



Blazor

C# running in the browser via
WebAssembly

Scott Sauber

Audience

- Mostly targeted for .NET developers
- JS Developers interested in WebAssembly

Agenda

- What is WebAssembly?
- What is Blazor?
- How does Blazor work?
- Demos
- Questions

Purpose

- Differentiate what is Blazor vs WebAssembly
- Get excited for the future

Who am I?

- Software Consultant at Lean TECHniques
- .NET/JS Developer
- React fanboy
- Actually enjoys JavaScript
- Blog primarily on ASP.NET Core on scottsauber.com
- Author of [Blazor Snippets for VS Code](#)

Current State of the SPA Front End

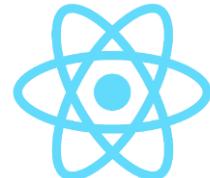
Pick a Language:



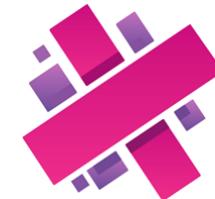
flow



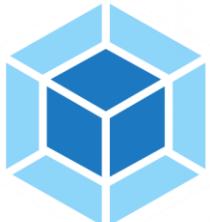
Pick a Framework:



ember



Pick your tools:



BABEL

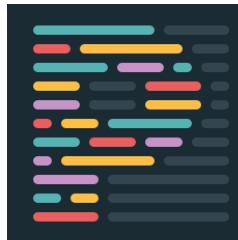


ESLint

Common



Google



So. Many. Decisions.

Editor

- Which one?
- Which plugins?
- Use built in terminal?
- Editor config

Module format

- ES6 Modules, CommonJS...

HTML generation

- Minify?
- Use plugin?
- Inject prod only concerns?
- Templating language?

Transpiling

- Native ES or diff language?
- Use experimental features?
- Which plugins?
- Production vs dev config

Bundler

- Webpack, Browserify, Rollup...

Linting

- Which linter?
- Enable which rules?
- Warning or error?
- Which plugins?
- Use a preset?

Testing

- Framework?
- Assertion Library?
- Helpers?
- Test file location?
- File naming?
- What environment?
- Mocking?
- Code Coverage
- Continuous Integration

Project structure

- By file type or feature?
- Centralize API?
- Allow Inline JS?
- Extract to POJOs?

HTTP

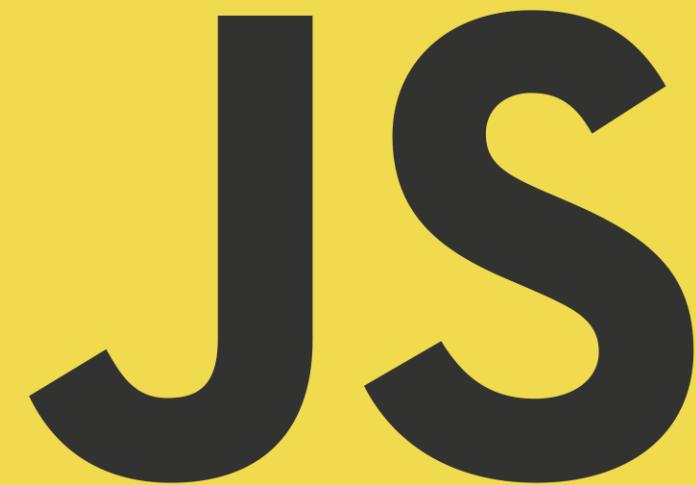
- Library
- Mock schema format
- Mock data generation
- Mock server

Production build

- Minification
- Sourcemaps
- Bundle splitting
- Cache busting
- Error logging



At the end of the day....

A large, bold, black "JS" logo is centered on a solid yellow rectangular background. The letters are stylized with thick, rounded strokes. The "J" has a vertical stem and a curved hook, while the "S" is a continuous curve with a small loop at the top. The yellow background is a solid color with no texture or gradients.

Problems

- Whole host of people don't like JS
 - Dynamically typed
 - Less integration, more stitching
 - Browser support
 - Moves too fast, lots of choice, intimidating
 - `node_modules`
- SPA's are more expensive to maintain
 - Front-end + Back-end team
 - Training up full stack to be great at both (very difficult)
- When using a different language than JS on the backend...
 - Duplicate Business Logic (like validation) or just have server
 - No IDE/compiler help between backend models + front end making AJAX calls
 - Unless bringing in yet another tool

What is Web Assembly (WASM)?

