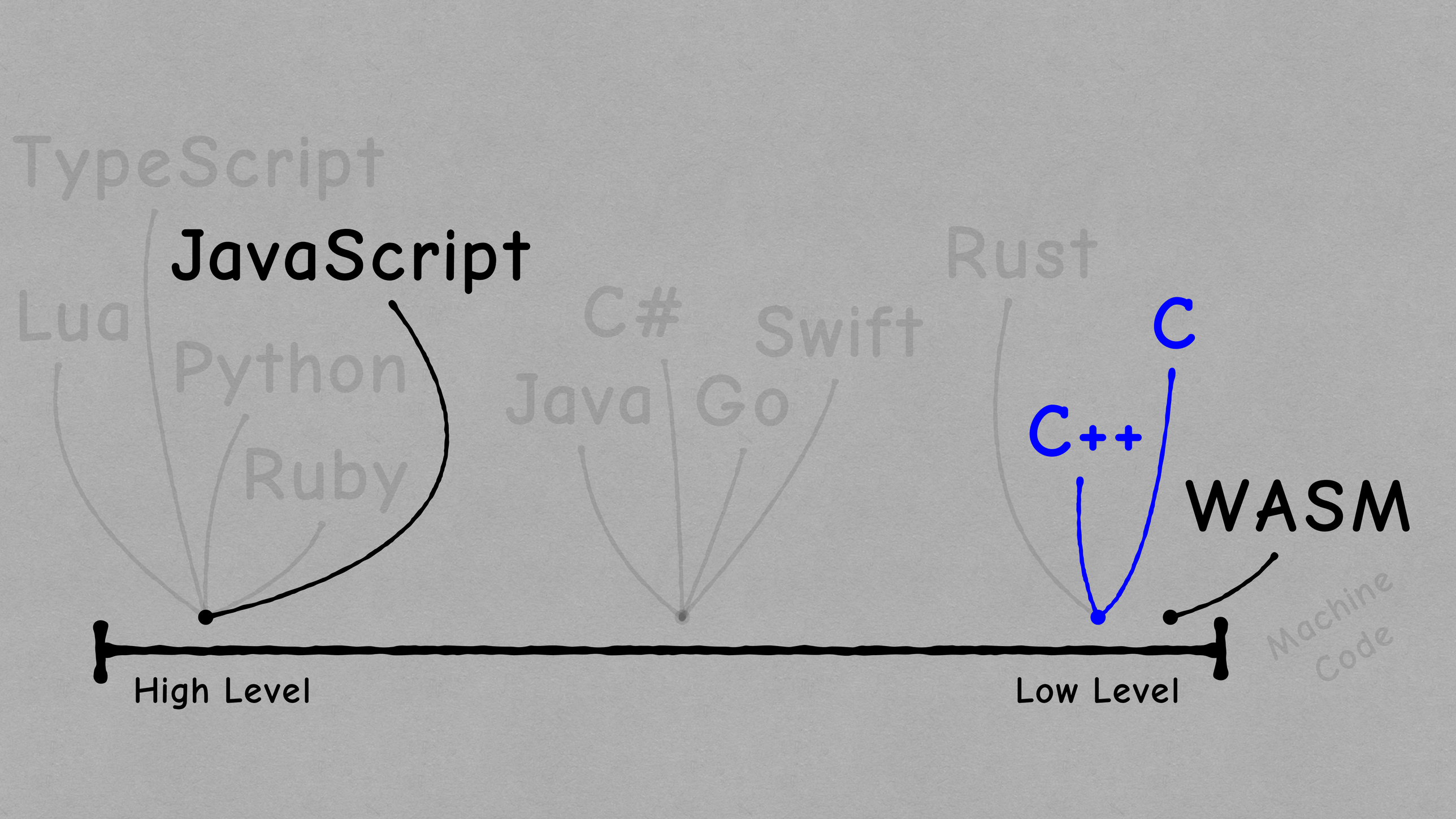


Photo by Chris McClanahan (CC BY-SA 2.0)


console.log()
Canvas APIs
...other funcs
WASM


TypeScript
JavaScript
Lua
Python
Ruby
C#
Java
Go
Swift
Rust
C
C++
WASM
Machine Code
High Level
Low Level

Firefox Nightly File Edit View History Bookmarks Tools Window Help 75% Nov 20 3:02

Unity WebGL Player | Tanks!

localhost:8000/Tanks/

unity WebGL

Tanks!

Firefox Nightly File Edit View History Bookmarks Tools Window Help 70% Nov 20 3:07

EpicZenGarden

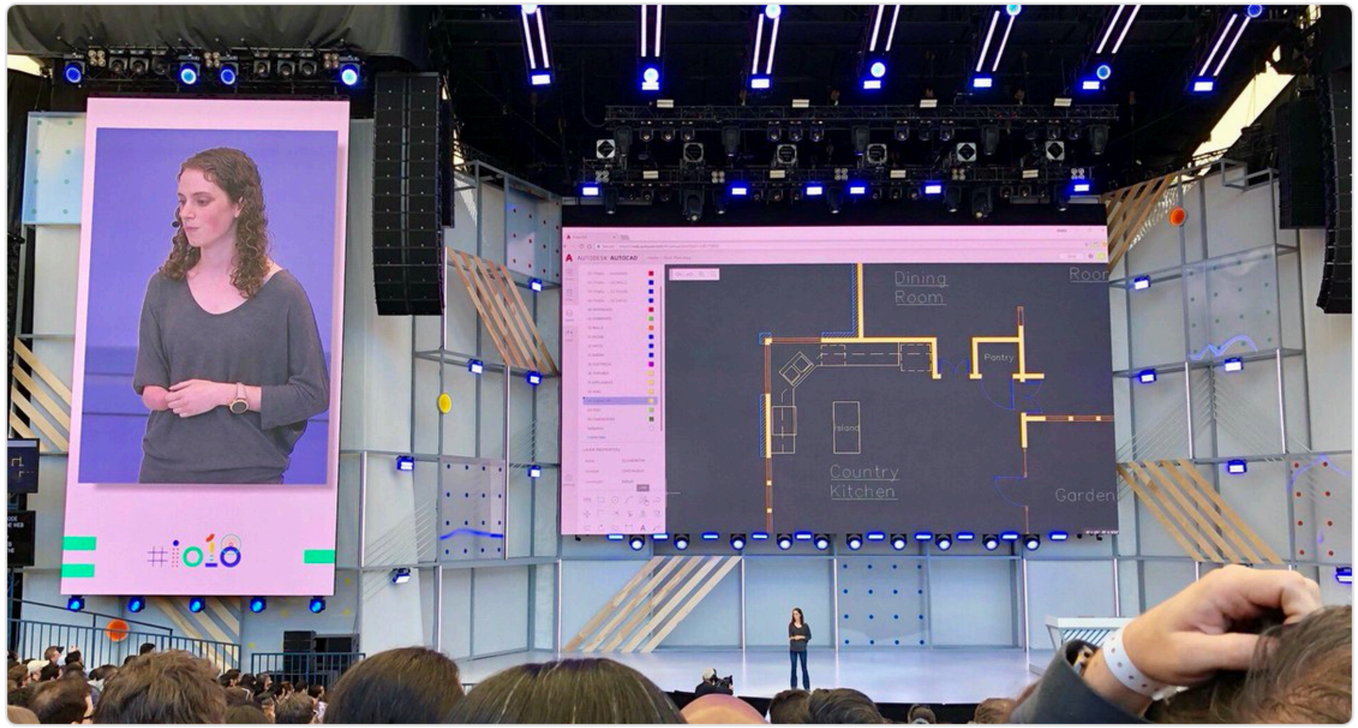
localhost:8000/ZenGarden/ZenGarden/EpicZenGarden.ht

