WebAssembly and the Future of the Web

ABOUT ME

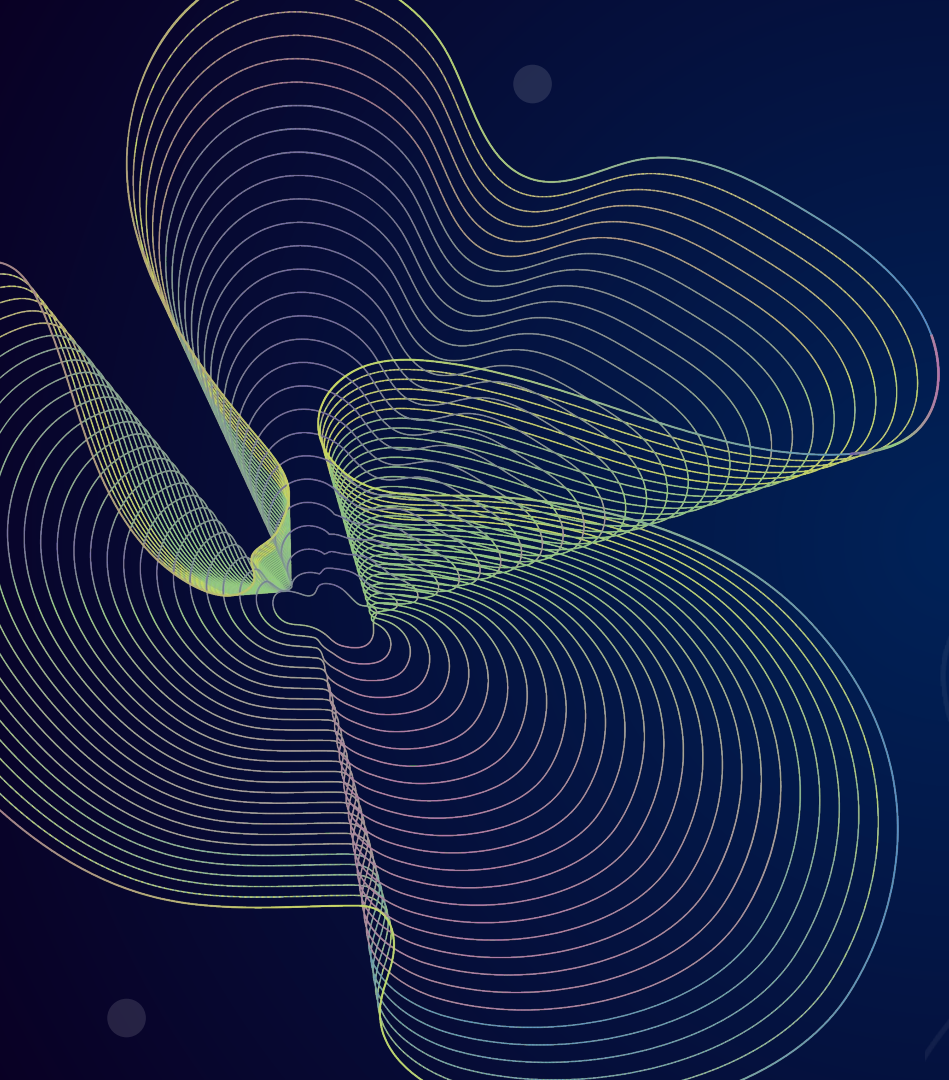
- Mxs. Kassian Rosner Wren
- Developer Advocate Extraordinaire
- Twitch streamer of hardware and software development @nodebotanist
- Author of 2 books on JavaScript robotics
- Catparent to a cantankerous old man and a very skittish girl