- WebAssembly (WASM) is a low-level binary format language that can be run in modern web browsers that runs at near-native speeds.
- Compilation Target for other languages
- Browser standard
- No more JS monopoly



WEBASSEMBLY

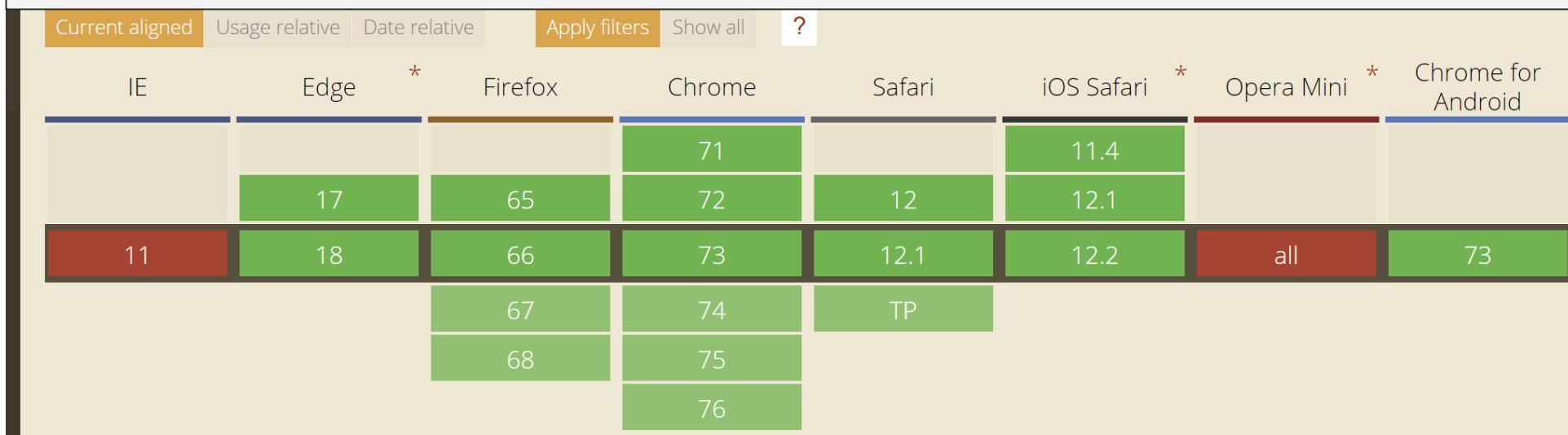
Is WASM Ready?

Can I use webassembly ?  Set

2 results found

WebAssembly  - OTHER

And you can polyfill WASM with asm.js!



What is Blazor?

- Blazor is a .NET SPA framework ([currently in Preview](#)) maintained by Microsoft using C# and HTML that runs *in the browser* via WebAssembly....



Wait a second....



It's a Standard, not a plugin!

What is Blazor?

- Blazor is a .NET SPA framework ([currently in Preview](#)) maintained by Microsoft using C# and HTML that runs *in the browser* via WebAssembly....
- Uses Razor syntax
 - Browser + L + Razor = Blazor
- Uses component-based architecture
- Runs on top of Mono
 - Blazor == UI Framework == MVC or Web Forms
 - Mono == Runtime == .NET Framework or .NET Core
- Development led by Steve Sanderson, of KnockoutJS fame



Was Experimental – Now Committed & Preview

- “... With this newest Blazor release we’re pleased to announce that **Blazor is now in official preview!** Blazor is no longer experimental and we are committing to ship it as a supported web UI framework including support for running client-side in the browser on WebAssembly...”

[.NET Core 3.0 Preview 4 announcement](#)

So I can write C# in the Browser!?!

- Blazor is .NET Standard 2 compliant
- However, not all .NET Standard 2 API's are implemented running in browser make sense
 - Examples
 - System.Net.Mail
 - System.IO
 - These throw Platform Not Supported exceptions
- But a lot do make sense
 - HttpClient => AJAX



Blazor Provides Calling C# from JS + vice versa

- C# Wrappers on top of JS API's
 - LocalStorage
 - PaymentRequest
 - Or any npm library
- C# maps to JS pretty well
 - async/await
 - Task => Promise

Why would you be interested in this?

