

WebAssembly and .NET

Joanna Lamch

2019





- *Who am I ?*
- Joanna Lamch

JLamch@gmail.com

JLamch.net

ProgramistkaIKot.pl

- Microsoft fangirl
- Developer C#

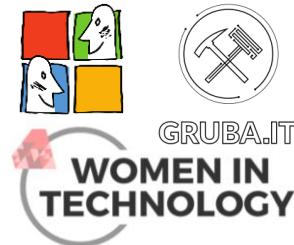
.NET Framework 1.1

15 years (+ overtime)

Xamarin

SIENN

- Community

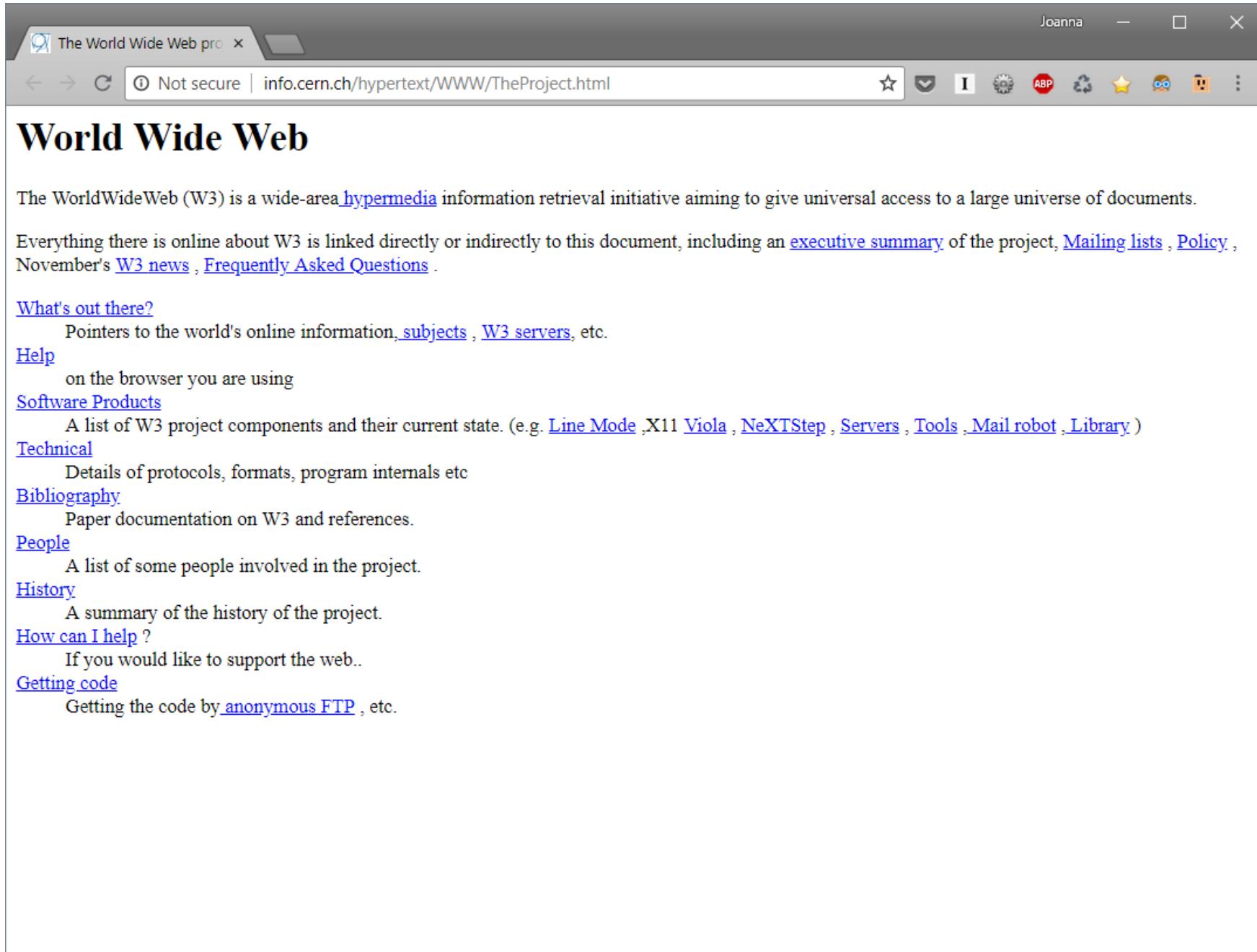


Śląska Grupa Microsoft
Women In Technology
Gruba.IT



Back in the days...

