WEBASSEMBLY

&

blazor

Look ma, no JavaScript… it's Blazor!

Some bits about me..

zühlke
empowering ideas

Jose Luis Latorre Millas
Lead Software Architect
Jose.latorremillas@zuehlke.com / joslat@gmail.com
Linkedin: www.linkedin.com/in/joslat/
Twitter: @joslat

Blazor..

- SPA framework.
- .NET Core 3.0 in the browser.
- Yes, this means No JavaScript.
- Base technology is WebAssembly.
- Near native speeds.
- Open Source.
- Looks like Microsoft has learned from previous experiences..

WEBASSEMBLY

Webassembly.org

"WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server applications."

Short overview

- A low-level binary format, assembly like, language.
- Performance close to native.
- Compatible with 88.8% of browsers.
- A Standard.

«WebAssembly, or wasm, is the most significant new technology to come to the web platform in a decade."

"it's the ultimate culmination of web development capabilities"

"WebAssembly is a game-changing technology"

Source – Mozilla (and many more..)
https://research.mozilla.org/webassembly/

The story of WebAssembly

Asm.js (2014)

Ext. features 2018

asm.js

Working Draft — 18 August 20

Latest version:

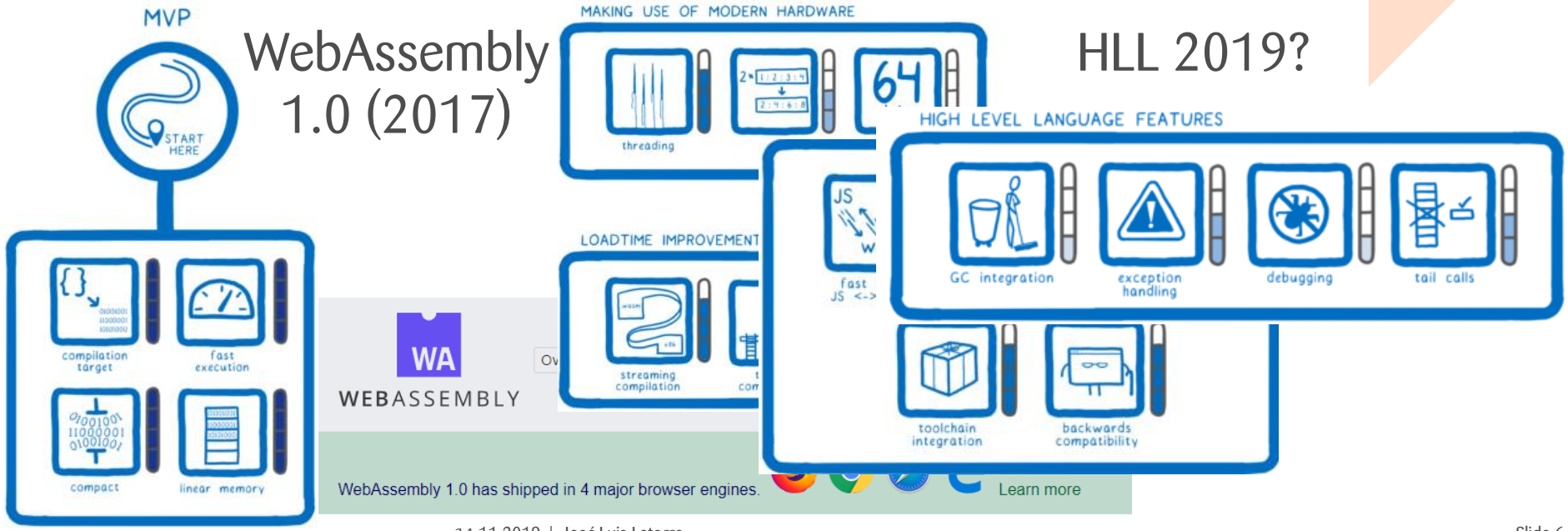
http://asmjs.org/spec/latest/

Editors:

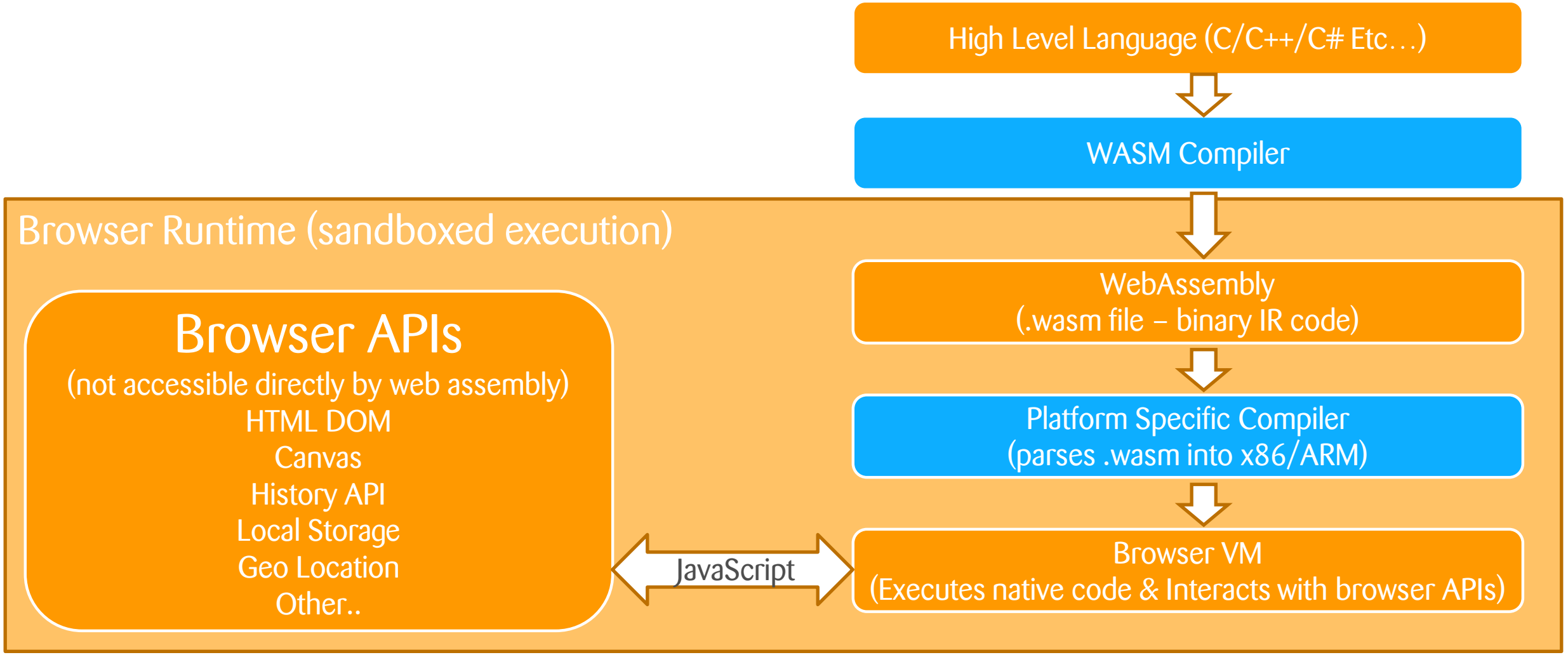
David Herman, Mozilla, <dherman@mozilla.cc
Luke Wagner, Mozilla, <luke@mozilla.com>
Alon Zakai, Mozilla, <azakai@mozilla.com>