Clear cache (243 MB) FullScreen

AutoCAD
@AutoCAD

Follow

“The AutoCAD team took a 35 year old code base...and got it to compile directly in a browser using WebAssembly.... So now the power of AutoCAD is just a link away.” #io18 #io2018keynote



2:20 PM - 8 May 2018 from San Francisco, CA

```
C:\>
```

Firefox Developer Edition File Edit View History Bookmarks Tools Window Help 33% 22:36

DOSBox SVN, CPU speed: max

assets.metacade.com/emulators/win311vr.html

http://assets.metacade.com/emulators/win311vr.html

Program Manager

File Options Window Help

NCSA Mosaic

WS_FTPLE

mIRC

Eudora Light

Games

Netscape 2.0

Netscape Navigator Gold

Accessories

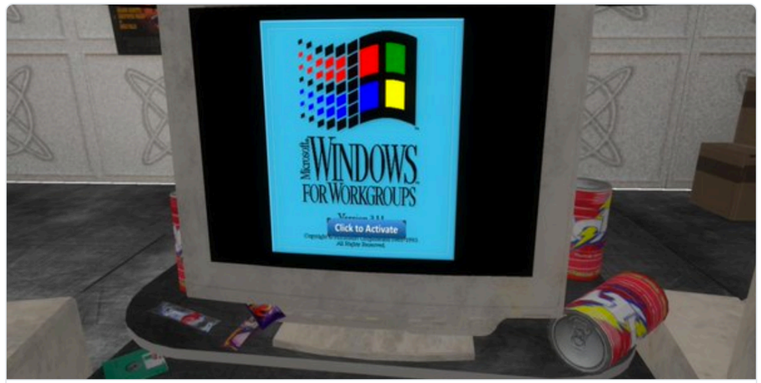
Main

MAME

Click to Activate

James Baicoianu
@bai0

Windows 3.1 is 25 this month 🎂🎉🍾 Now you can use it in #WebVR thanks to #emscripten #wasm #threejs and #janusweb



Windows 3.1 in WebVR | JanusWeb
Happy 25th birthday, Windows 3.1!
assets.metacade.com

1:54 AM - 10 Apr 2017


SOON

When should you reach for WebAssembly?

Re-use existing code
Use “better” languages

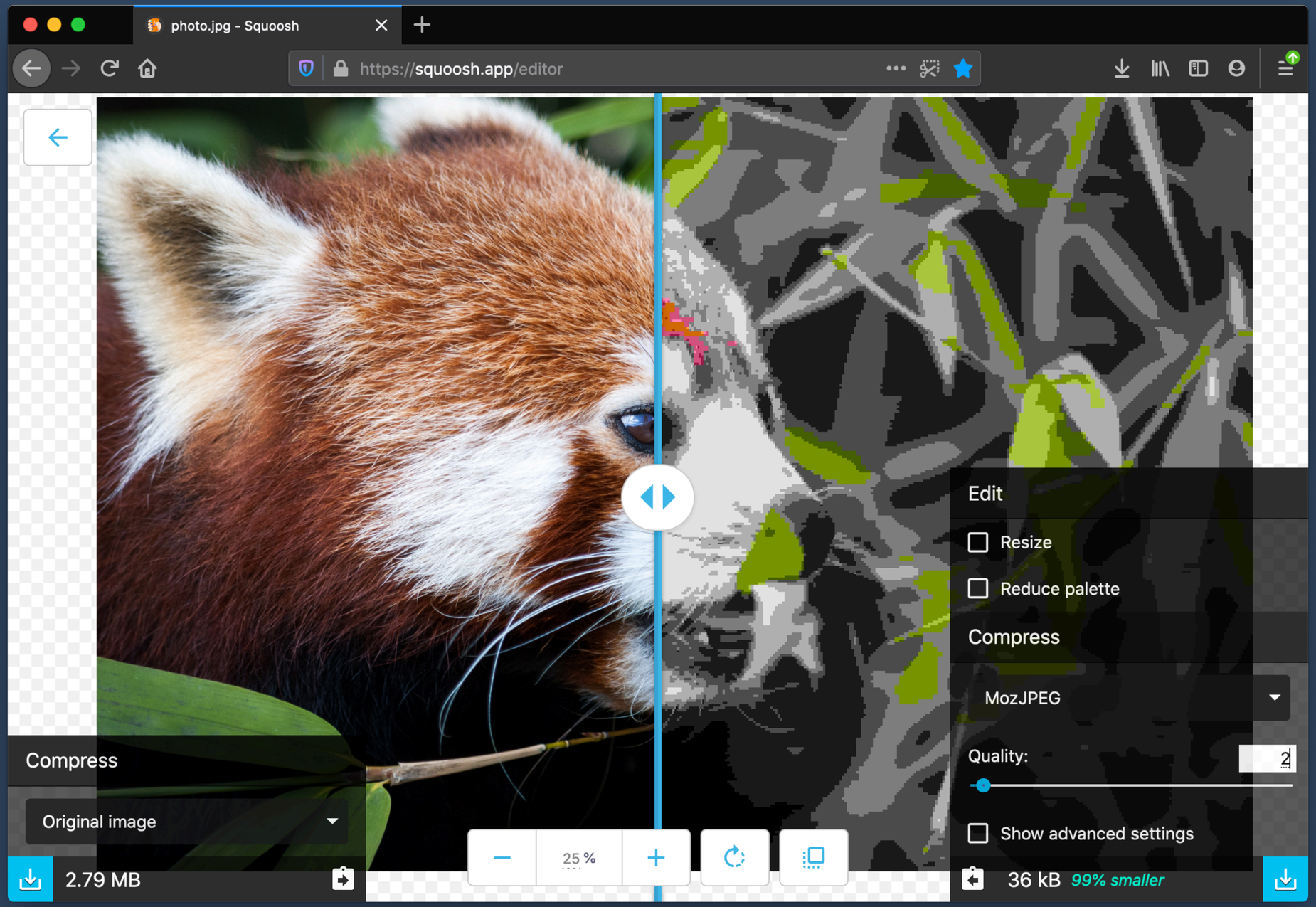


photo.jpg - Squoosh
https://squoosh.app/editor
Edit
Resize
Reduce palette
Compress
MozJPEG
Quality:
2
Show advanced settings
Compress
Original image
2.79 MB
25 %
36 kB
99% smaller