WHAT WEBASSEMBLY IS

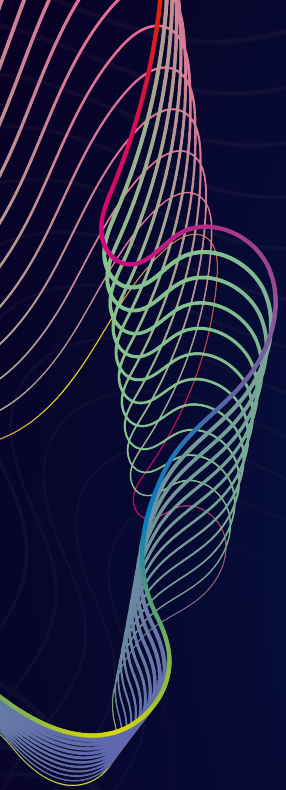
WHY WEBASSEMBLY MATTERS

HOW TO USE WEBASSEMBLY

THE FUTURE OF THE WEB WITH WEBASSEMBLY



WHAT WEBASSEMBLY IS



THE PRECURSOR TO WEBASSEMBLY: ASM.JS

- Created by a team at Mozilla in 2013
- Allowed developers to use a subset of JS that enabled low-level system calls
- Had a type “hint” system (a | 0 meant a 32-bit integer)
- Made a splash in 2014 when the Unreal game engine was ported to asm.js to run in browsers

HISTORY OF WEBASSEMBLY

2013-14

ASM.js is created to allow low-level instructions to be executed by JS

2015

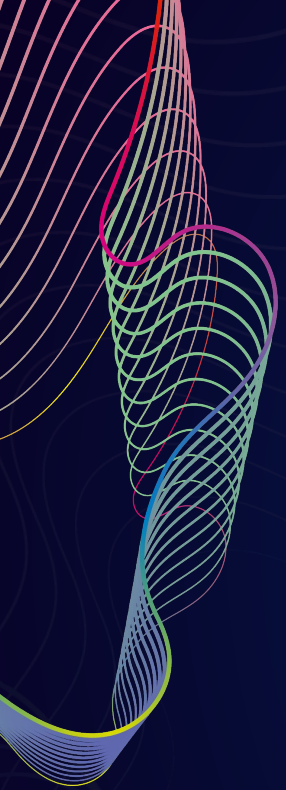
Browser makers team up to create the WASM specification team

2017

WASM version 1 is implemented in all modern browsers

NOW

WASM is widely available, with all modern browser support and several languages providing compilers



THE BASICS OF WEBASSEMBLY

- Created by a team composed of engineers from major browser makers (Google, Microsoft, Mozilla, and Apple)
- An assembly-like language that can be written in, but is also meant to be a transpile target for other high-level languages
- Shares the VM with JavaScript-- the first language to ever do this!
- Allows the browser to take advantage of bytecode-level instructions