- C# is a fantastic language
 - ...not that JavaScript isn't...but statically typed languages are winning (see: TS, Flow, Reason, etc.)
 - [46% of respondents to npm survey are using TypeScript](#)
- ASP.NET Core performance
 - [#7 on TechEmpower](#)
 - 8x faster than Node, 1.5x faster than Netty (Java), 47x faster than Django (Python), 7x faster than Kotlin, etc.
- Share logic with existing .NET backend
 - Validation logic
 - Models from Server when retrieve from the Client
- Consolidate frontend and backend teams under one language

Demo #1

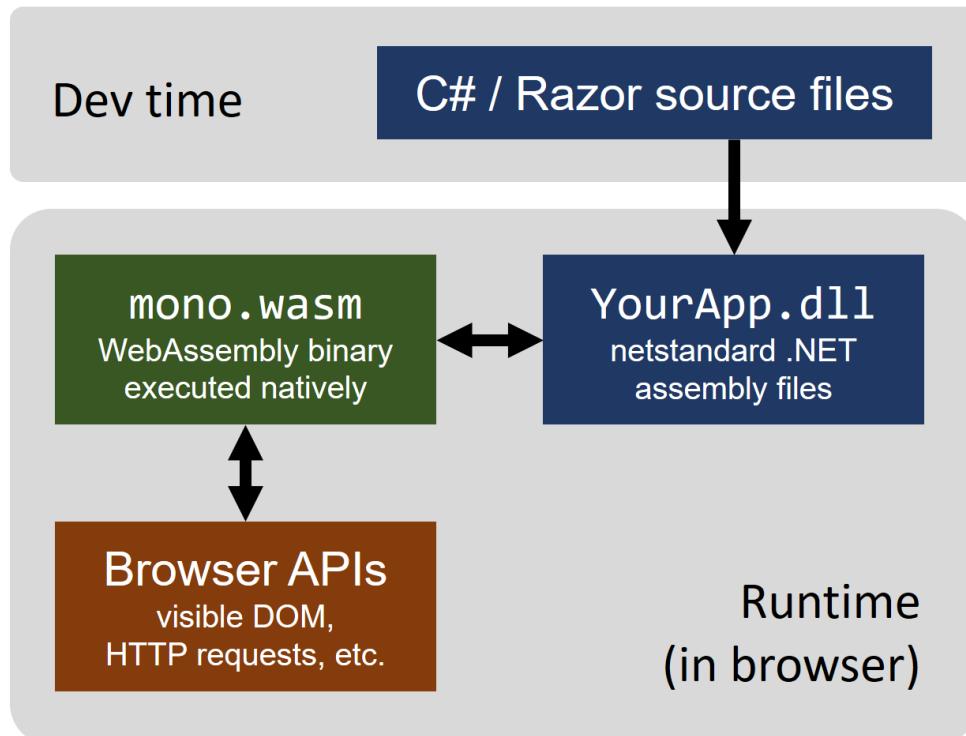
- Hello World on Blazor
- Component Architecture
- Dependency Injection
- Sharing logic

Rapid Fire Questions

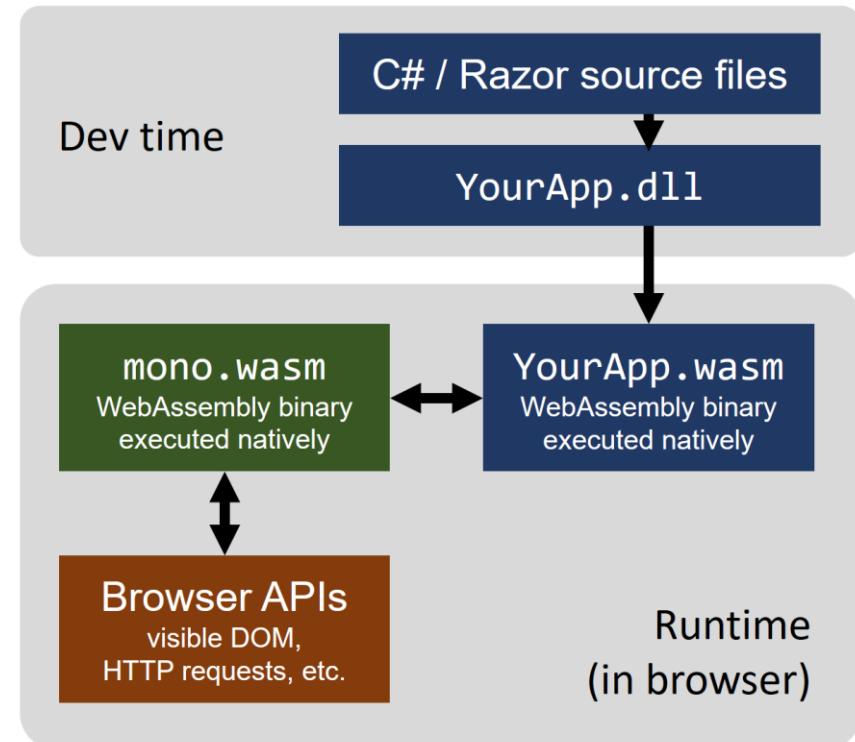
- How big is it?
 - 2.4MB
 - Very little work done thus far to optimize
- Do WASM files cache like JS and CSS files?
 - Yes
- How does it work under the hood?