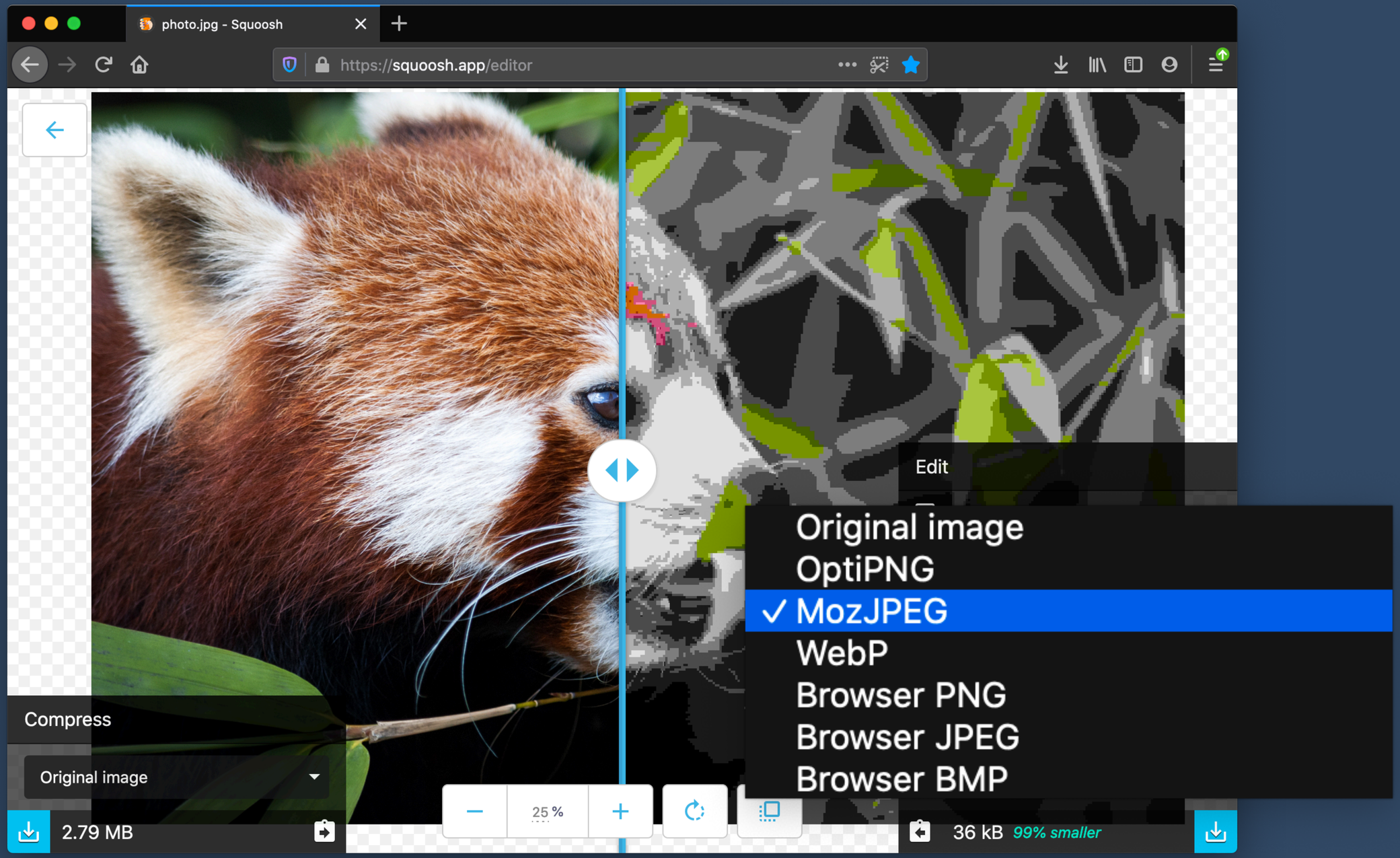


photo.jpg - Squoosh
https://squoosh.app/editor
Edit
Original image
OptiPNG
MozJPEG
WebP
Browser PNG
Browser JPEG
Browser BMP
Compress
Original image
25 %
2.79 MB
36 kB 99% smaller

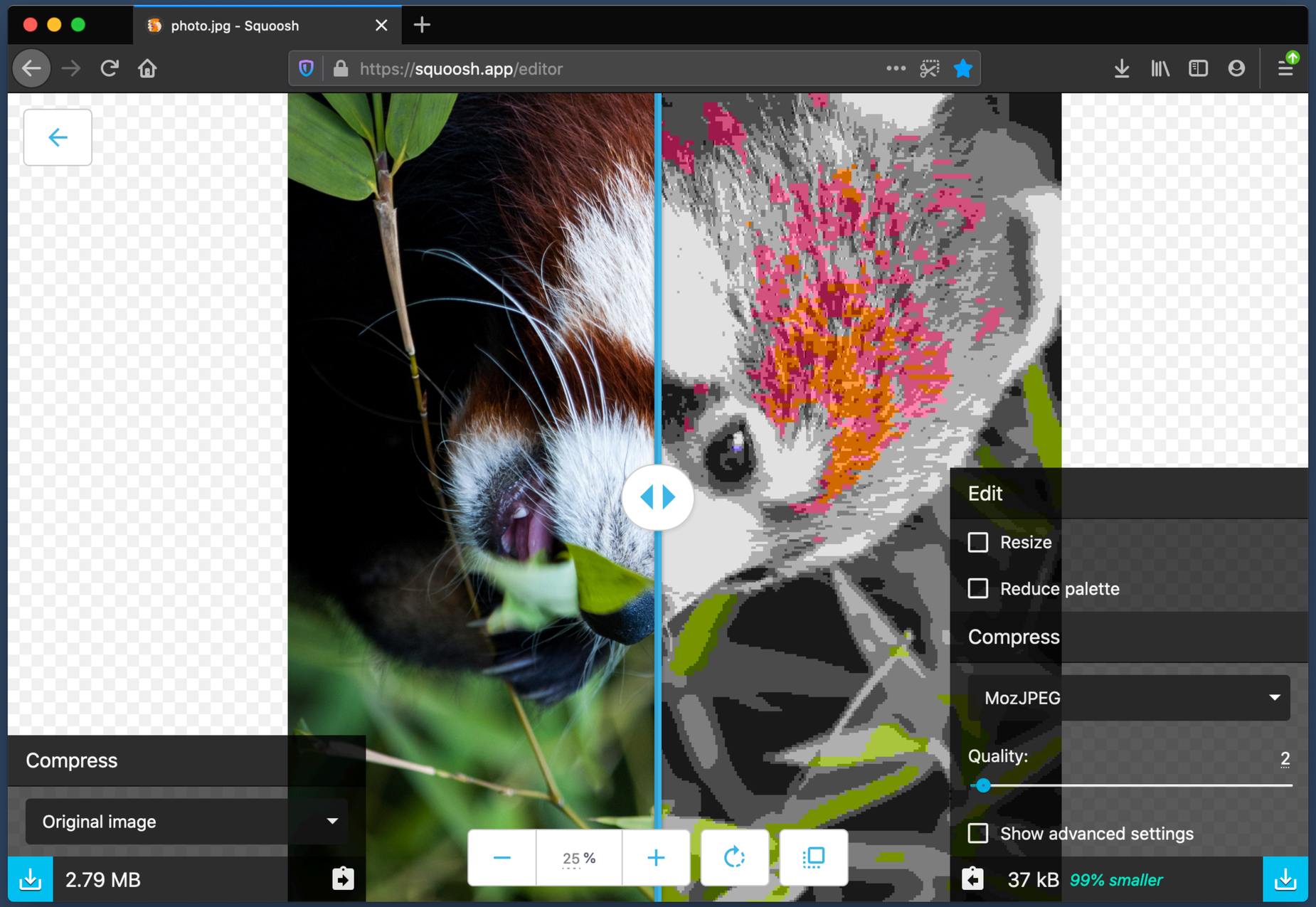
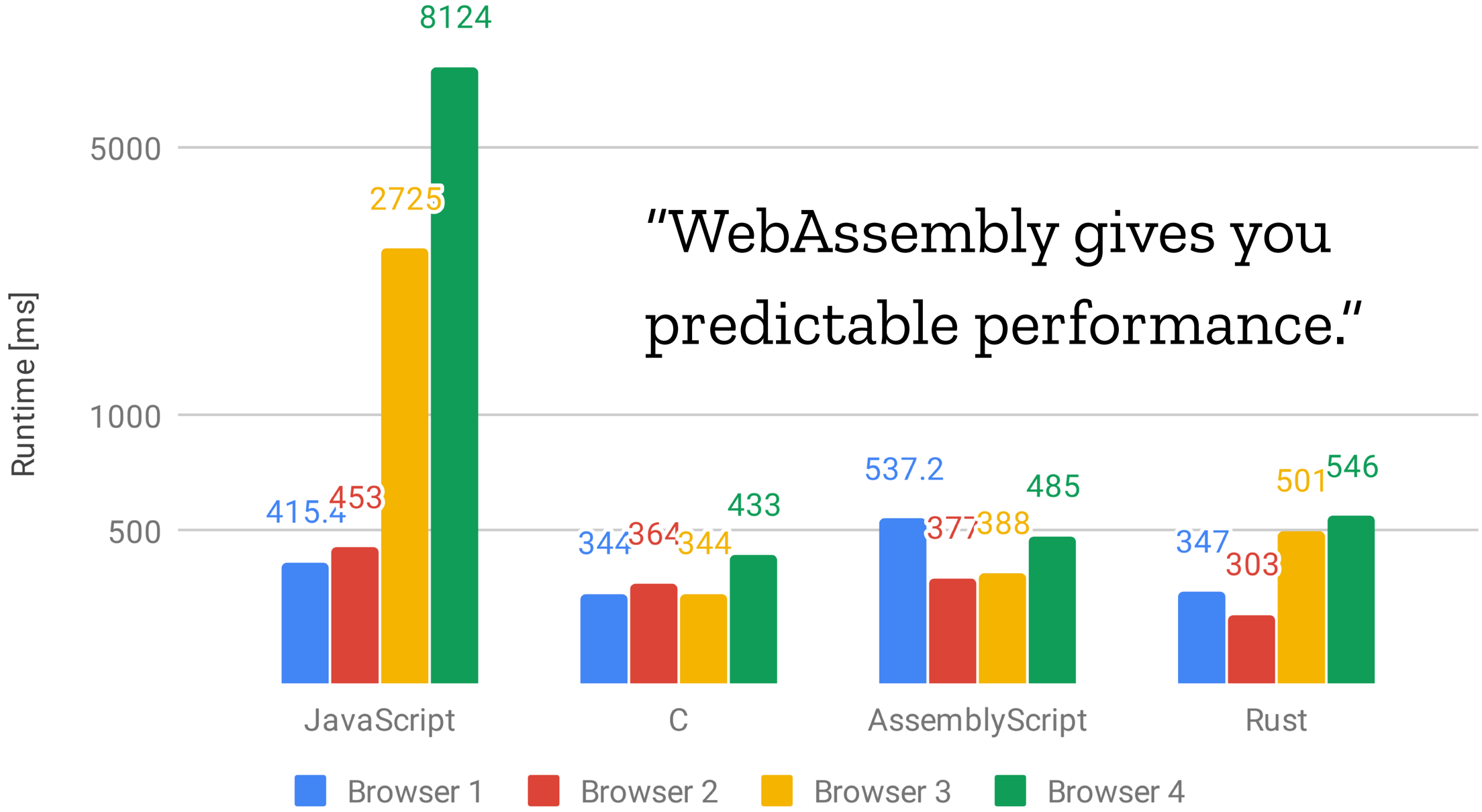
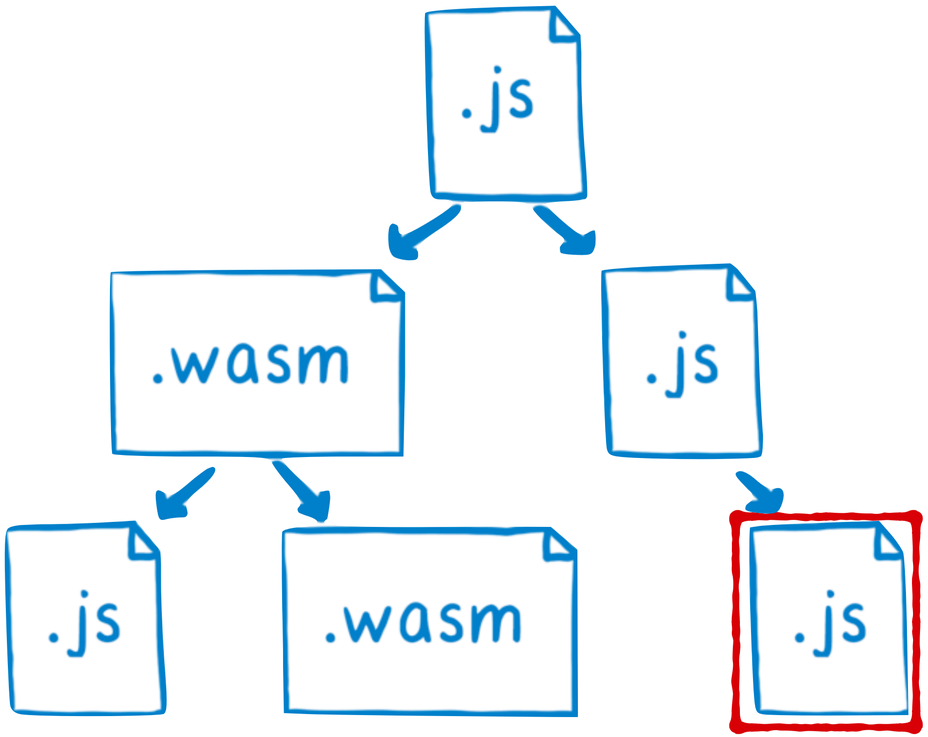