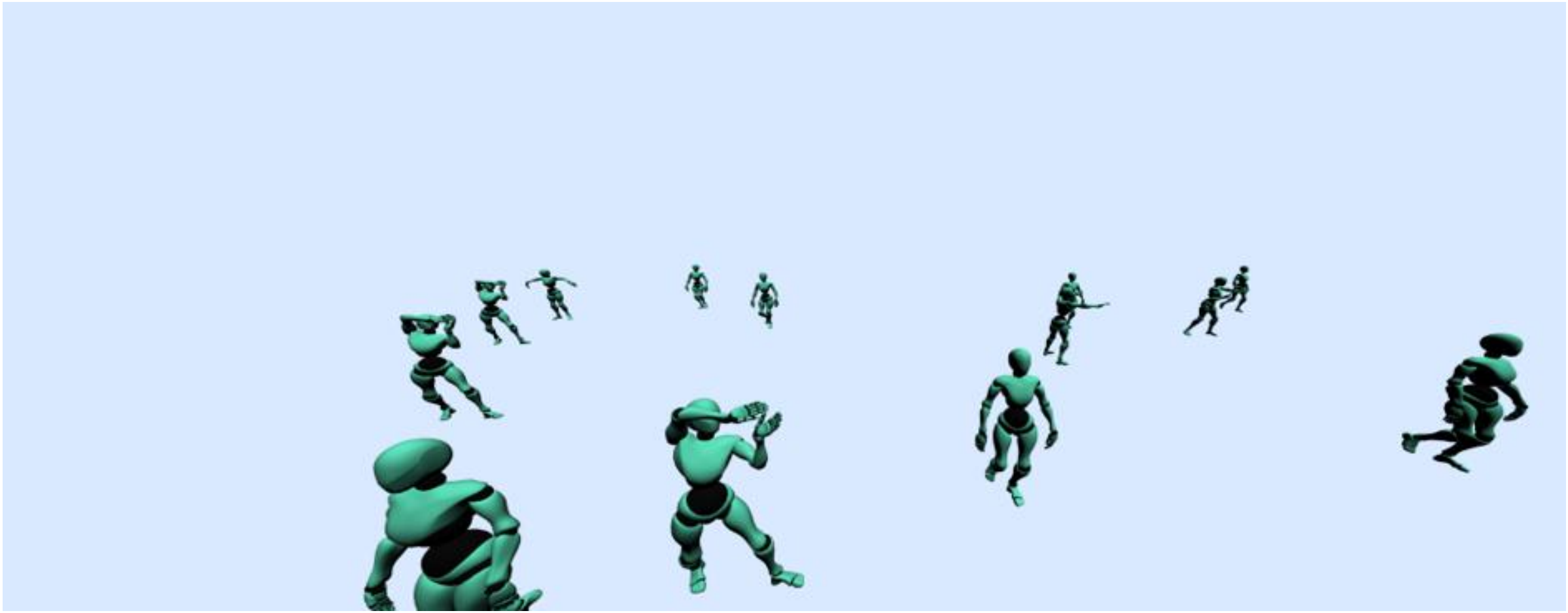
Abstract

This specification defines **asm.js**, a strict subset of
target language for compilers. This sublanguage eff
memory-unsafe languages like C or C++. A combin
JavaScript engines to employ an ahead-of-time (AO
code.



WebAssembly simplified



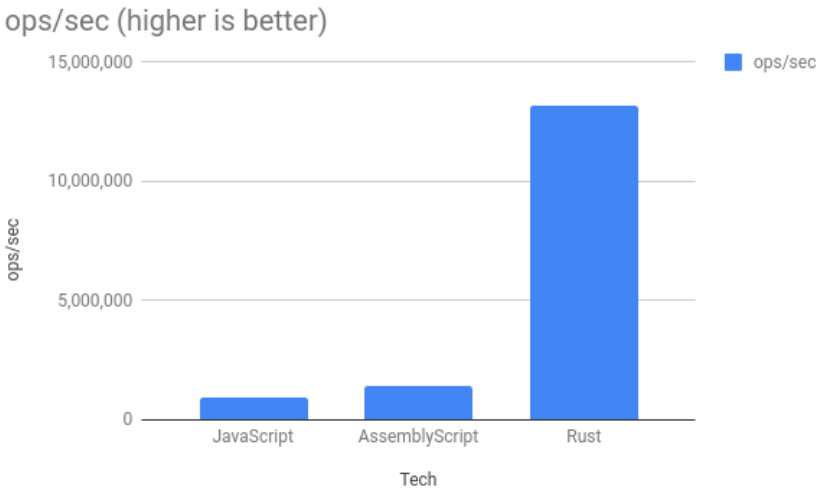


http://aws-website-webassemblyskeletalanimation-ffaza.s3-website-us-east-1.amazonaws.com/