THE PROCESS OF WEBASSEMBLY

HIGH-LEVEL CODE

Write your code in a high-level programming language

TRANSPILE

Use a transpiler to turn your code into WebAssembly text code

LOAD AND COMPILE

Load the WebAssembly into the browser, which compiles it into bytecode

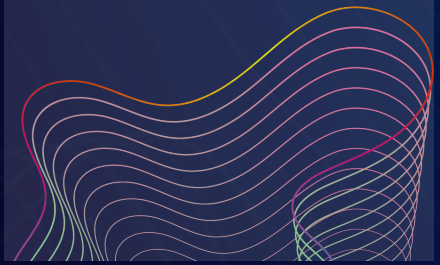
RUN

The browser runs the compiled bytecode

THE STRUCTURE OF WEBASSEMBLY MEMORY

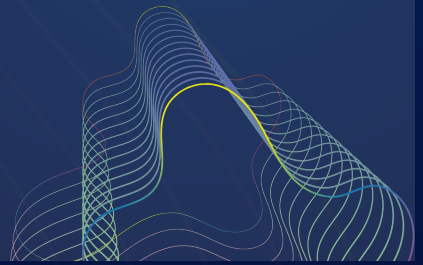
INSTRUCTIONS

The instructions transpiled from the higher-level code



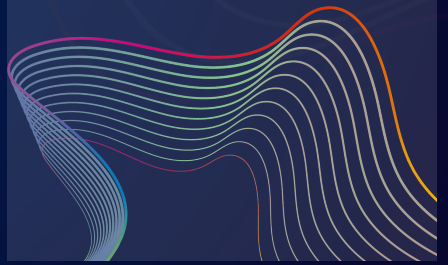
EXECUTION STACK

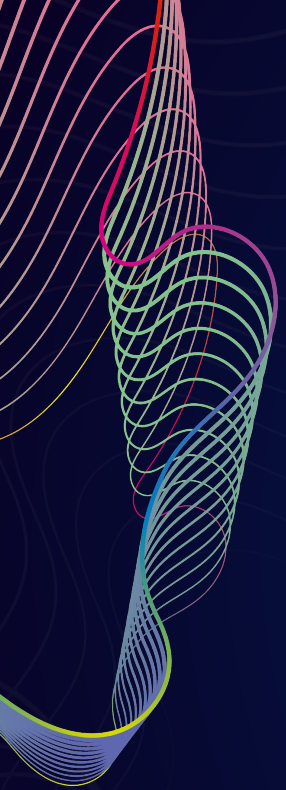
Execution stack, FIFO queue that stores intermediate function state



LINEAR MEMORY

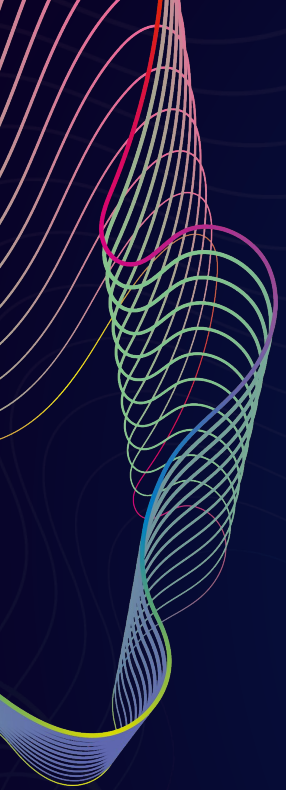
Heap-like memory that is shared with the JS code running the module