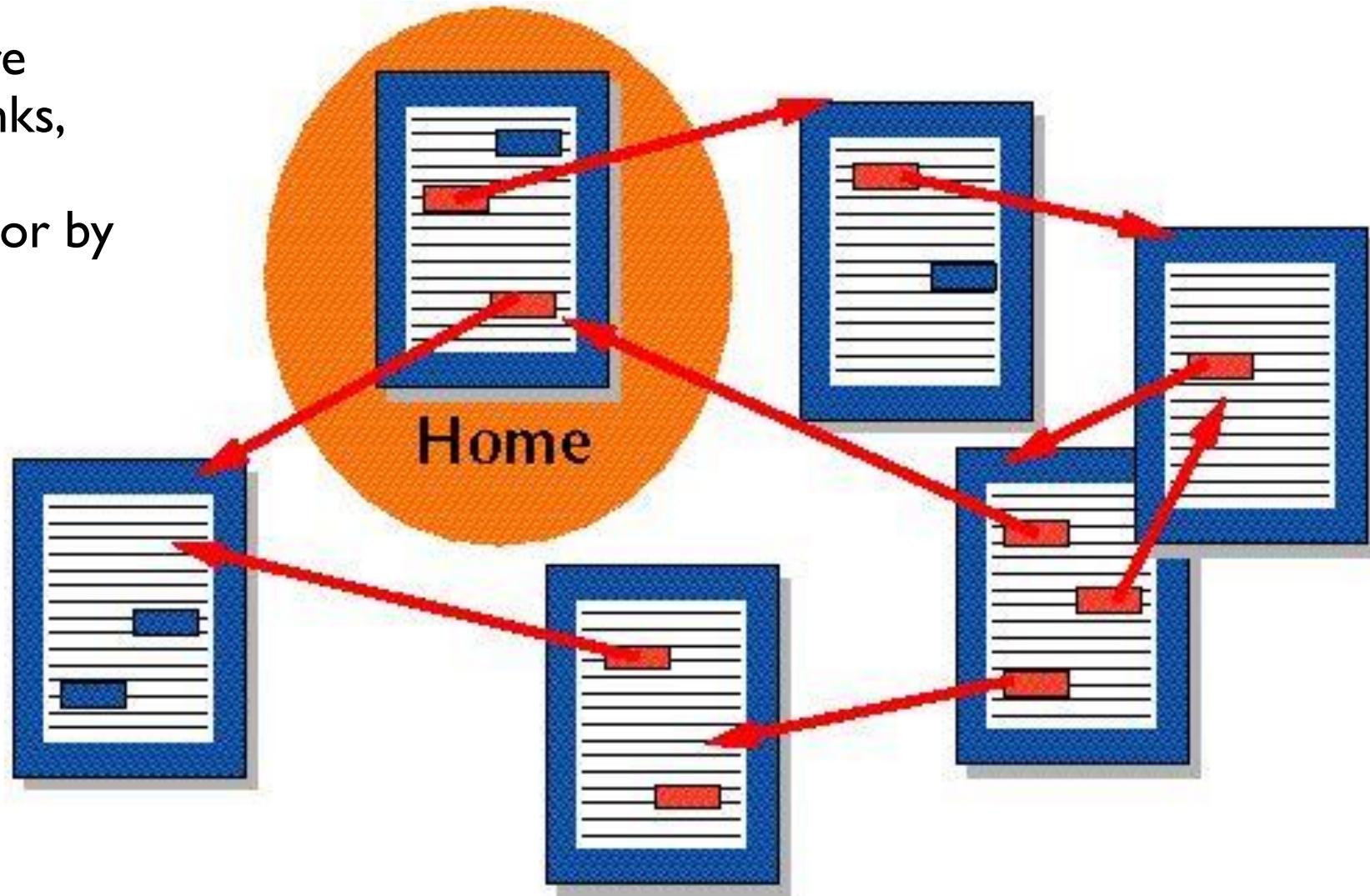




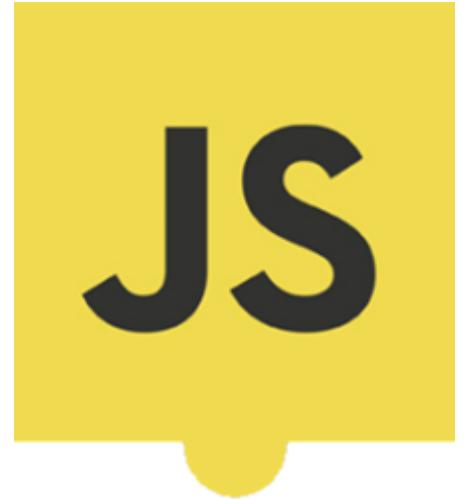
1991

Hypertext, HTML, HTTP

Hypertext documents are interconnected by hyperlinks, which are activated by a mouseClick, keypress set or by touching the screen



JavaScript

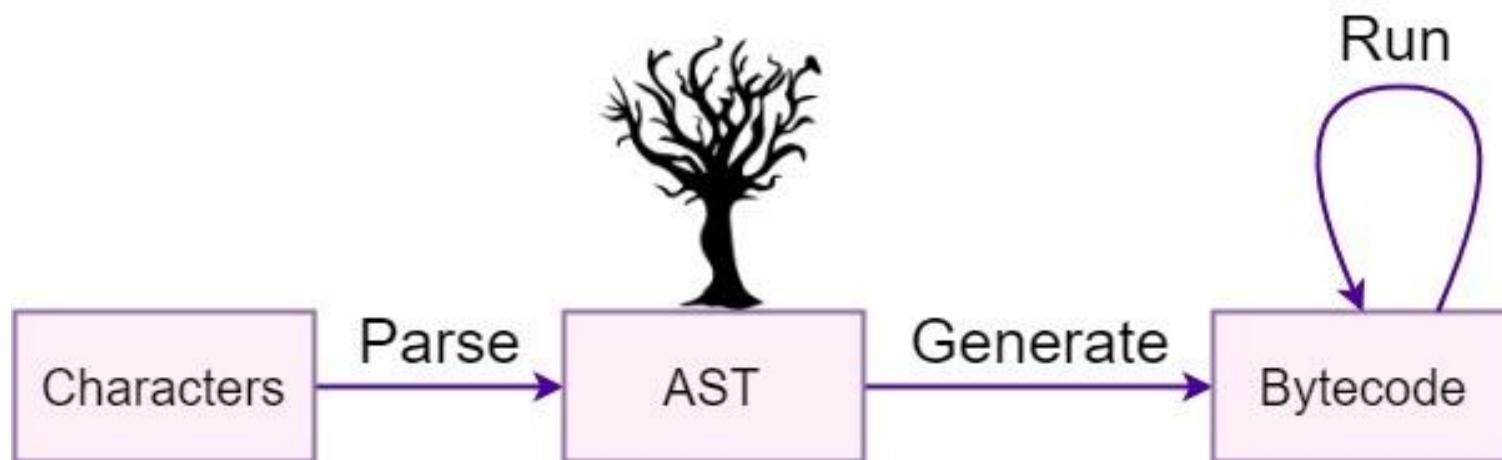