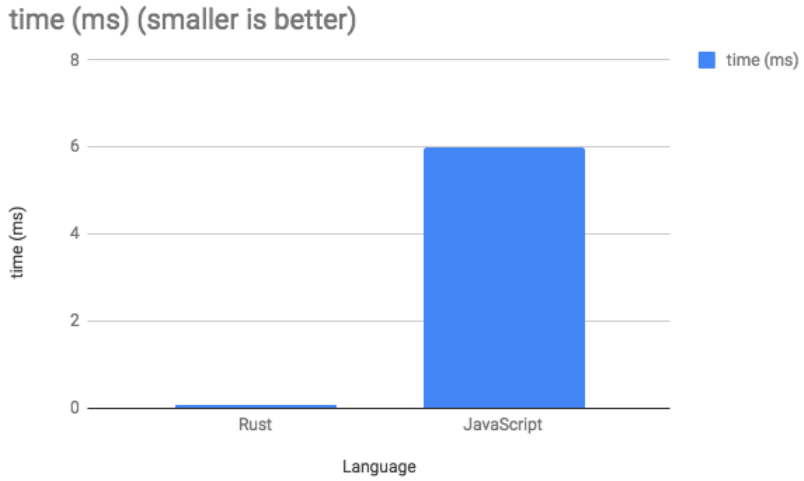
DEMO

Doom3 (nsfw) - https://wasm.continuation-labs.com/d3demo/

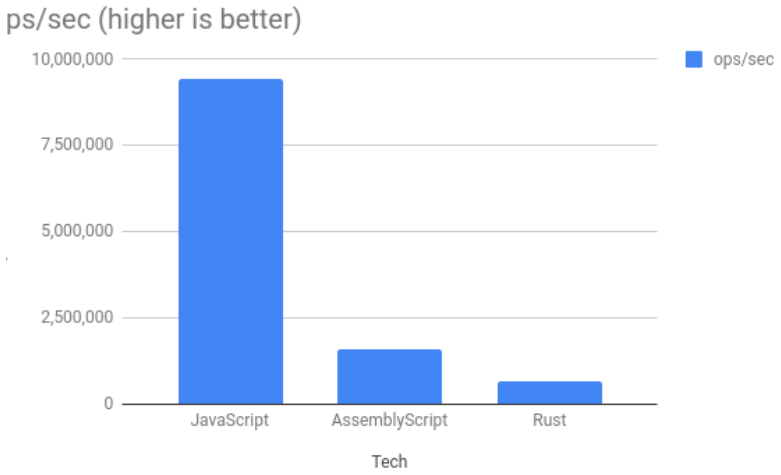
Benchmarks?



addition



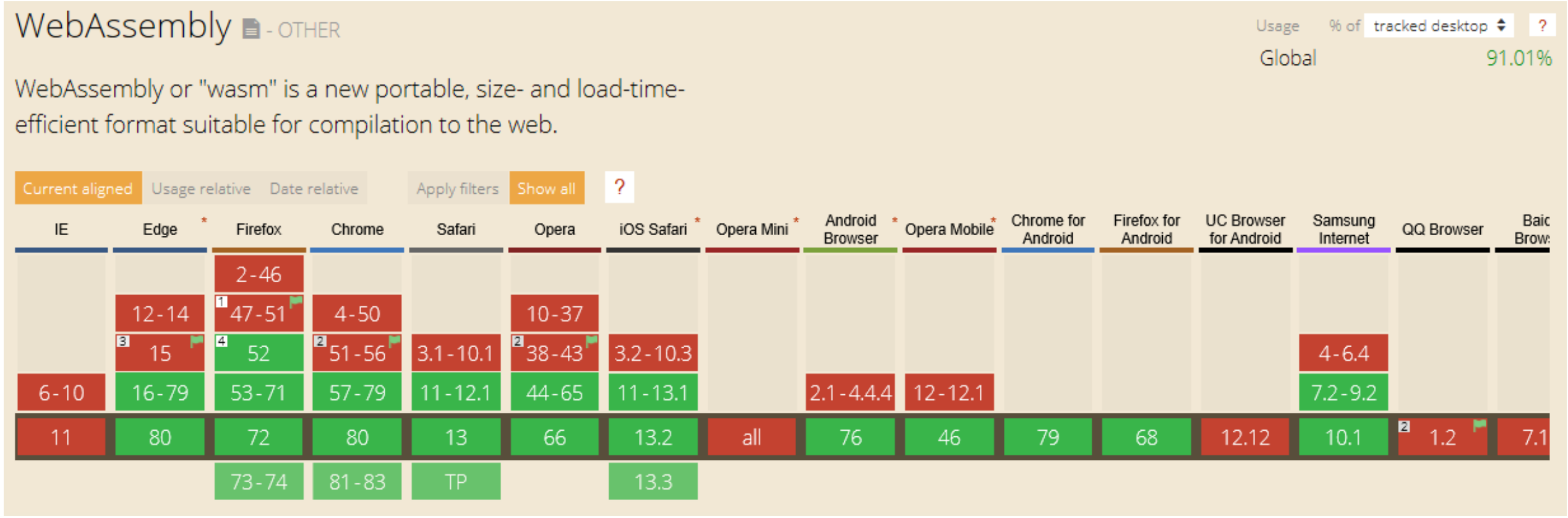
Numeric Array handling



String manipulation

Source: https://blog.sqreen.com/webassembly-performance/

Adoption



Source: https://caniuse.com/#feat=wasm

& blazor

Why.NET for browser apps?

Modern languages (C#)