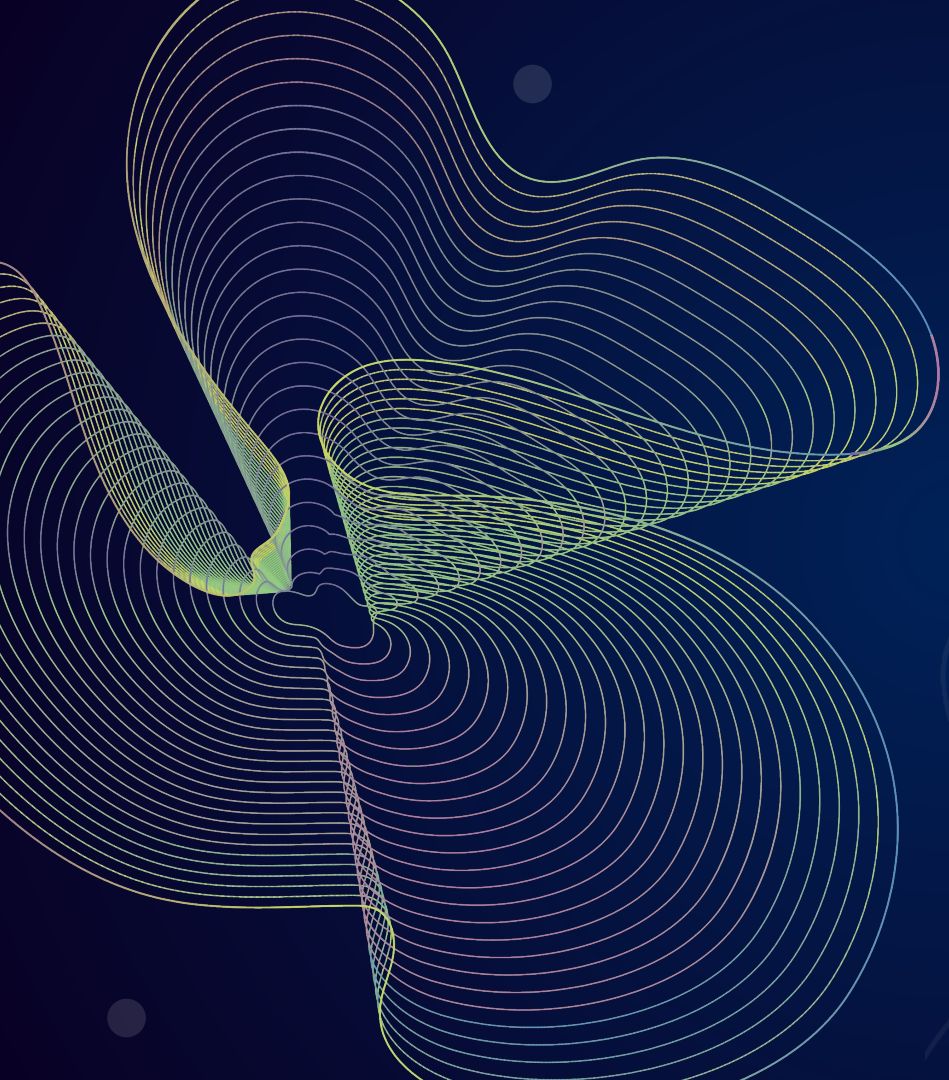
PERFORMANCE BENEFITS OF WEBASSEMBLY

- Decoding/compiling WebAssembly is simpler and can be split over multiple threads
- You can stream decoding and compilation, meaning the code can be ready to run the instant it's done downloading
- There is only one optimization step, and there is less need for monitoring the runtime of WebAssembly code



WebAssembly Quirks

- WebAssembly only supports 4 data types: 32 and 64-bit integers and floats; this means a lot of glue code needs to be written to pass complex data types between WebAssembly and JavaScript
- WebAssembly cannot perform I/O tasks, like sockets and filesystem functions
- WebAssembly doesn't have garbage collection (yet)



