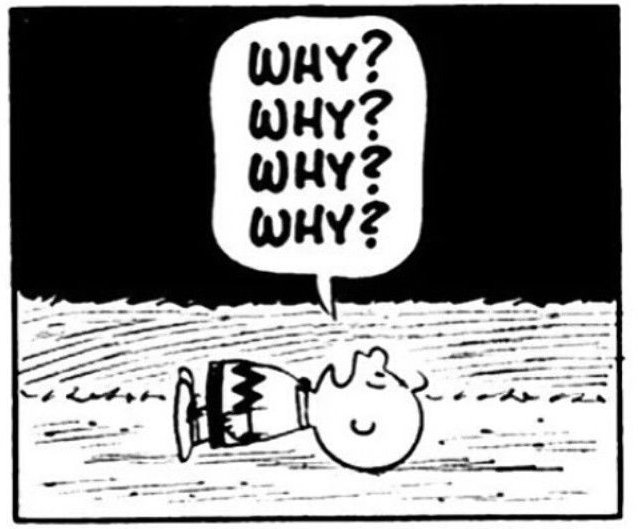
Time for Tech: WASM + Svelte

(and a little bit of Rust)

WWW

- **What?** WASM (Rust) + Component Framework (Svelte)
- **Why?**
  - Svelte by word of mouth, web frameworks feel more and more heavy
  - Understand WASM hype
  - Interest in Rust
  - Found tutorial that connects all the above 🎉
- **Where?** -> Github

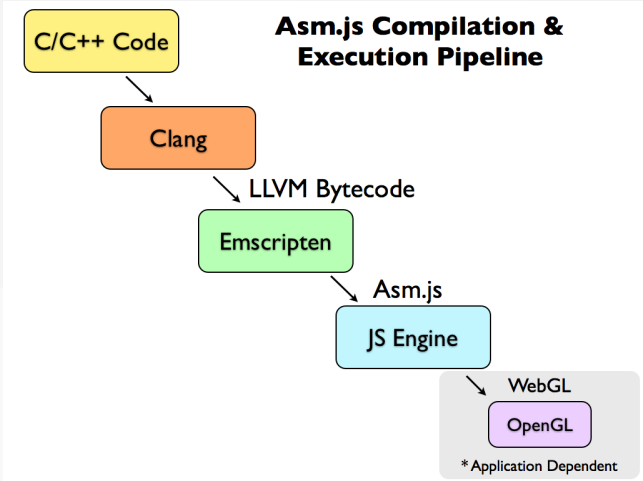


WASM Basics

- Binary format
- 40+ "source" languages
- Runtime environments ("virtual stack machines") with AOT/JIT or interpreted
- "really portable" (now srsly!)
- W3C maintained

WASM History

- asm.js (since 2013):
  - regular JS code (subset)
  - statically typed (with JS tricks) - only 4 types
  - from C via https://emscripten.org/ (+ lot of other lanugages supported)
  - approx. 2x times slower than C
  - Browsers had special optimizations for it
  - (considered) deprecated due to WASM



WASM formats

C source code and corresponding WebAssembly

C source code	WebAssembly .wat text format	WebAssembly .wasm binary format

`int factorial(int n) {`
`  if (n == 0)`
`  return 1;`
`  else`
`  return n * factorial(n-1);`
`}`
	`(func (param i64) (result i64)`
	`  local.get 0`
	`  i64.eqz`
	`  if (result i64)`
	`  i64.const 1`
	`  else`
	`  local.get 0`
	`  local.get 0`
	`  i64.const 1`
	`  i64.sub`
	`  call 0`
	`  i64.mul`
	`  end)`
		`00 61 73 6D 01 00 00 00`
		`01 06 01 60 01 7E 01 7E`
		`03 02 01 00`
		`0A 17 01`
		`15 00`
		`20 00`
		`50`
		`04 7E`
		`42 01`
		`05`
		`20 00`
		`20 00`
		`42 01`
		`7D`
		`10 00`
		`7E`
		`0B`
		`0B`

WASM Runtimes

- Browser: JS interaction via pointers & copies(!)
- WAVM: compiles to real machine code (with help of standard C toolchain LLVM)
- Wasmtime: standalone runtime; focus on security (Linux, macOS)
- Wasmer: server runtime, can convert in OS-native executables
- WasmEdge: runtime with focus on cloud (language support)

WASM Runtimes 2

- Docker tech preview (Beta, based on WasmEdge)

  > *If WASM+WASI existed in 2008, we wouldn't have needed to create Docker. (Docker Co-Founder)*

- Kubernetes (via WasmEdge)
- AWS Lambda
- ...