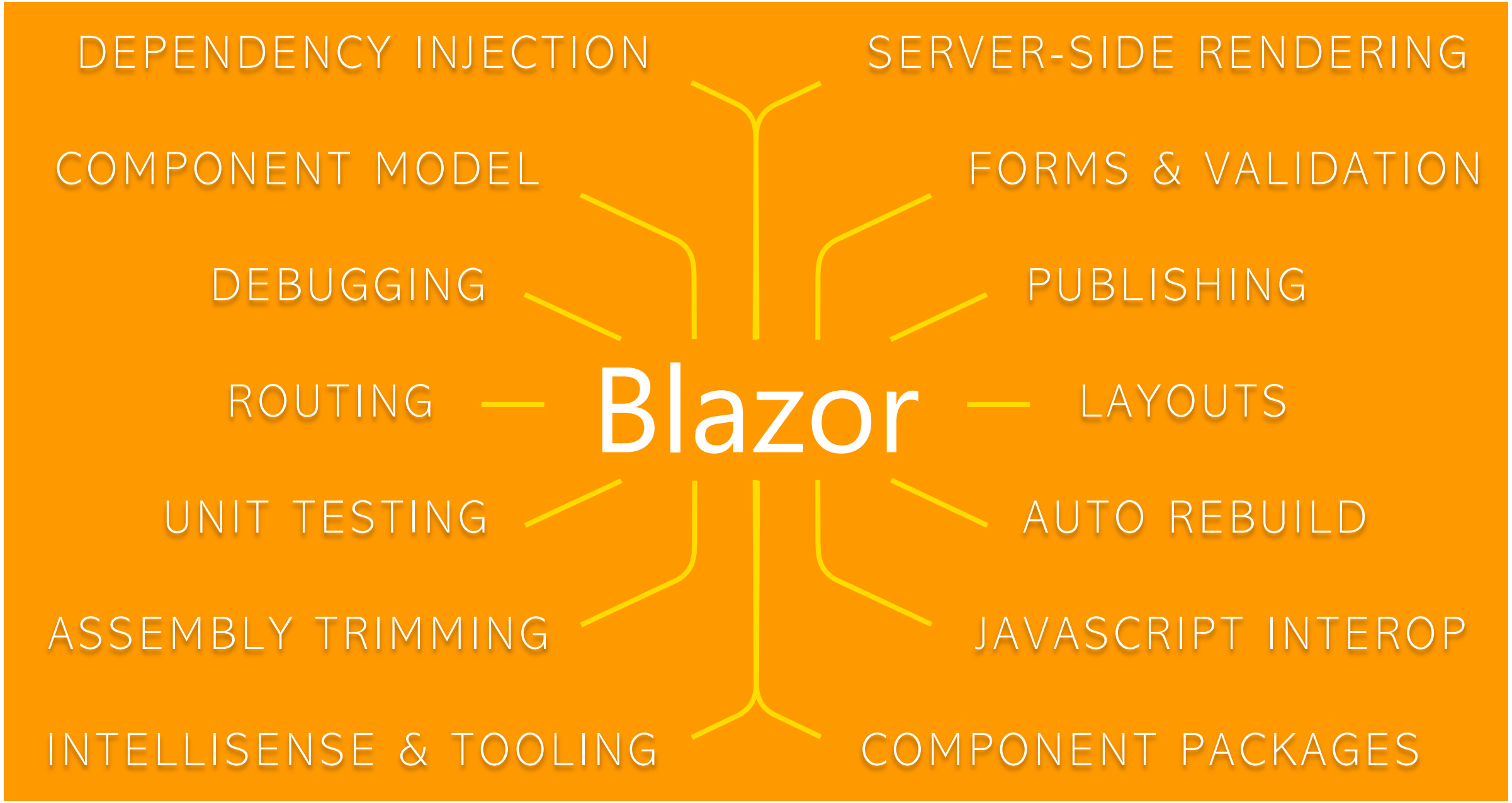
.NET ecosystem

Performance

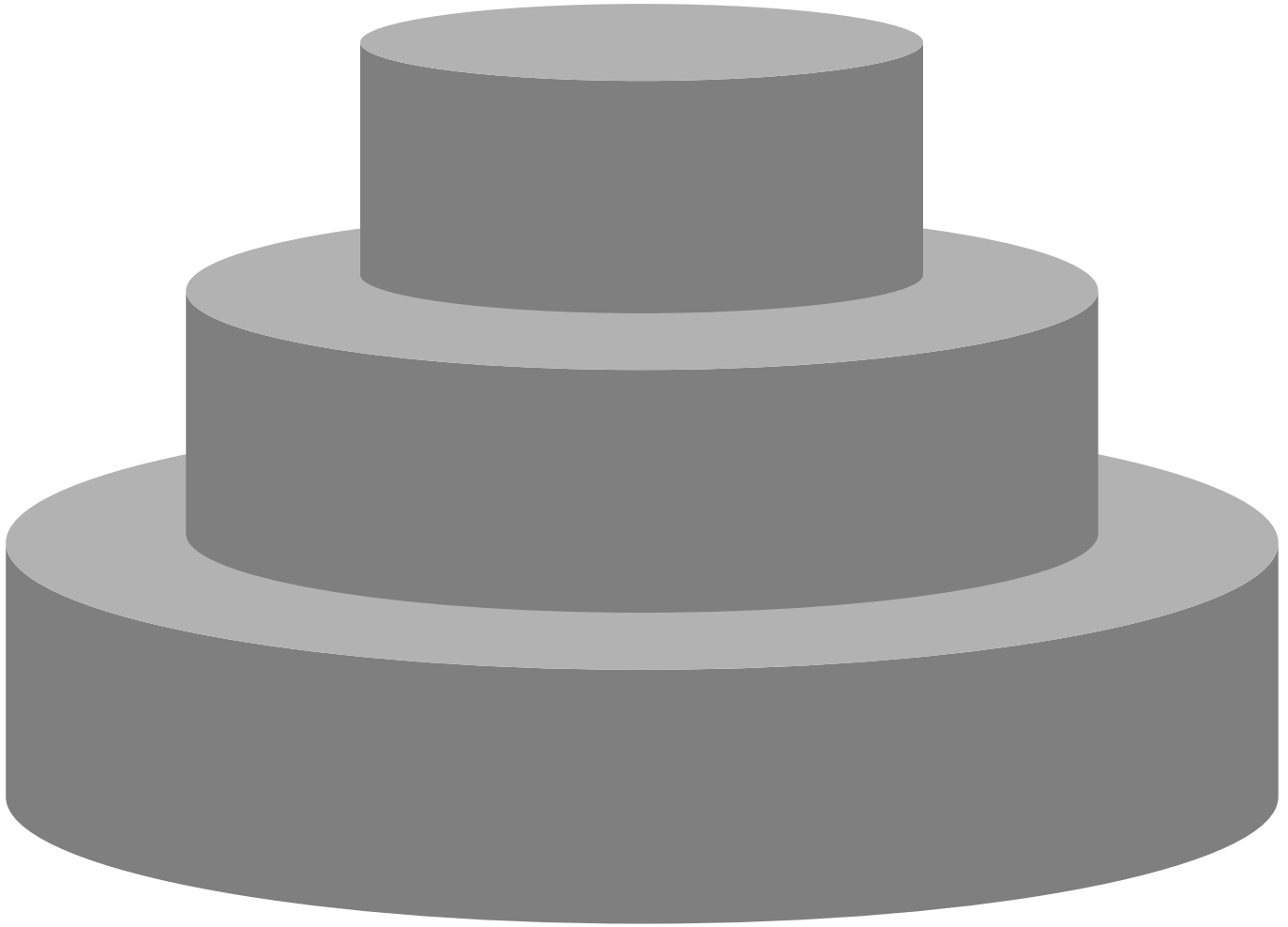
Full-stack

Tools

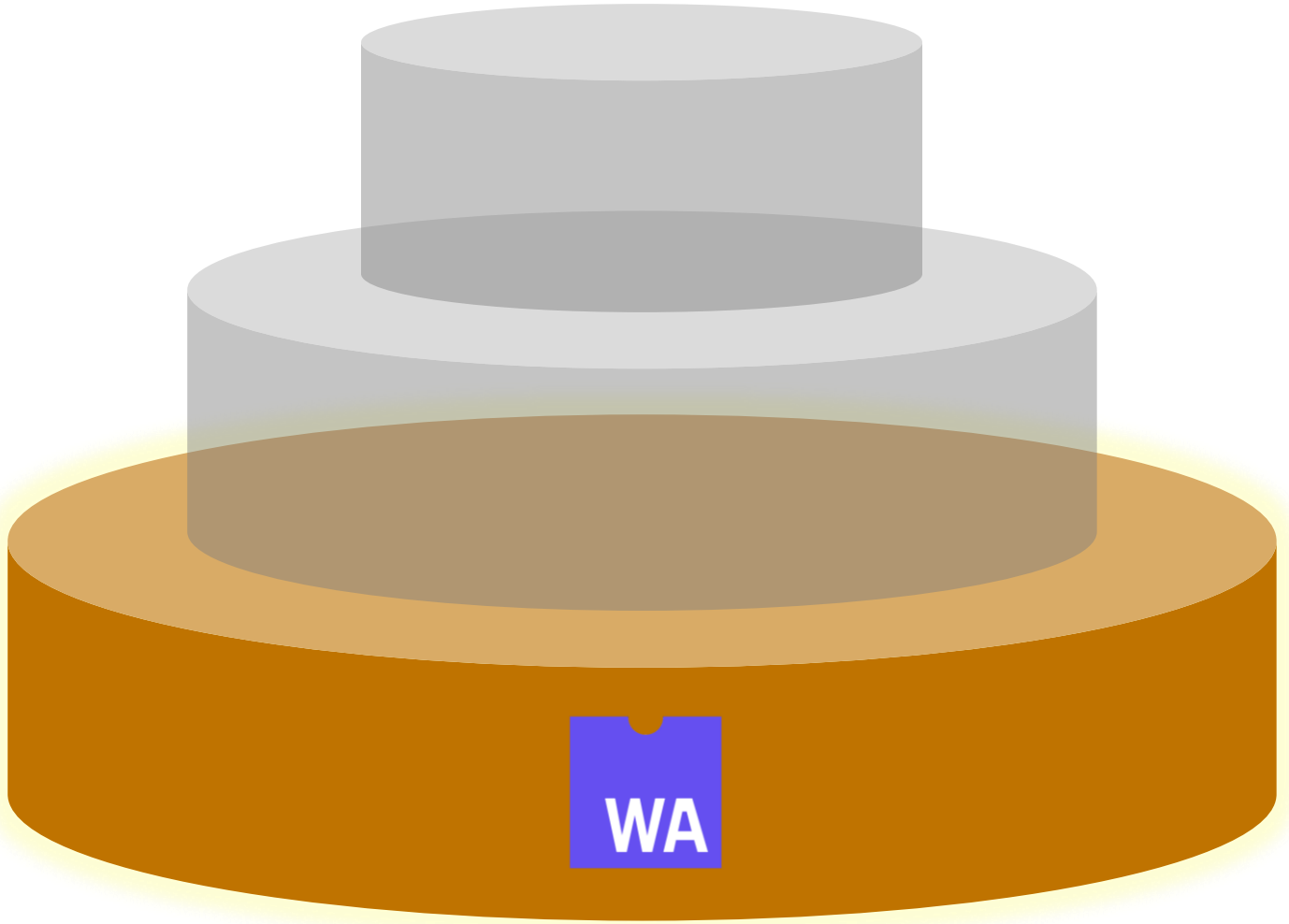
Stable & mature


DEPENDENCY INJECTION
SERVER-SIDE RENDERING
COMPONENT MODEL
FORMS & VALIDATION
DEBUGGING
PUBLISHING
ROUTING
Blazor
LAYOUTS
UNIT TESTING
AUTO REBUILD
ASSEMBLY TRIMMING
JAVASCRIPT INTEROP
INTELLISENSE & TOOLING
COMPONENT PACKAGES