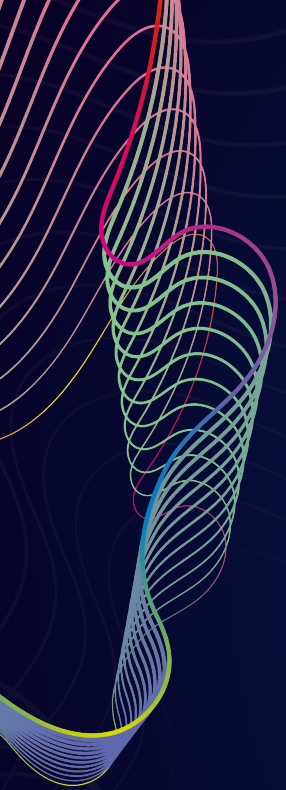
WHY WEBASSEMBLY MATTERS

ENHANCING THE BROWSER'S CAPABILITIES

ALLOWING RE-USE OF CODE BASES

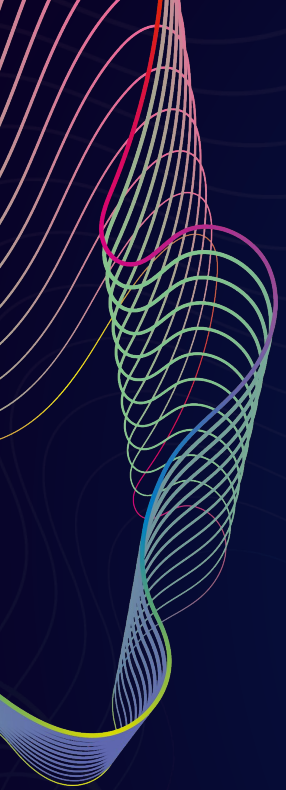
DOING THINGS JAVASCRIPT ISN'T GREAT AT

INCREASING PERFORMANCE OF WEB APPS



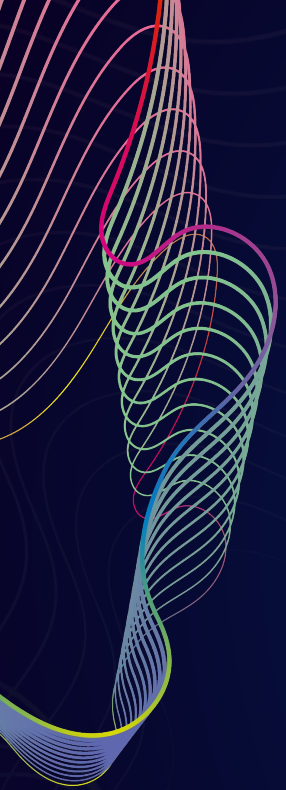
Enhancing the Browser's Capabilities

- While JavaScript still has to handle all the I/O code, WebAssembly allows many applications that could not operate in the browser due to complexity or performance to now operate on the web
- Low-level system operations used to be off-limits, but with WebAssembly not only are they available, but the fact that the browser compiles the code for whatever architecture it runs on makes it extremely easy to use.
- WebAssembly is a separate language designed to have backwards-compatibility, so adding to the WebAssembly feature set doesn't affect JavaScript



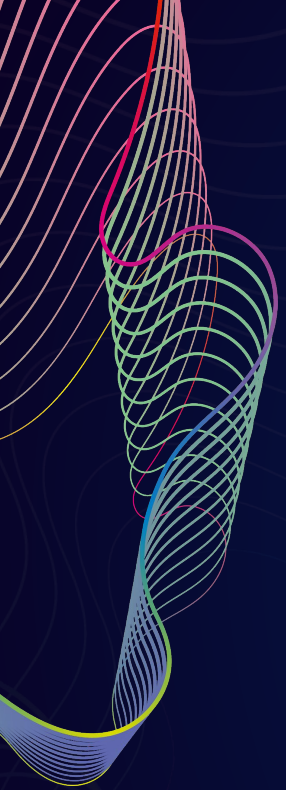
ALLOWS RE-USE OF CODE BASES

- Applications that used to be desktop-only can be ported to the web using WebAssembly
- Libraries no longer necessarily need to be re-written in JavaScript in order to operate in the browser
- Allows JavaScript to interact with and take advantage of vast amounts of code that used to be unusable in the browser
- Examples: AutoCAD, Google Earth



