

# 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?

- Lead Software Developer at Iowa Bankers
- Primarily .NET Developer
- React fanboy
- Actually enjoys JavaScript
- Blog primarily on ASP.NET Core on [scottsauber.com](https://scottsauber.com)

# Current State of the SPA Front End

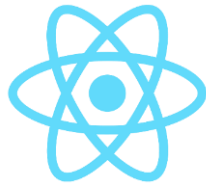
Pick a Language:



flow



Pick a Framework:



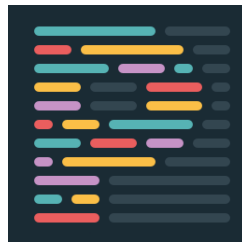
ember



Pick your tools:



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 solid yellow square that serves as a background for the 'JS' text.

**JS**



# 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
    - Plz don't just have client...plz
  - 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?

## Browser compatibility

New compatibility tables are in beta ▾

														
														
Basic support	57	16	52 <sup>*</sup> ▼	No	44	11	57	57	Yes	52 <sup>*</sup> ▼	?	11	7.0	8
CompileError	57	16	52 <sup>*</sup>	No	44	11	57	57	Yes	52 <sup>*</sup>	?	11	7.0	8

And you can polyfill WASM with asm.js!

Memory	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8
Module	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8
RuntimeError	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8
Table	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8
compile	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8
compileStreaming	61	16	58	No	47	No	61	61	No	58	?	No	No	No
instantiate	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8
instantiateStreaming	61	16	58	No	47	No	61	61	No	58	?	No	No	No
validate	57	16	52 *	No	44	11	57	57	Yes 	52 *	?	11	7.0	8

# What is Blazor?

- Blazor is an **experimental** .NET SPA framework 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 an **experimental** .NET SPA framework 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

# 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
- Future: automatically read TSD's and generate C# bindings



# 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.)
    - [Airbnb React Native blog](#)
- Share logic with existing .NET backend
  - Validation logic
  - Models from Server when retrieve from the Client
- Get off the JS churnwagon
- Consolidate frontend and backend teams under one language

# Demo #1

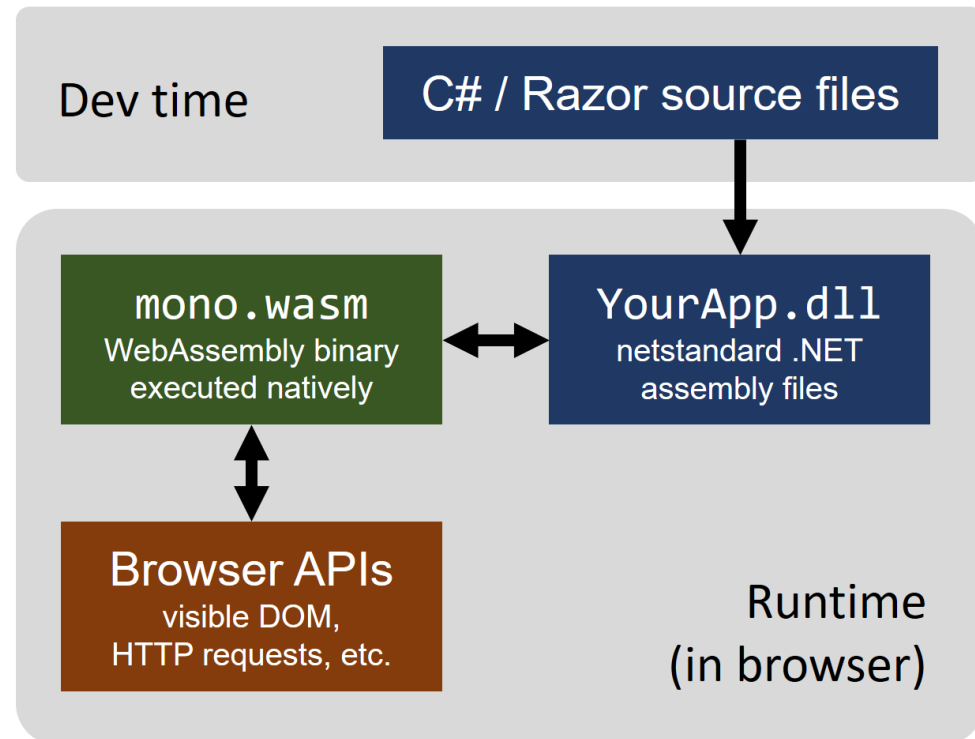
- [Install VS Extension](#)
  - Templates, Razor Tooling
- Hello World on Blazor
- Component Architecture
- Dependency Injection
- Sharing logic