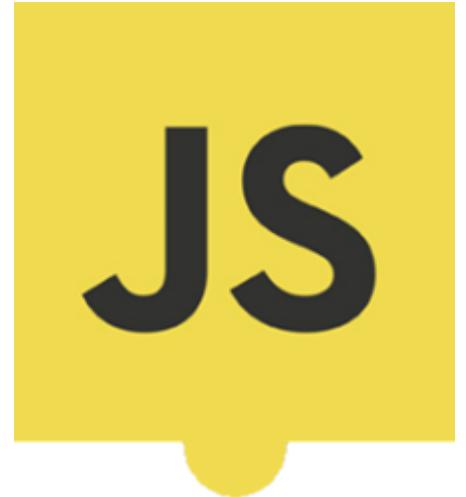


created in 10 days in May 1995, by Brendan Eich for Netscape

Adding interactivity to HTML pages

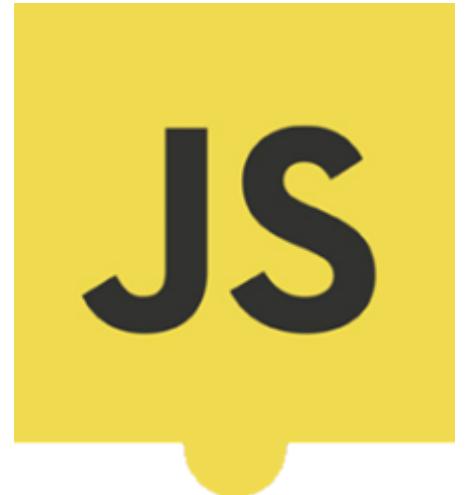
1995

JavaScript



1995

JavaScript



1995

<https://www.smashingmagazine.com/2017/05/abridged-cartoon-introduction-webassembly/> Lin Clark