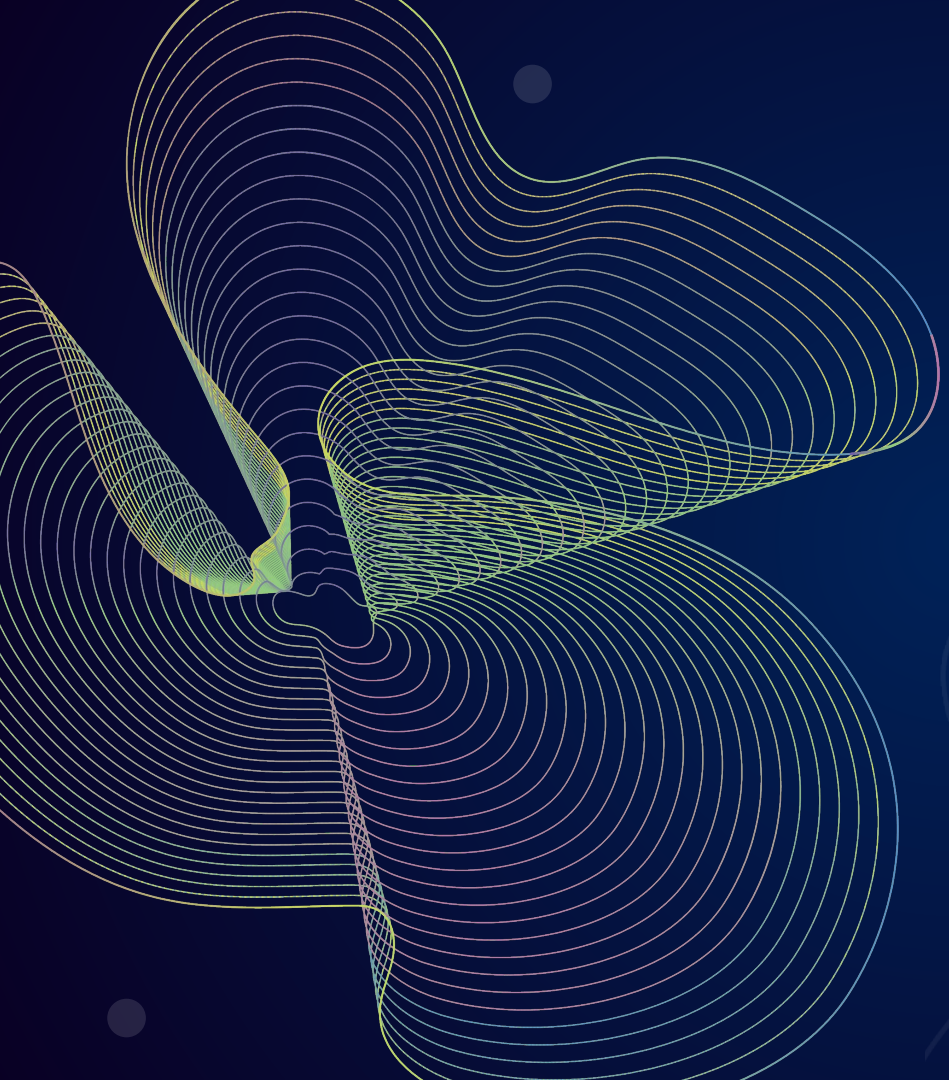
DOING THINGS JAVASCRIPT ISN'T GREAT AT

- While WebAssembly only has four types, the types are numeric and provide more precision with calculations than JavaScript alone
- WebAssembly can take on computation-heavy tasks with much better performance than JavaScript

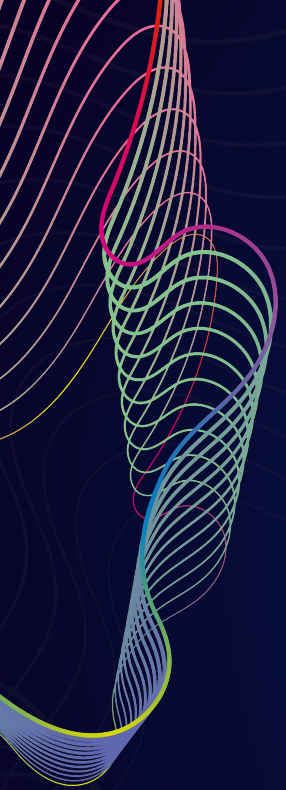


INCREASING PERFORMANCE OF WEB APPS

- WebAssembly allows code to run at the system level, which can be multitudes faster than JavaScript code at certain tasks
- The streaming compilation of WebAssembly, coupled with advancements like HTTP/2, can allow for much faster page loads/time to first operation
- Allowing WebAssembly to take on tasks that are easily compile-optimized and not done quickly by JavaScript can speed up web applications by magnitudes