How does Blazor work?

Today



Future



[Source](#)

Why Mono? Why not .NET Core?

- Already Client-side-focused
 - Xamarin, Unity, etc.
 - .NET Core is Server-side-focused
- Already developed for unique platforms (iOS, watchOS, PS4, etc.)
- Already had linker (DLL trimmer/tree shaker) for Xamarin
- They got it working first
- Long term they want to consolidate on .NET Core

Demo #2

- LocalStorage C# Wrapper
- Code: <https://github.com/scottsauber/BlazorToDoMVC>

What else can we do?

- Blazor's component model is de-coupled from the Browser
- ...SO...

What else can we do?

- Blazor on Electron
 - Cross-platform desktop framework. Write once, run anywhere.
 - [Proof of Concept Running on .NET Core](#)
- Why?
 - Faster Code Execution
 - Full Debugger in VS
 - .NET Core instead of Mono
 - Access to Desktop API's

Demo #3

- Blazor on Electron
 - Electron.App
- Code:
<https://github.com/SteveSandersonMS/BlazorElectronExperiment.Sample>

What else can we do?

- Blazor on the Server
 - Feels client-side
 - Changes streamed via WebSocket
- Why?
 - Small bundle (~400KB)
 - Code runs on .NET Core server, no constraints
 - Full Debugger in VS
- Why not?
 - More load on server
 - Does not support disconnects

Demo #4

- Blazor on the Server

Blazor 3rd party Components

- [Telerik](#)
- [Syncfusion](#)
- [DevExpress](#)

Current Status - What's there?

- Component Model
- Routing
- Layouts
- Dependency Injection
- JS interop
- Share Components between projects
- Debugging in Chrome – Shift + ALT + D
- Forms and Validation
- VS and some VS Code support
- Blazor Server-Side
- Server Side Rendering

Current Status - What's coming?

- Better tooling
- Hot reloading
- AOT
- Smaller bundle size
- AuthN + AuthZ work
- Debugging in VS

So. Many. Decisions.

Editor

- Which one?
- Which plugins?
- Use built in terminal?
- Editor config

Module format

- ES6 Modules, CommonJS...

HTML generation

- Minify?
- Use plugin?
- Inject prod only concerns?
- Templating language?

Transpiling

- Native ES or diff language?
- Use experimental features?
- Which plugins?
- Production vs dev config

Bundler

- Webpack, Browserify, Rollup...

Linting

- Which linter?
- Enable which rules?
- Warning or error?
- Which plugins?
- Use a preset?

Testing

- Framework?
- Assertion Library?
- Helpers?
- Test file location?
- File naming?
- What environment?
- Mocking?
- Code Coverage
- Continuous Integration

Project structure

- By file type or feature?
- Centralize API?
- Allow Inline JS?
- Extract to POJOs?

HTTP

- Library
- Mock schema format
- Mock data generation
- Mock server

Production build

- Minification
- Sourcemaps
- Bundle splitting
- Cache busting
- Error logging



So. Many. Decisions.

Editor

Which one?

Which plugins?

Use built in terminal?

Bundler

Webpack, Browserify, Rollup...

Linting

Which linter?

Project structure

By file type or feature?

Centralize API?

Allow inline JS?

Extract to POJOs?