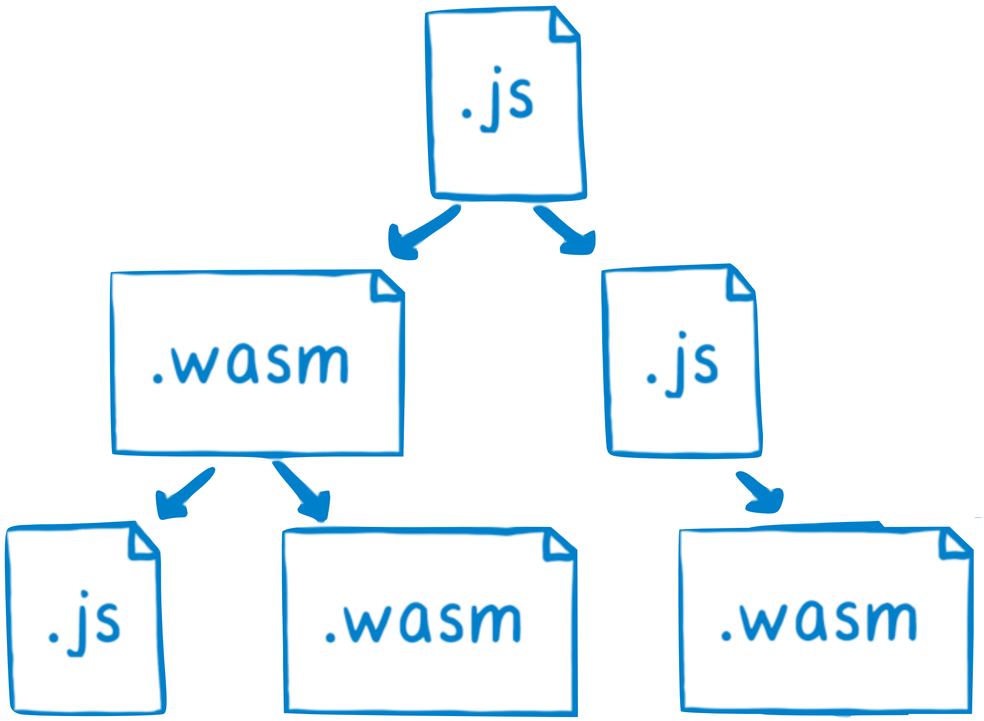
photo.jpg - Squoosh
https://squoosh.app/editor
Edit
Resize
Reduce palette
Compress
MozJPEG
Quality:
2
Show advanced settings
37 kB
99% smaller
Compress
Original image
2.79 MB
25 %

Speed comparison per language (lower is better)



"WebAssembly gives you predictable performance."