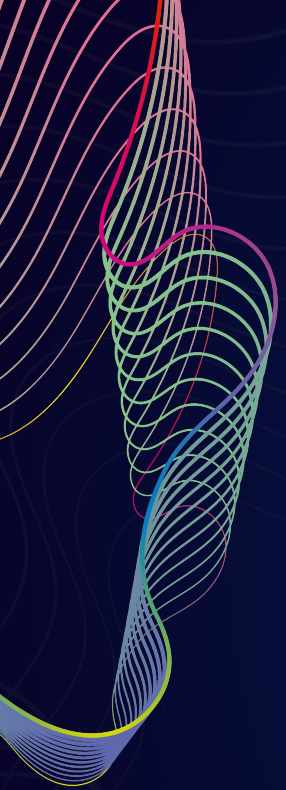


HOW TO USE WEBASSEMBLY

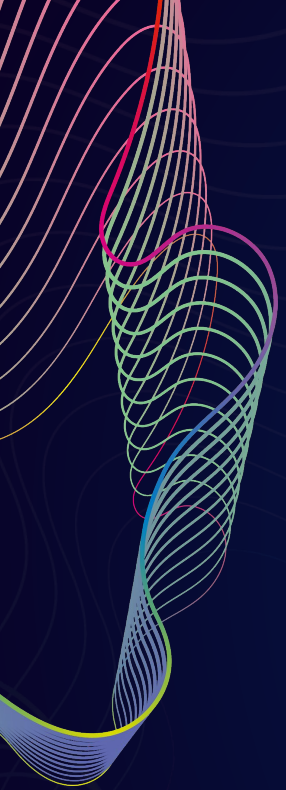


LET'S BUILD A WASM APP

- We'll write some C++ code to calculate prime numbers to a certain point
- Then we'll transpile it to WASM format with emscripten
- We'll also use emscripten to generate a boilerplate with an HTML and JS file to load it in for us
- When we load it, it'll print out the prime numbers