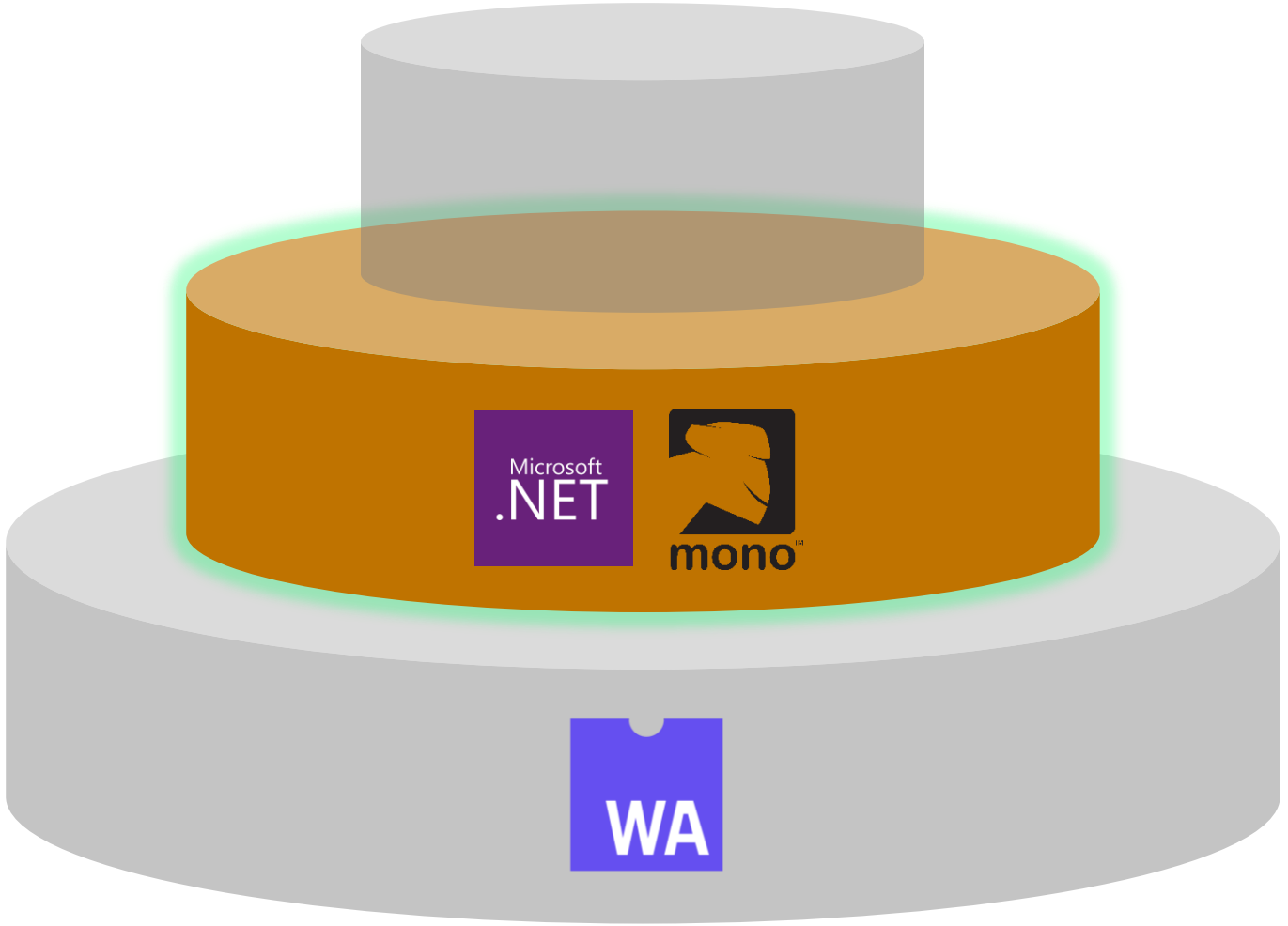
Ultra simple Architecture



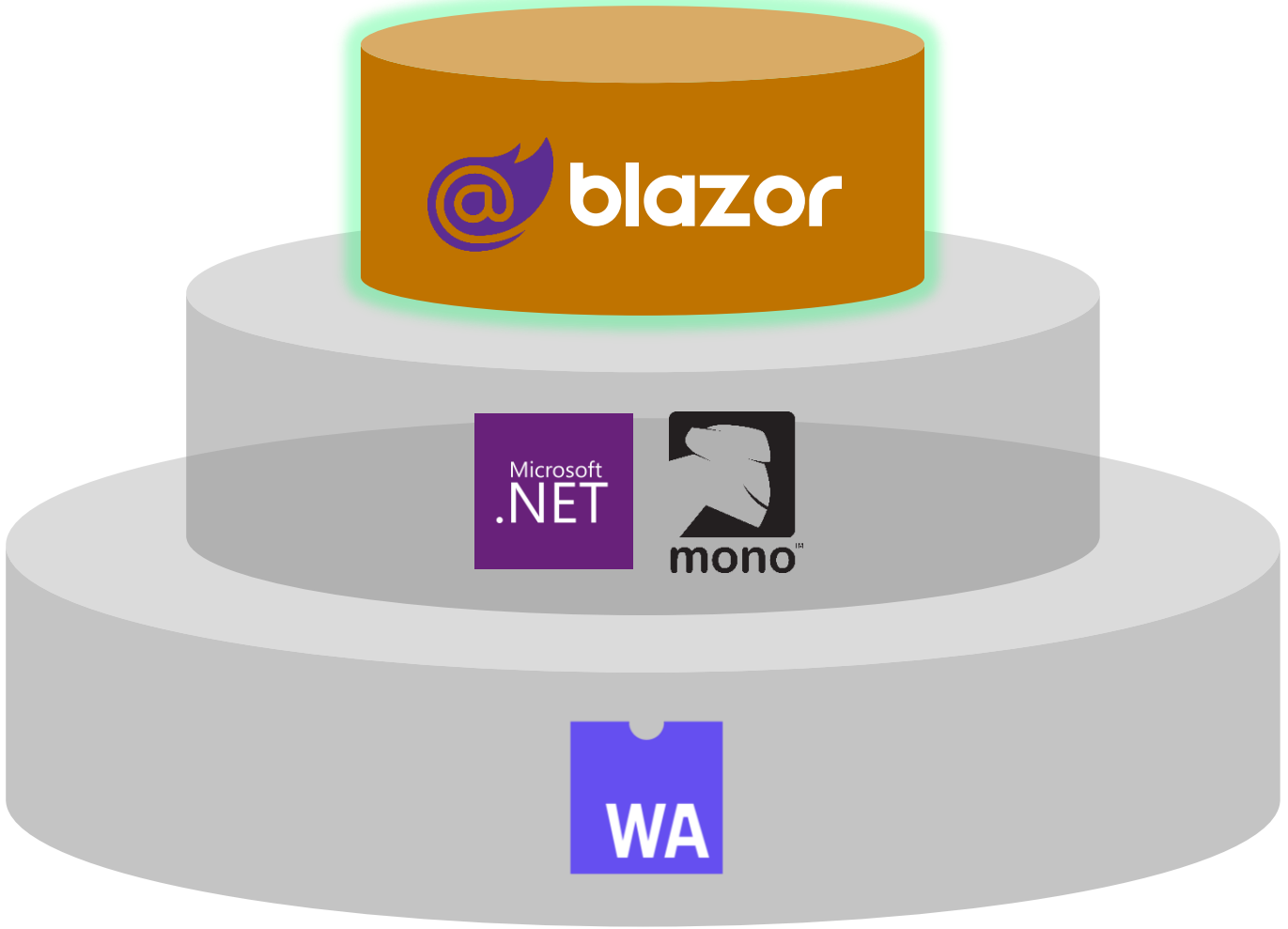
Ultra simple Architecture



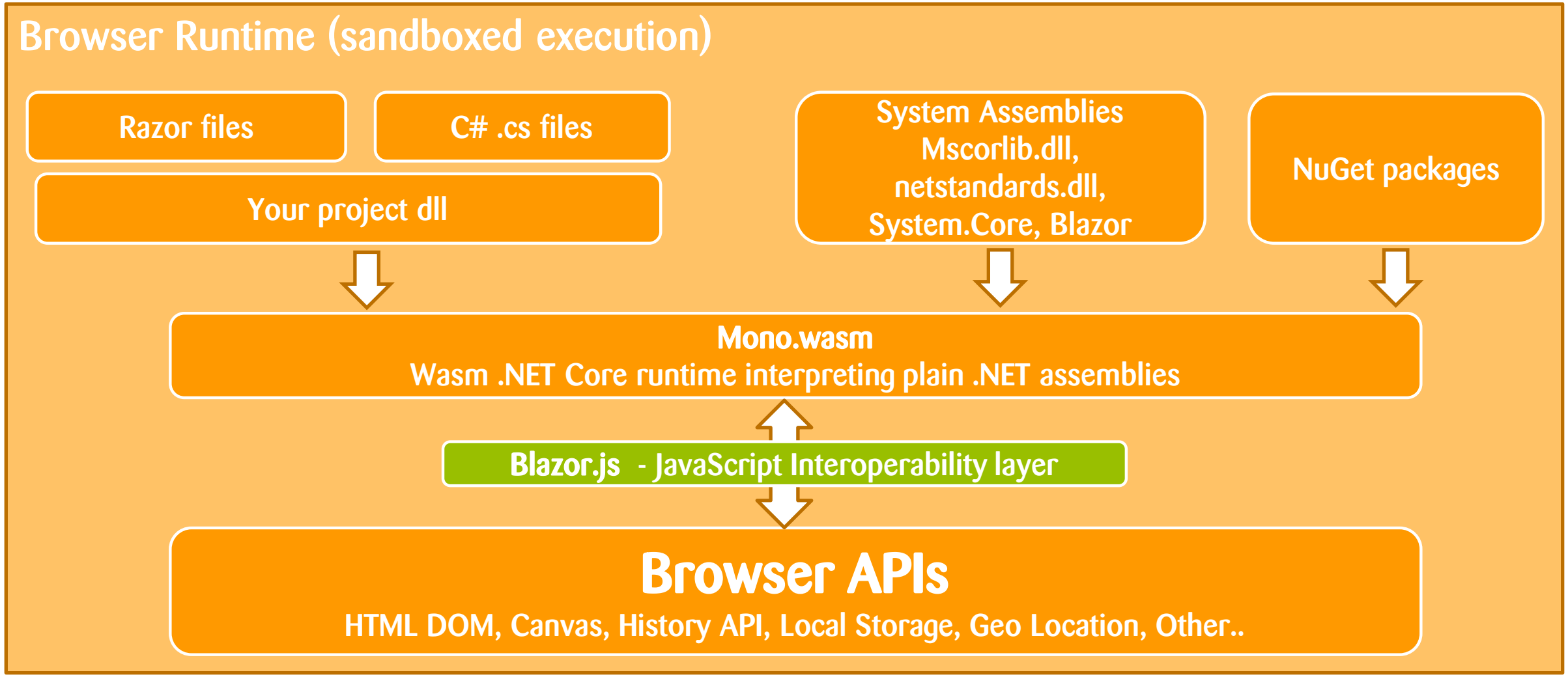
Ultra simple Architecture



Ultra simple Architecture



Blazor simplified



1. Client-side

2. Server-side