.js
.wasm
.js
.js
.wasm
.js

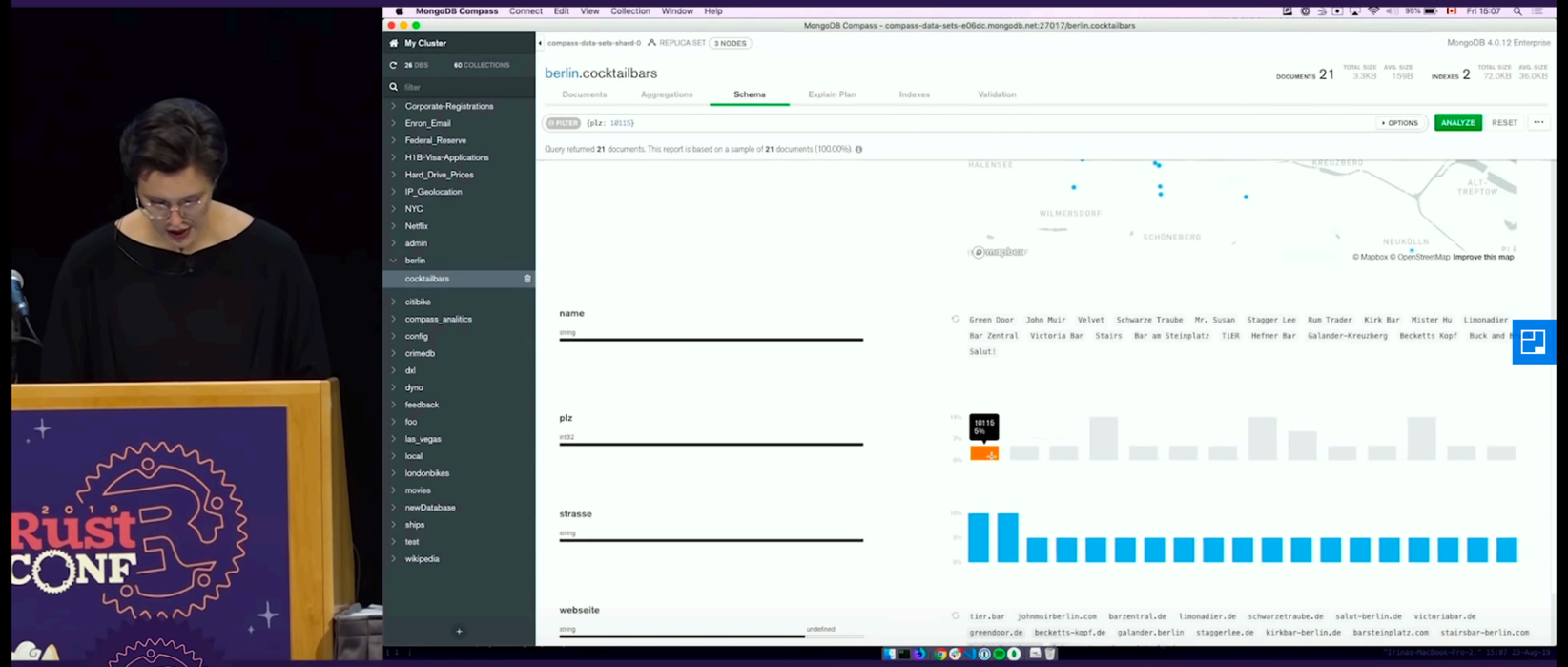

.js
.wasm
.js
.js
.wasm
.wasm

RustConf 2019 - From Electron,

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

YouTube

Search



7:02 / 36:38

RustConf 2019 - From Electron, to Wasm, to Rust (Aaand Back to Electron) by Irina Shestak

15,542 views • Sep 16, 2019

385 17 SHARE SAVE

RustConf 2019 - From Electron,

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

YouTube

Search

JS

121000ms → 3000ms

* for 10000 documents

Rust

13:13 / 36:38

RustConf 2019 - From Electron, to Wasm, to Rust (Aaand Back to Electron) by Irina Shestak

15,542 views • Sep 16, 2019

385 17 SHARE SAVE

Oxidizing Source Maps with Rus

https://hacks.mozilla.org/2018/01/oxidizing-source-maps-with

mozilla HACKS

Download Firefox

Search Mozilla Hacks

Oxidizing Source Maps with Rust and WebAssembly

By Nick Fitzgerald

Share This

Edit: Further algorithmic improvements yielded speedups of up to **10.9x faster** than the original

what is described here, for total

out these extra gains in Speed

Without Wizardry!

Tom Tromey and I have replaced

source map parser with Rust cod

The WebAssembly is up to **5.89 times faster**

the JavaScript implementation on realistic benchmarks operating on real world source maps. Additionally, performance is also more consistent: relative standard deviations decreased.

We removed JavaScript code that had been written in a convoluted and unidiomatic way in the name of performance, and replaced it with idiomatic Rust code that performs even better.

RustConf 2019 - From Electron,

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

YouTube

Search

Rust lets you write performant and clean code right off the bat.

@_Irina

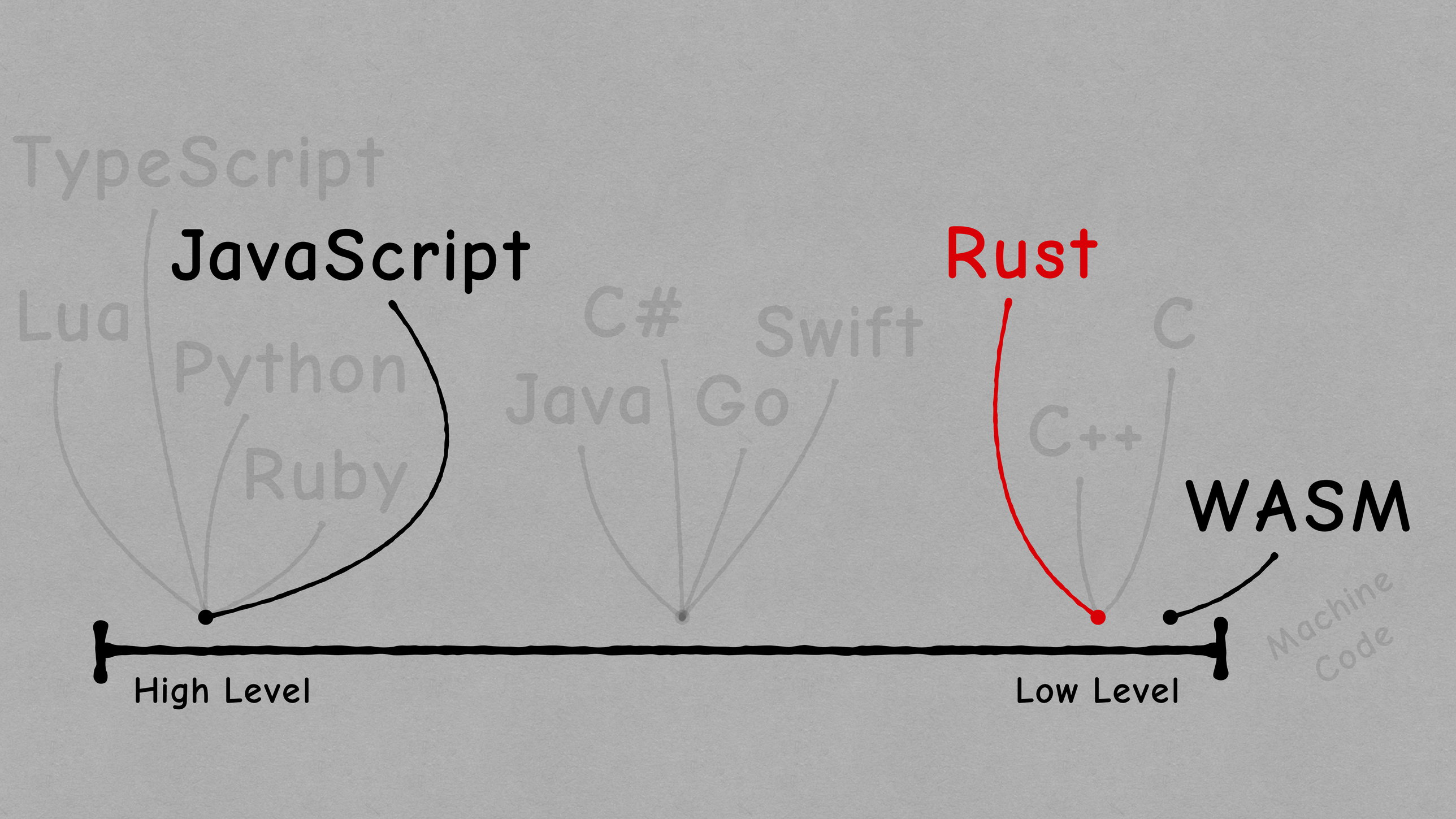
RustConf | Aug 2019

19:48 / 36:38

RustConf 2019 - From Electron, to Wasm, to Rust (Aaand Back to Electron) by Irina Shestak