HTTP

Library

Mock schema format

Mock data generation

Mock server

Production build

Minification

Sourcemaps

Bundle splitting

Cache busting

Error logging

The remainder of these you've likely already decided on the backend!

Minify?

Use plugin?

Inject prod only concerns?

Templating language?

Transpiling

Native ES or diff language?

Use experimental features?

Which plugins?

Production vs dev config

Testing

Framework?

Assertion Library?

Helpers?

Test file location?

File naming?

What environment?

Mocking?

Code Coverage

Continuous Integration



Current State of the SPA Front End

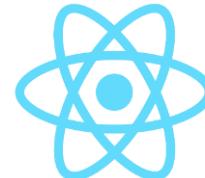
Pick a Language:



flow



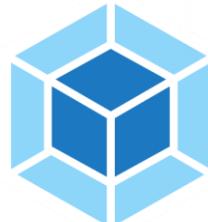
Pick a Framework:



ember



Pick your tools:



BABEL

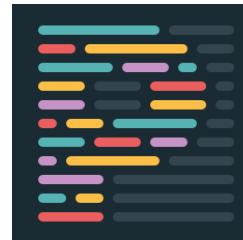


ESLint

Common

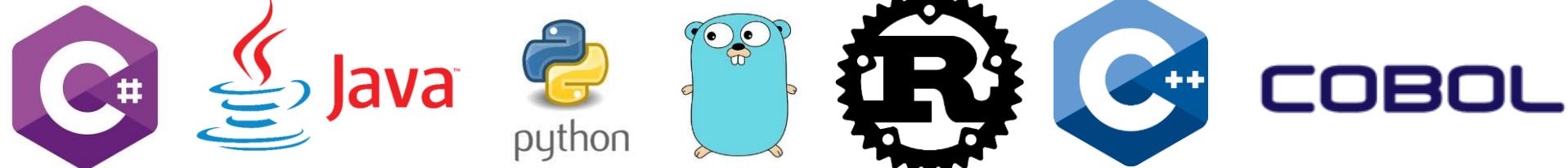


Google



Future State of the Front End? (besides JS)

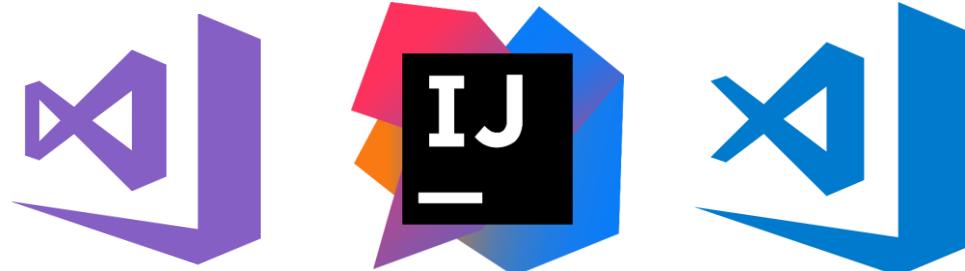
Pick a Language:



Pick a Framework:



Pick your tools:



Future

- Currently 3.0 preview4
- Client-side Blazor is NO LONGER experimental and will be shipped sometime in the future.
- Server-side Blazor will ship with .NET Core 3.0 later this year.

Takeaways

- WASM is AWSM
- Potential of Blazor
- WASM has potential to radically disrupt WebDev
- Start thinking about “would this code run ok in the browser?”
 - Separate domain + input validation

How do I get started?

- Today:
 - .NET Core 3.0 Preview 4 SDK (3.0.100-preview4-011223)
 - Visual Studio 2019 (Preview 4 or later) with the ASP.NET and web development workload selected.
 - The [latest Blazor extension](#) from the Visual Studio Marketplace.
 - The Blazor templates on the command-line:
 - `dotnet new -i Microsoft.AspNetCore.Blazor.Templates::3.0.0-preview4-19216-03`
- Future:
 - NET Core 3.0+
 - VS/VSCode/whatever

Resources

- <https://blazor.net>
 - Microsoft Documentation
- <https://learn-blazor.net>
 - Community-led Documentation
- <https://github.com/aspnet/blazor>
 - Blazor Source Code
- <https://github.com/mbasso/awesome-wasm> and <https://github.com/appcypher/awesome-wasm-langs>
 - Lists of what other languages are doing with WASM

Questions?

Thanks!