Client-side
(Blazor WebAssembly)

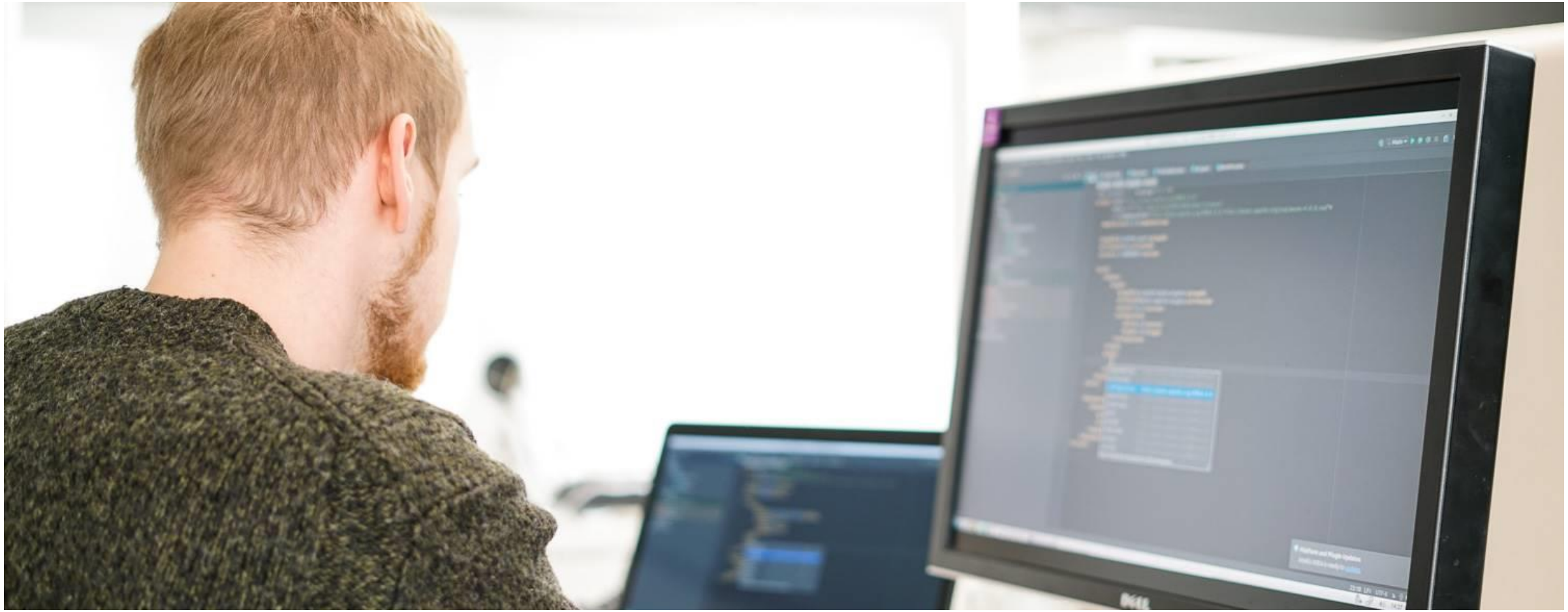
Preview now
Release target: May 2020

Great for:
- Offline support / PWA
- Zero latency UI
- Static-files hosting
- De-loading the server

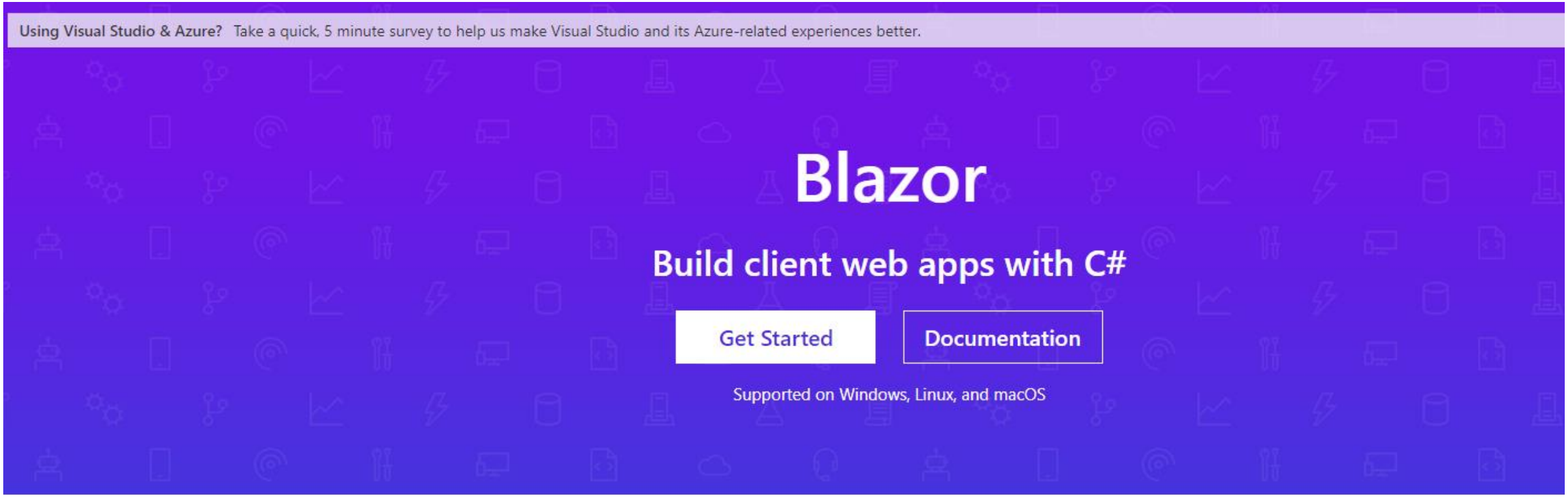
Server-side
(Blazor Server)