15,542 views • Sep 16, 2019

385 17 SHARE SAVE


TypeScript
JavaScript
Rust
Lua
Python
C#
Swift
C
Java
Go
C++
Ruby
WASM
Machine Code
High Level
Low Level

~~What *is* WebAssembly~~

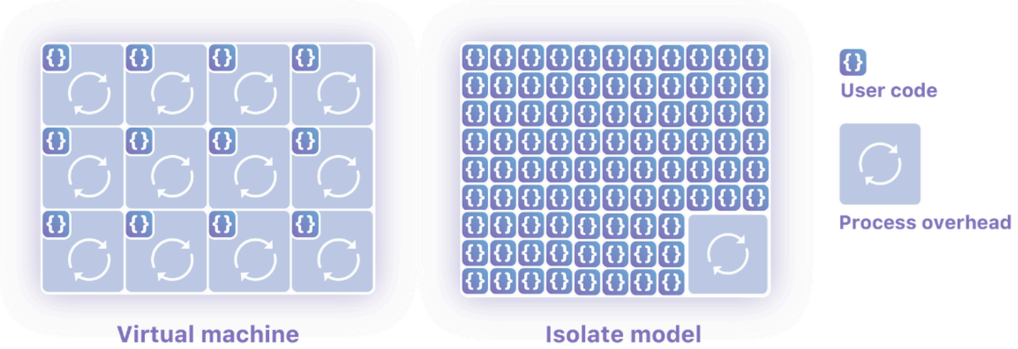
...outside the browser?

HTML / CSS / JS

HTML / CSS / JS

HTML / CSS / JS / WASM

A single runtime can run hundreds or thousands of isolates, seamlessly switching between them. Each isolate's memory is completely isolated, so each piece of code is protected from other untrusted or user-written code on the runtime. Isolates are also designed to start very quickly. Instead of creating a virtual machine for each function, an isolate is created within an existing environment. This model eliminates the cold starts of the virtual machine model.



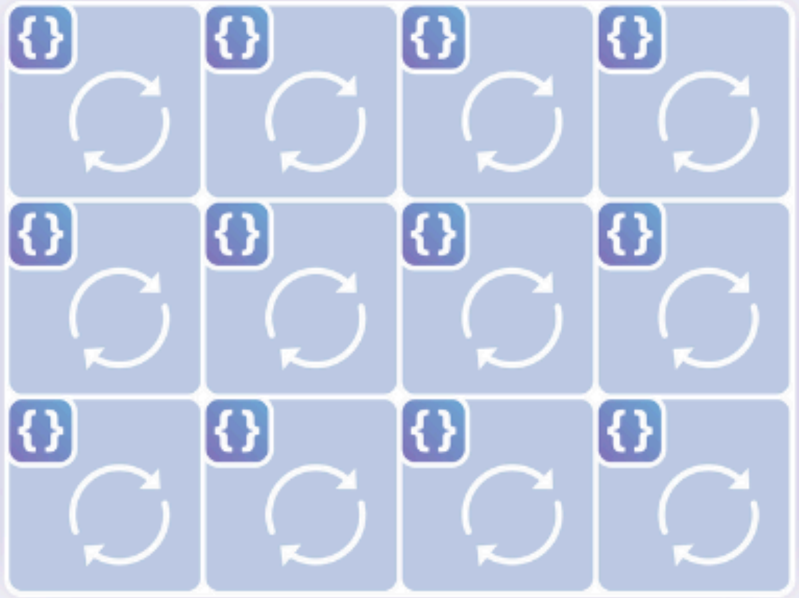
Unlike other serverless providers which use containerized processes each running an instance of a language runtime, Workers pays the overhead of a JavaScript runtime once on the start of an edge container. Workers processes are able to run essentially limitless scripts with almost no individual overhead by creating an isolate for each Workers function call. Any given isolate can start around a hundred times faster than a Node process on a container or virtual machine. Notably, on startup isolates consume an order of magnitude less memory.

How It Works - Cloudflare Work

https://workers.cloudflare.com/docs/reference/workers-co

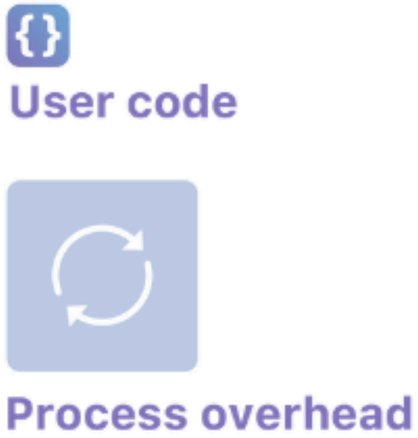
CLOUDFLARE

DEVELOPERS DOCS SPONSORSHIPS OPEN SOURCE SERVERLIST NEWSLETTER



Virtual machine

Isolate model



Bindings

Routes

Runtime

APIs

essentially limitless scripts with almost no individual overhead by creating an isolate for each Workers function call. Any given isolate can start around a hundred times faster than a Node process on a container or virtual machine. Notably, on startup isolates consume an order of magnitude less memory.

HTML / CSS / JS / WASM

HTML / CSS / JS / WASM

Wasmtime — a small and efficient r

https://wasmtime.dev

Wasmtime

A small and efficient runtime for WebAssembly & WASI

Start using the latest developments in WebAssembly and WASI with the Wasmtime runtime.

Wasmtime supports the WebAssembly Interface Types proposal:

Installing Wasmtime

It looks like you're running macOS or Linux. To download and install Wasmtime, run the following in your terminal, then follow the on-screen instructions.

```
curl https://wasmtime.dev/install.sh -s
```

Announcing Lucet: Fastly's nati

https://www.fastl

fastly

Sign Up

Announcing Lucet: Fastly's native WebAssembly compiler and runtime

By Pat Hickey, Sr. software engineer, March 28, 2019 in Edge

Today, we are thrilled to announce the open sourcing of Lucet, Fastly's native WebAssembly compiler and runtime. WebAssembly is a technology created to enable web browsers to safely execute programs at near-native speeds. It has been shipping in the four major browsers since early 2017.

Lucet is designed to take WebAssembly beyond the browser, and build a platform for faster, safer execution on Fastly's edge cloud. WebAssembly is already supported by many languages including Rust,

BYTECODE ALLIANCE

moz://a fastly intel Red Hat

WASI |

https://wasi.dev

WASI

WASI

The WebAssembly System Interface

WASI is a modular system interface for WebAssembly. As described in the initial announcement, it's focused on security and portability.

WASI is being standardized in a subgroup of the WebAssembly CG. Discussions happen in GitHub issues, pull requests, and bi-weekly Zoom meetings.

For a quick intro to WASI, including getting started using it, see the intro document.

The Wasmtime runtime's tutorial contains examples for how to target WASI from C and Rust. The resulting .wasm modules can be run in any WASI-compliant runtime.

For more documentation, see the documents guide.

