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:
- ES6
- TS
- flow
- RE
- cljs

Pick a Framework:
- React
- Angular
- Vue
- ember
- Svelte
- Preact
- LWC
- Common

Pick your tools:
- Gulp
- Babel
- ESLint
- Common
### So. Many. Decisions.

<table>
<thead>
<tr>
<th>Editor</th>
<th>Bundler</th>
<th>Project structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which one?</td>
<td>Webpack, Browserify, Rollup...</td>
<td>By file type or feature?</td>
</tr>
<tr>
<td>Which plugins?</td>
<td>Linting</td>
<td>Centralize API?</td>
</tr>
<tr>
<td>Use built in terminal?</td>
<td>Which linter?</td>
<td>Allow Inline JS?</td>
</tr>
<tr>
<td>Editor config</td>
<td>Enable which rules?</td>
<td>Extract to POJOs?</td>
</tr>
<tr>
<td></td>
<td>Warning or error?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Which plugins?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use a preset?</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Testing</strong></td>
<td><strong>HTTP</strong></td>
</tr>
<tr>
<td></td>
<td>Framework?</td>
<td>Library</td>
</tr>
<tr>
<td></td>
<td>Assertion Library?</td>
<td>Mock schema format</td>
</tr>
<tr>
<td></td>
<td>Helpers?</td>
<td>Mock data generation</td>
</tr>
<tr>
<td></td>
<td>Test file location?</td>
<td>Mock server</td>
</tr>
<tr>
<td></td>
<td>File naming?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>What environment?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mocking?</td>
<td>Production build</td>
</tr>
<tr>
<td></td>
<td>Code Coverage</td>
<td>Minification</td>
</tr>
<tr>
<td></td>
<td>Continuous Integration</td>
<td>Sourcemaps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bundle splitting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cache busting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Error logging</td>
</tr>
<tr>
<td>Transpiling</td>
<td>[Native ES or diff language?]</td>
<td></td>
</tr>
<tr>
<td>Use experimental features?</td>
<td>[Use a preset?]</td>
<td></td>
</tr>
<tr>
<td>Which plugins?</td>
<td>[Warning or error?]</td>
<td></td>
</tr>
<tr>
<td>Production vs dev config</td>
<td>[Which rules?]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Cory House’s [Building a JavaScript Development Environment Pluralsight course](https://www.pluralsight.com)
At the end of the day....
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
Is WASM Ready?

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
“... 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**

- **Dev time**
  - C# / Razor source files
  - `mono.wasm`
    - WebAssembly binary executed natively
  - `YourApp.dll`
    - netstandard .NET assembly files

- **Runtime (in browser)**
  - Browser APIs
    - visible DOM, HTTP requests, etc.

**Future**

- **Dev time**
  - C# / Razor source files
  - `YourApp.dll`

- **Runtime (in browser)**
  - `mono.wasm`
    - WebAssembly binary executed natively
  - `YourApp.wasm`
    - WebAssembly binary executed natively

- Browser APIs
  - visible DOM, HTTP requests, etc.
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.

<table>
<thead>
<tr>
<th>Editor</th>
<th>Bundler</th>
<th>Project structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which one?</td>
<td>Webpack, Browserify, Rollup...</td>
<td>By file type or feature?</td>
</tr>
<tr>
<td>Which plugins?</td>
<td></td>
<td>Centralize API?</td>
</tr>
<tr>
<td>Use built in terminal?</td>
<td></td>
<td>Allow Inline JS?</td>
</tr>
<tr>
<td>Editor config</td>
<td></td>
<td>Extract to POJOs?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module format</th>
<th>Linter</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES6 Modules, CommonJS...</td>
<td>Which linter?</td>
<td>Library</td>
</tr>
<tr>
<td>HTML generation</td>
<td>Enable which rules?</td>
<td>Mock schema format</td>
</tr>
<tr>
<td>Minify?</td>
<td>Warning or error?</td>
<td>Mock data generation</td>
</tr>
<tr>
<td>Use plugin?</td>
<td>Which plugins?</td>
<td>Mock server</td>
</tr>
<tr>
<td>Inject prod only concerns?</td>
<td>Use a preset?</td>
<td></td>
</tr>
<tr>
<td>Templating language?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transpiling</th>
<th>Testing</th>
<th>Production build</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native ES or diff language?</td>
<td>Framework?</td>
<td>Minification</td>
</tr>
<tr>
<td>Use experimental features?</td>
<td>Assertion Library?</td>
<td>Sourcemaps</td>
</tr>
<tr>
<td>Which plugins?</td>
<td>Helpers?</td>
<td>Bundle splitting</td>
</tr>
<tr>
<td>Production vs dev config</td>
<td>Test file location?</td>
<td>Cache busting</td>
</tr>
<tr>
<td></td>
<td>File naming?</td>
<td>Error logging</td>
</tr>
<tr>
<td></td>
<td>What environment?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mocking?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Code Coverage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuous Integration</td>
<td></td>
</tr>
</tbody>
</table>
The remainder of these you’ve likely already decided on the backend!
Current State of the SPA Front End

Pick a Language:
- ES6
- TS
- flow
- RE
- cljs

Pick a Framework:
- React
- Angular
- Vue
- Ember
- Svelte
- Common

Pick your tools:
- Gulp
- Babel
- ESLint
- Common
Future State of the Front End? (besides JS)

Pick a Language:
- C#
- Java
- Python
- R
- C++
- COBOL

Pick a Framework:
- @
- uno
- Ooui
- ???

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

Slides at scottsauber.com
Resources

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

Slides at scottsauber.com
Questions?
Thanks!