Shipped
Released in .NET Core 3.0

Great for:
- Thin client
- Full runtime
- All code on server
- More security



blazor
DEMO(s)



Getting started… With the latest preview go to: “http://blazor.net”

Go to blazor.net

And follow the yellow path…

- Hint: It will bring you to..
  - https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor
- Click on “Get Started” and now, follow the steps..
- Tips
  - Install and use Visual Studio 2019 preview (recommended)
  - Otherwise might have issues…
  - Read the announcement in full ☺
    - https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-preview-1-release-now-available/

Server-side Blazor
(.NET Core)

Web app
Every interaction handled on server
Prerendered HTML (optional)

Client-side Blazor
(Mono.wasm)

Web app with client-side execution
Loaded from web server
Can work offline via Service Worker

OS-installed Blazor PWA
(Mono.wasm)

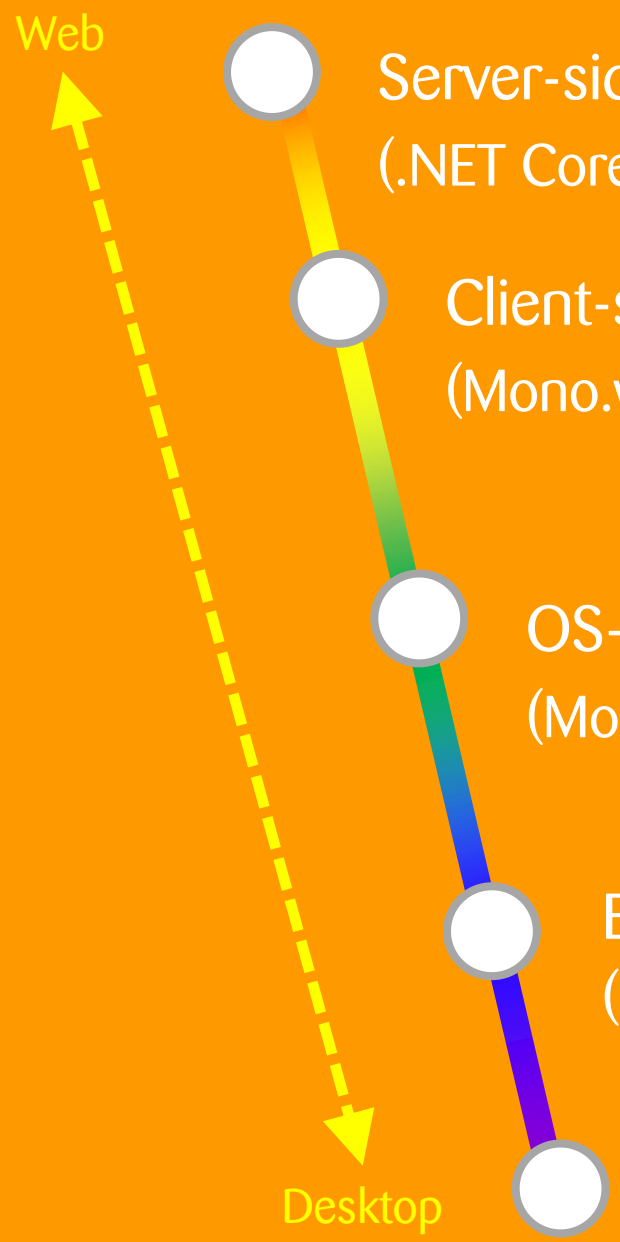
Appears as native app (own window)
Works offline or online

Electron + Blazor
(.NET Core)

Appears as native app (own window)
Works offline or online

Native UI + Blazor
(.NET Core)

Same programming model, but
rendering non-HTML UI



Server-side Blazor
(.NET Core)

Web app
Every interaction handled on server
Prerendered HTML (optional)

Client-side Blazor
(Mono.wasm)

Web app with client-side execution
Loaded from web server
Can work offline via Service Worker

OS-installed Blazor PWA
(Mono.wasm)

Appears as native app (own window)
Works offline or online

Electron + Blazor
(.NET Core)

Appears as native app (own window)
Works offline or online

Native UI + Blazor
(.NET Core)

Same programming model, but
rendering non-HTML UI

Web

Desktop

Server-side Blazo
(.NET Core)

Client-side Bl
(Mono.wasm)

OS-installe
(Mono.was

Electro
(.NET C

Nativ
(.NET Core)

tion handled on server
HTML (optional)

client-side execution
web server
ine via Service Worker

ative app (own window)
or online

ative app (own window)
or online

mming model, but
rendering non-HTML UI

Client-side
(Blazor WebAssembly)

Preview now
Release target: May 2020

Server-side
(Blazor Server)

Shipped
Part of .NET Core 3.0

Blazor + Electron
(Desktop app with web rendering)

Uncommitted. Samples:
aka.ms/blazor-electron
aka.ms/webwindow

Blazor + Native UI
(Mobile/desktop with native OS UI)

Uncommitted. Samples:
aka.ms/mobile-blazor-bindings

Conclusion

- 3 main takeaways:
  1. Full .NET Standard 2.0
  2. No JavaScript
  3. Server-side Shipped & Client-side coming May 2020

Questions?

Thanks for having me ☺ .NET User Group Bern!

zühlke
empowering ideas

Jose Luis Latorre Millas
Lead Software Architect
Jose.latorremillas@zuehlke.com / joslat@gmail.com
Linkedin: www.linkedin.com/in/joslat/
Twitter: @joslat