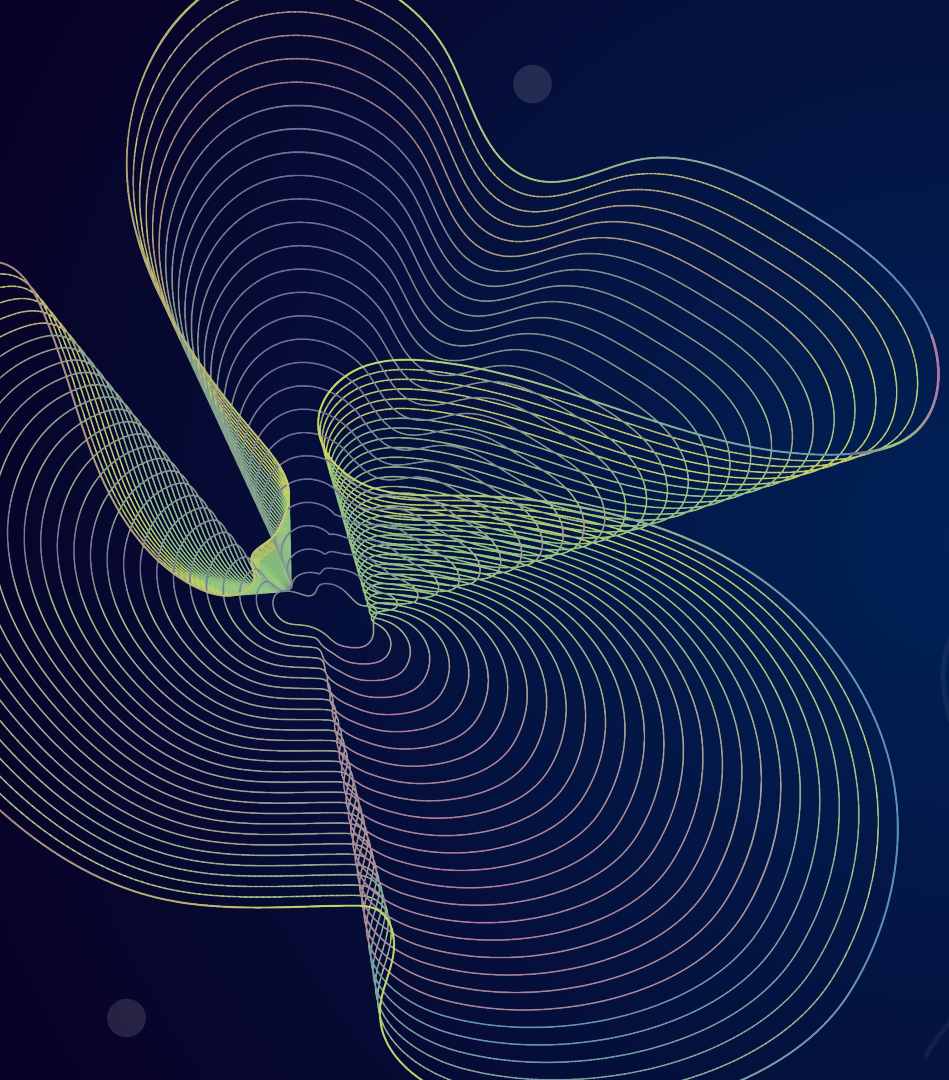


THAT'S NEAT, BUT WHAT CAN IT REALLY DO?



IMAGEMAGICK

- Graphic manipulation with ImageMagick in the browser
- Allows us to do complex image manipulation quickly
- Leverages a well-documented, battle-tested C++ library instead of having to write our own



THE FUTURE OF THE WEB WITH WEBASSEMBLY

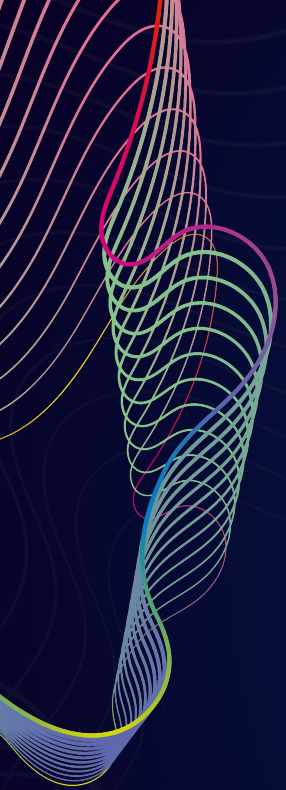
THE FUTURE IS ALREADY AVAILABLE

91.4%

Of global users according to caniuse.com

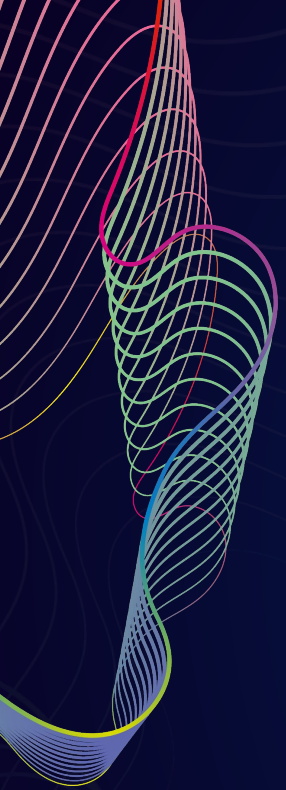
34

Languages at at least ‘usable’ level according to awesome-wasm-langs



Future plans for the WASM specification

- Garbage collection
- Reference types
- Bulk memory operations
- Multiple memories



Does WebAssembly intend to replace JavaScript?

The short answer is no; while there are developers who will try to write their entire application in another language, WebAssembly cannot do everything that JavaScript can. There will be JavaScript in the future, as far as WebAssembly stands.

THANKS FOR LISTENING

Mxs. Kassian Rosner Wren
the@nodebotanist
@nodebotanist