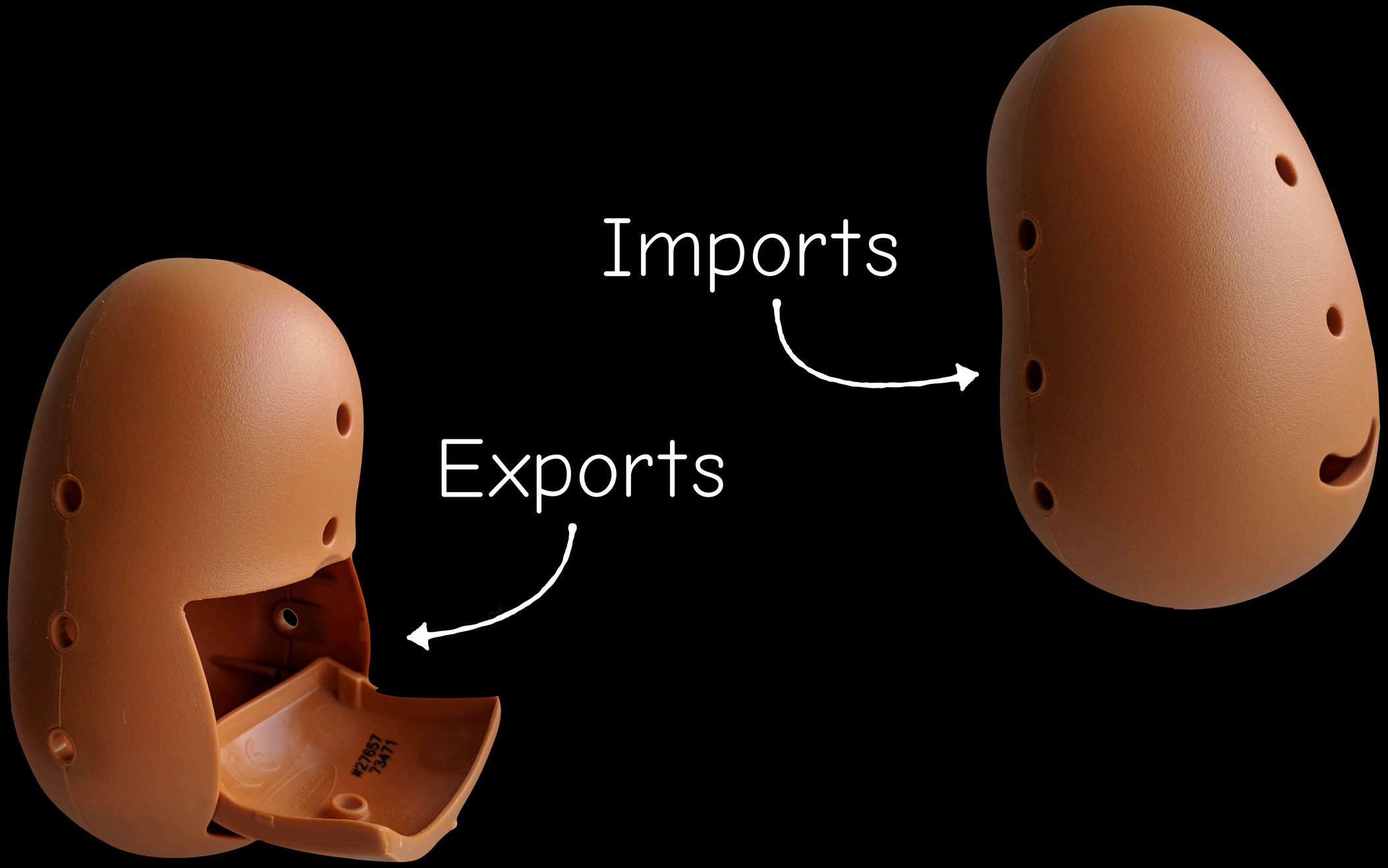
Photo by Renoir Gaither

x86

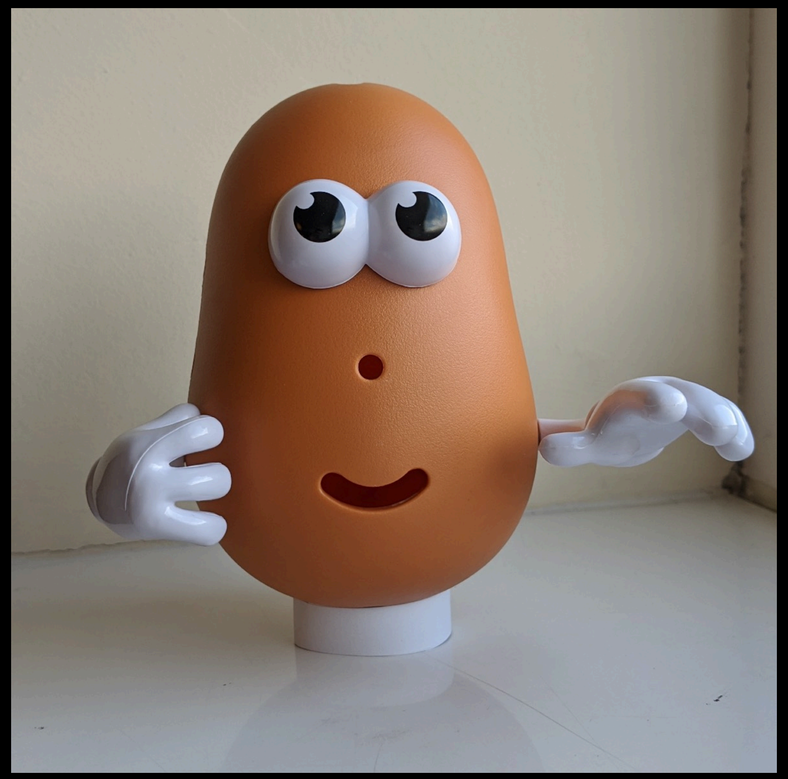
x64

ARMv8

WASM

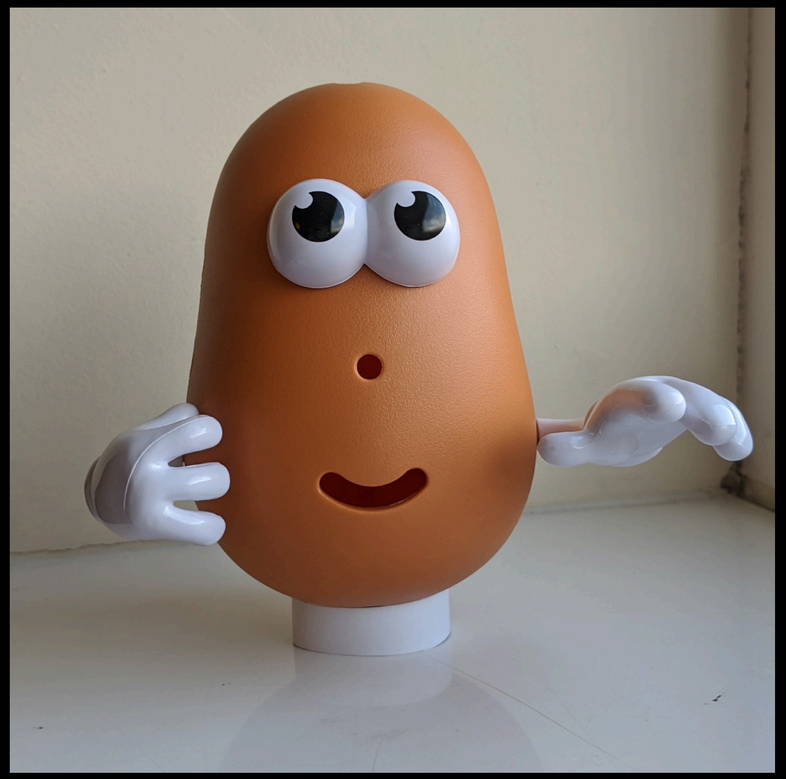

Imports
Exports
#271657
73471

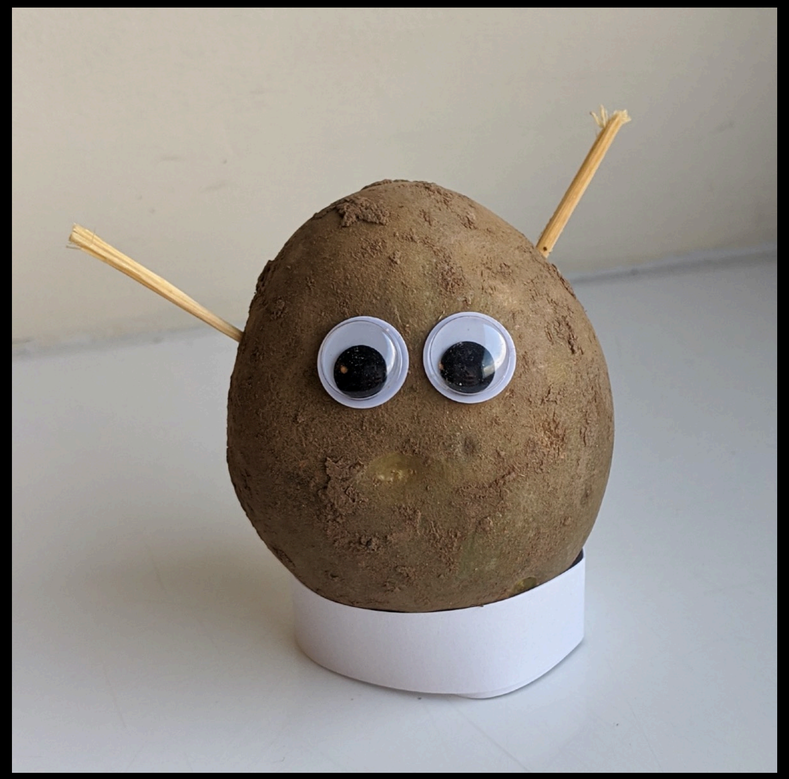
Math



Read/Write Files

...and other capabilities





WASM Module

Machine Code

```
(module
    (import "wasi_unstable" "fd_write"
            (func $fd_write (param i32 i32 i32 i32)
                            (result i32))) ;; number of bytes written

    (memory 1) ;; Allocate one 64 KiB page of memory

    ;; Write 'hello world\n' to memory at an offset of 8 bytes
    (data (i32.const 8) "hello world\n")

    (func $main (export "_start")
        (i32.store (i32.const 0) (i32.const 8))   ;; iov.iov_base
        (i32.store (i32.const 4) (i32.const 12))  ;; iov.iov_len

        (call $fd_write
            (i32.const 1)  ;; file_descriptor - 1 for stdout
            (i32.const 0)  ;; *iovs
            (i32.const 1)  ;; iovs_len
            (i32.const 20) ;; nwritten
        )
        drop ;; drop number of bytes written from top of stack
    )
)
```

```
(module
    (import "wasi_unstable" "fd_write"
            (func $fd_write (param i32 i32 i32 i32)
                            (result i32))) ;; number of bytes written



        (call $fd_write
            (i32.const 1)  ;; file_descriptor - 1 for stdout
            (i32.const 0)  ;; *iovs
            (i32.const 1)  ;; iovs_len
            (i32.const 20) ;; nwritten
        )
```

fish ~/demo

```
demo  cat src/main.rs

File: src/main.rs

1  fn main() {
2      println!("Hello, world!");
3  }

demo  cargo wasi build --release
   Compiling demo v0.1.0 (/Users/dan/demo)
    Finished release [optimized] target(s) in 0.21s
  Optimizing with wasm-opt
demo  wasmtime ./target/wasm32-wasi/release/demo.wasm
Hello, world!
demo
```

fish ~/demo

```
demo  wasm2wat ./target/wasm32-wasi/release/demo.wasm
        | rg -n '"fd_write"'
26:  (import "wasi_unstable" "fd_write" (func $fd_write (type 10)))
demo
```

fish ~/demo

```
demo  wasmtime ./target/wasm32-wasi/release/demo.wasm
Hello, world!
```

WASI Web Polyfill

https://wasi.dev/polyfill/ 170%

Browse... demo.wasm

```
Hello, world!
```

This is a simple Web polyfill demo of WASI, a portable system interface for WebAssembly, allowing simple WASI programs that print to stout to be run in a browser. See wasi.dev for more information on using WASI.

WASI

Lightweight Sandboxing

fish ~/demo

```
demo ❯ cargo build --release
   Compiling demo v0.1.0 (/Users/dan/demo)
    Finished release [optimized] target(s) in 0.35s
demo ❯ ./target/release/demo
Hello, world!
demo ❯
```

fish ~/demo

```