# Rapid Fire Questions

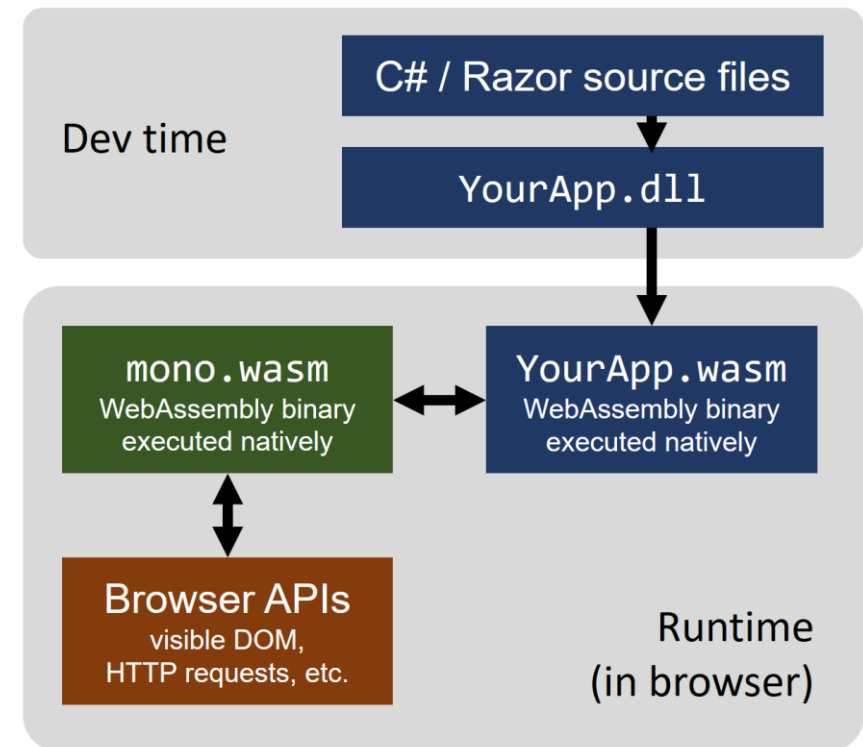
- How big is it?
  - 1.9MB in Dev, 1.4MB in Prod
  - Very little work done thus far to optimize
- Do WASM files cache like JS and CSS files?
  - Yes
- How does it work under the hood?

# How does Blazor work?

## Today



## Future



# 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
- Blazor on...
- The Server
  - Possible Replacement for MVC/Razor Pages if you prefer component-based over MV\*?
  - Changes streamed via WebSocket
- 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>



# Demo #4

- Blazor on the Server – if we have time

# What's the hold up?

- Currently 0.7 – client-side **won't** ship with .NET Core 3 (Q1/Q2-next year) per Damian Edwards.
  - Component model will ship in ASP.NET Core 3 (best of both worlds with partials and Tag Helpers)
- What's there
  - Component Model
  - Routing
  - Layouts
  - Dependency Injection
  - JS interop
  - Share Components between projects
  - Debugging – Shift + ALT + D
- What's still coming
  - Better tooling
  - Forms and Validation
  - Hot reloading
  - AOT
  - Better Linker Assembly Trimming

# 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

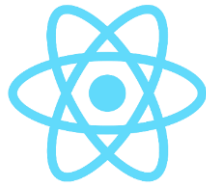
Pick a Language:



flow



Pick a Framework:



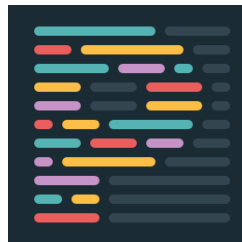
ember



Pick your tools:



ESLint



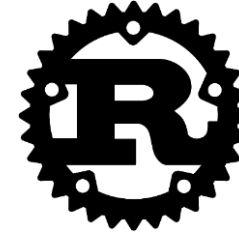
Common

Google

# Future State of the Front End?



Pick a Language:



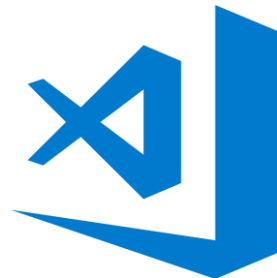
COBOL

Pick a Framework:

Blazor

???

Pick your tools:



# Future

- Blazor is still experimental...
- However... I would be pretty surprised if they don't ship this.
- Start thinking about “would this code run ok in the browser?”
  - Separate domain + input validation
- But still do NOT commit to Blazor yet for anything remotely real
- I repeat

# Takeaways

- WASM is AWSM
- Potential of Blazor
- WASM has potential to radically disrupt WebDev



# Resources

- <https://blazor.net>
  - Microsoft Documentation
- <https://learn-blazor.net>
  - Community-led Documentation
- <https://github.com/aspnet/blazor>
  - Blazor Source Code
- <https://github.com/scottsauber/iowa-code-camp-2018-blazor>
  - Code from demos today
- <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!