Use Cases (native code "everywhere")

- from asm.js: OpenGL/WebGL, SQLite, gnuplot, Unreal 3/4 engine (3 in 4 days!), ScummVM, Doom
- Simulation, image/sound/video processing, visualization, animation, compression, crypto mining
- e.g. Unity3D / Disney+ Client App for set top boxes (due to browser zoo; Rust -> WASM + own runtime)
- Edge computing: Vercel, Shopify, Cloudflare already with support
- Static UI + light-weight SSR backend in WASM (instead of client / server)

WASM Tooling

- Parallel text format (wat) for debugging and viewing in browser and tools
- WAPM: Package Registry
- WASI: Web Assembly System Interface to OS features (standard proposal), needs Wasmtime/browser polyfill; so far only usable from Rust & C/C++
- AssemblyScript: Type-script-like syntax
- WebAssembly.sh: Online shell for WASM modules

Why Rust + WASM

- Rust recommends WASM when coding for the web: https://www.rust-lang.org/what/wasm
- offers access to Web APIs (and thus DOM access)

Time for some code...

Svelte

- Svelte = slender / sleek
- *"Frameworks are not tools for organising your code, they are tools for organising your mind."*
- *"Svelte is a language."* (Rich Harris)
- *"Death to boilerplate"*

Svelte Basics

- Initiated by Rich Harris (React framework Next.JS at Vercel)
- Current version: 3
- **Compiles** to pure "highly optimized" JS
  (thus can be easily adapted to new compilation concepts / JS features)
- Everything unnecessary removed completely at compilation
- Max **compact** code (in comparison with e.g. React)
- Paradigm: **as less boilerplate as possible**
- Few "slim" concepts (e.g. reactivity on 4 short pages, tutorial 1-2 days)

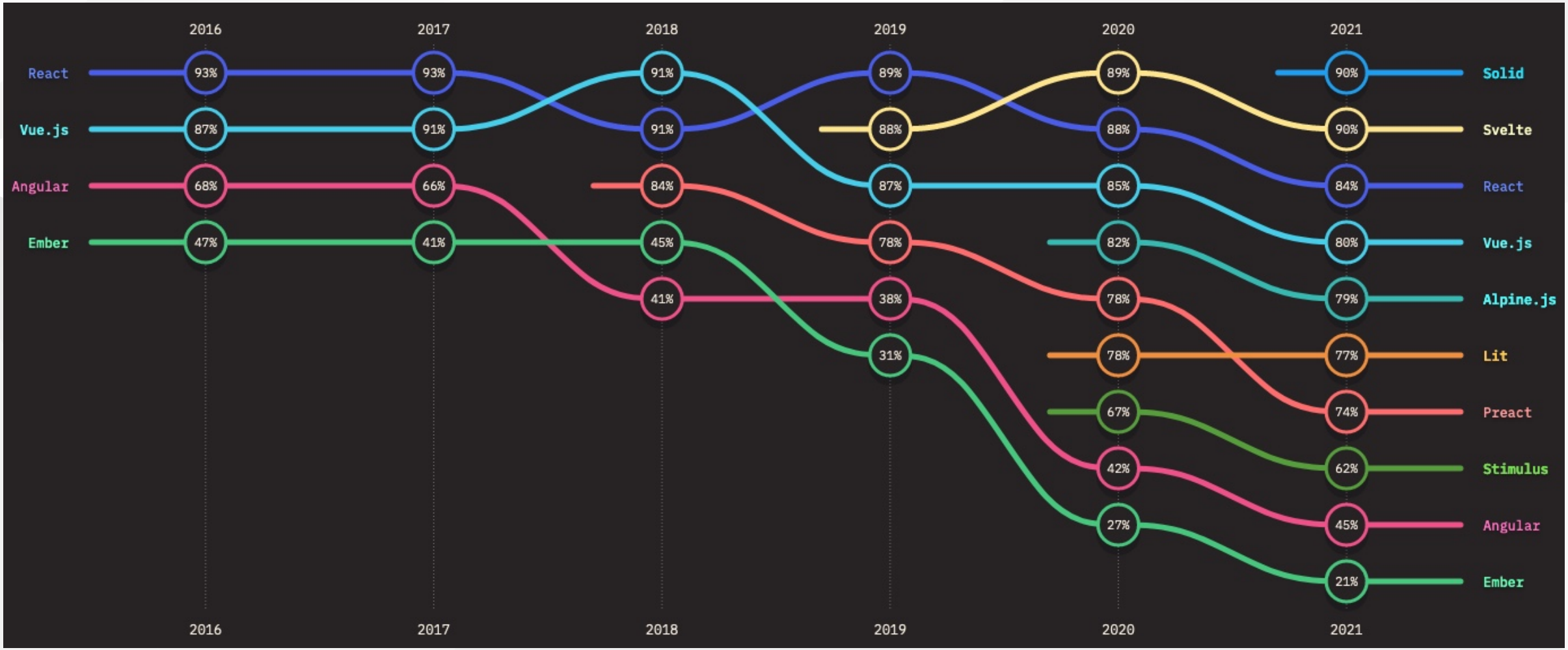
Svelte Core Concept

> *Svelte is a compiler that knows at build time how things could change in your app, rather than waiting to do the work at run time.*
> (https://svelte.dev/blog/virtual-dom-is-pure-overhead)

- **Performant**: keep track of which values are dependant upon other values instead virtual DOM comparison which can be expensive:
  - traversing DOM tree, diffing, suboptimal coding can lead to unnecessary re-calculations, ...