demo ❯ cargo build --release
   Compiling demo v0.1.0 (/Users/dan/demo)
    Finished release [optimized] target(s) in 0.35s
demo ❯ ./target/release/demo
Hello, world!
demo ❯ cargo wasi build --release
   Compiling demo v0.1.0 (/Users/dan/demo)
    Finished release [optimized] target(s) in 0.44s
  Optimizing with wasm-opt
demo ❯ wasmtime ./target/wasm32-wasi/release/demo.wasm
Error: Custom { kind: Other, error: "failed to find a preopened file
descriptor through which \"/Users/dan/.ssh/id_rsa\" could be opened"
}
1 demo ❯
```

fish ~/demo

```
demo ❯ cargo build --release
   Compiling demo v0.1.0 (/Users/dan/demo)
    Finished release [optimized] target(s) in 0.33s
demo ❯ ./target/release/demo
Hello, world!
demo ❯ cargo wasi build --release
   Compiling demo v0.1.0 (/Users/dan/demo)
    Finished release [optimized] target(s) in 0.30s
  Optimizing with wasm-opt
demo ❯ wasmtime ./target/wasm32-wasi/release/demo.wasm
Error: Custom { kind: Other, error: "failed to find a preopened file
descriptor through which \"/Users/dan/.ssh/id_rsa\" could be opened"
}
1 demo ❯ wasmtime --dir=/Users/dan/.ssh/ \
         ./target/wasm32-wasi/release/demo.wasm
Hello, world!
demo ❯
```

Yet another malicious package found in npm, targeting cryptocurrency wallets

JUNE 17, 2019 | IN APPLICATION SECURITY, ECOSYSTEMS, VULNERABILITIES | BY SIMON MAPLE

Cryptocurrency wallet developer Komodo has been in the news recently as the most recent victim of an attempted cryptocurrency attack by malicious code injection via npm dependencies. The EasyDEX-GUI

Universal binary format

Safe, portable runtimes

Wasmtime
as a Python module

demo.py

```python
import wasmtime
import markdown

print(markdown.render('# Hello, Python!'))
```

demo.py

```python
import wasmtime
import markdown

print(markdown.render('# Hello, Python!'))
```

Rust Library compiled to WebAssembly

demo.py

```python
import wasmtime
import markdown

print(markdown.render('# Hello, Python!'))
```

Rust Library compiled to WebAssembly

```
~/demos/python $ python3 demo.py
<h1>Hello, Python!</h1>
```

JS demo.mjs

```
1   import { render } from "./markdown.wasm";
2   console.log(render("# Hello, Node"))
```

⌥⌘1 screencast

```
🦀 ~/demos/node $ node --experimental-modules --loader ./loader.mjs
(node:951) ExperimentalWarning: The ESM module loader is experimenta
(node:951) ExperimentalWarning: --loader is an experimental feature.
y time
<h1>Hello, Node!</h1>
```

My Wasm + WebIDL bindings proje

localhost:8080 150%

Input markdown:

```
# Hello, Web!
```

Rendered markdown:

Hello, Web!

goto; copenhagen

GOTO Copenhagen 2019
Conference Nov. 18 - 20

Thank You

Dan Callahan
@callahad

More Info

Follow us @gotocph