JS
A

Framework Retention

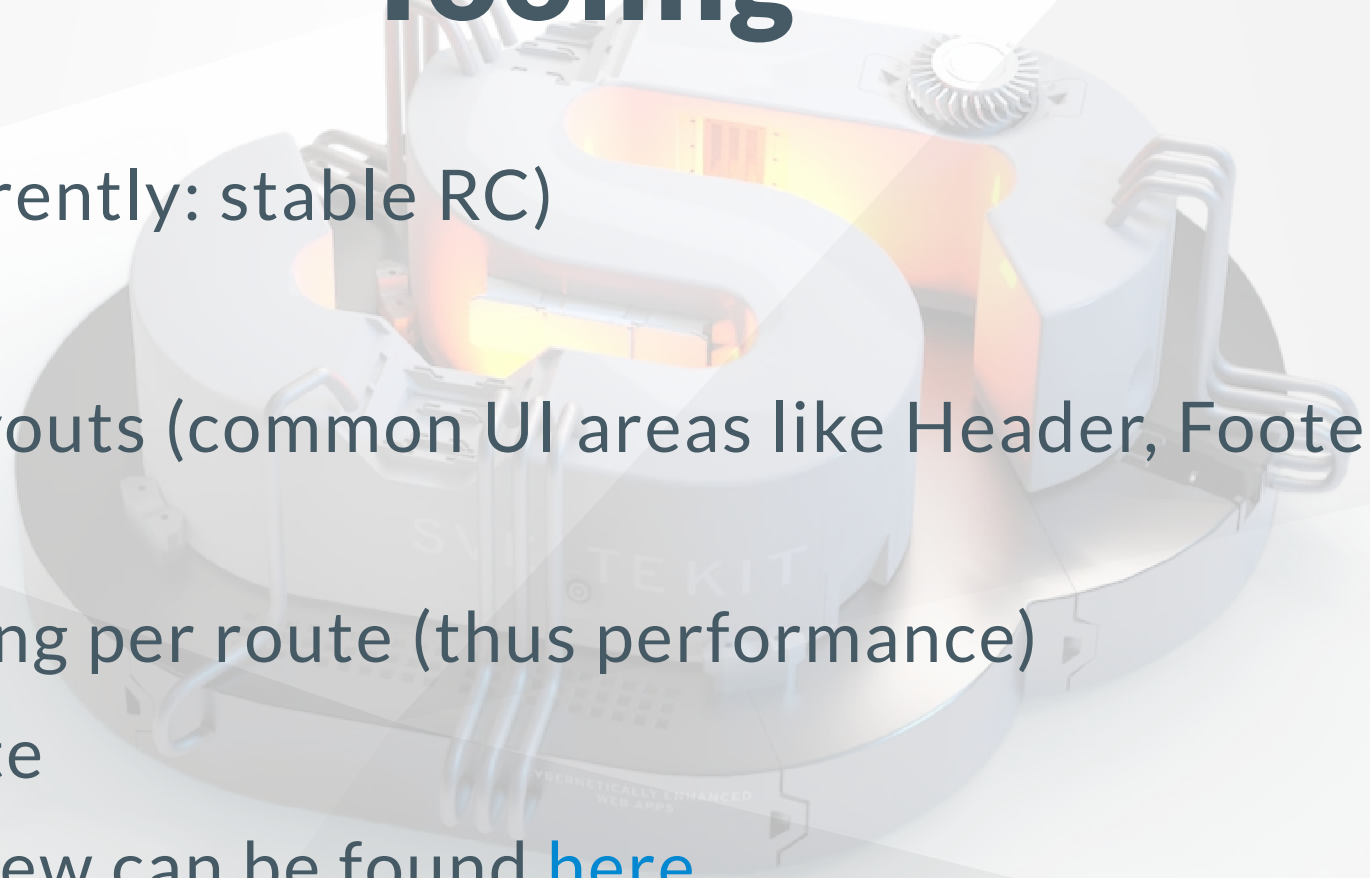


Comparison

- Angular, Vue & React = running on client side / shadow DOM
  - regarding SFC (single file components) similar to Vue
  - more compact code ("Death to boilerplate") - especially in comparison to React, but also to Vue
  - some same (e.g. Slots, onMount/tick) as Vue
  - Example: decision for Vue2 -> Svelte (instead Vue3)
- SolidJS: similar core concept; more boilerplate; aims at React
- RealWorld example: https://github.com/sveltejs/realworld
  - BTW: RealWorld CodeBase is awesome!

Time for some code...

Tooling



- **SvelteKit** (currently: stable RC)
  - Scaffolding
  - Routing, Layouts (common UI areas like Header, Footer), SSR (default)
  - Code-splitting per route (thus performance)
  - based on Vite
  - Good overview can be found here
- Svelte Native - based on NativeScript

UI Components Libs

- Svelte Material UI
  - Components can be used stand-alone (besides theming)
- Carbon Components
- Svelte Frameworks and Shocases

Where to go afterwards

- 10 reasons for Svelte
- Svelte Interactive Tutorial incl direct JS/CSS output
- Svelte Kit Tutorial: for Svelte and SvelteKit
- Svelte Examples
- Online interactive editors:
  - REPL
  - StackBlitz