Other plugins

Java Applets [1997]



ActiveX [1996]

Flash [1996]

Silverlight [2007]

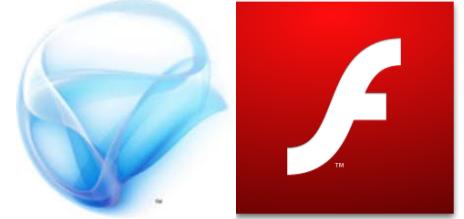
1996+

Other plugins

Java Applets [1997]



ActiveX [1996]



Flash [1996]

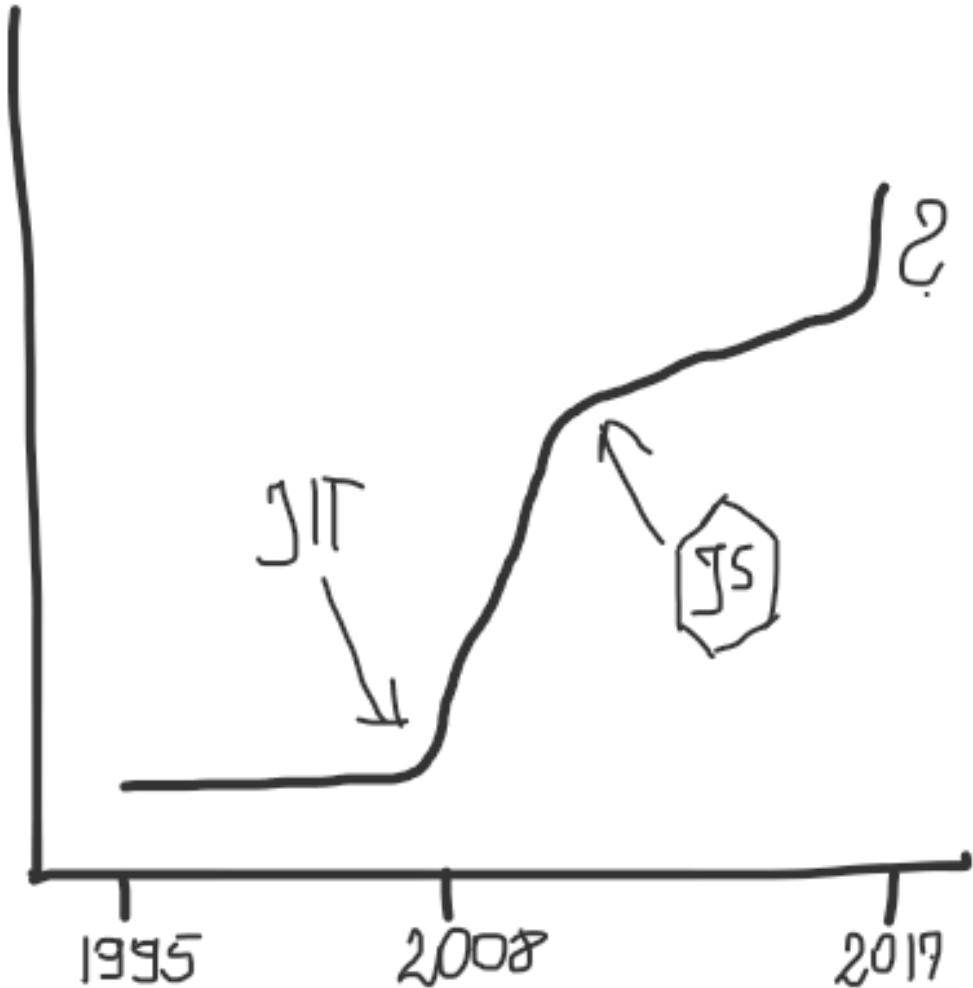
All of them are deprecated
or will be deprecated soon



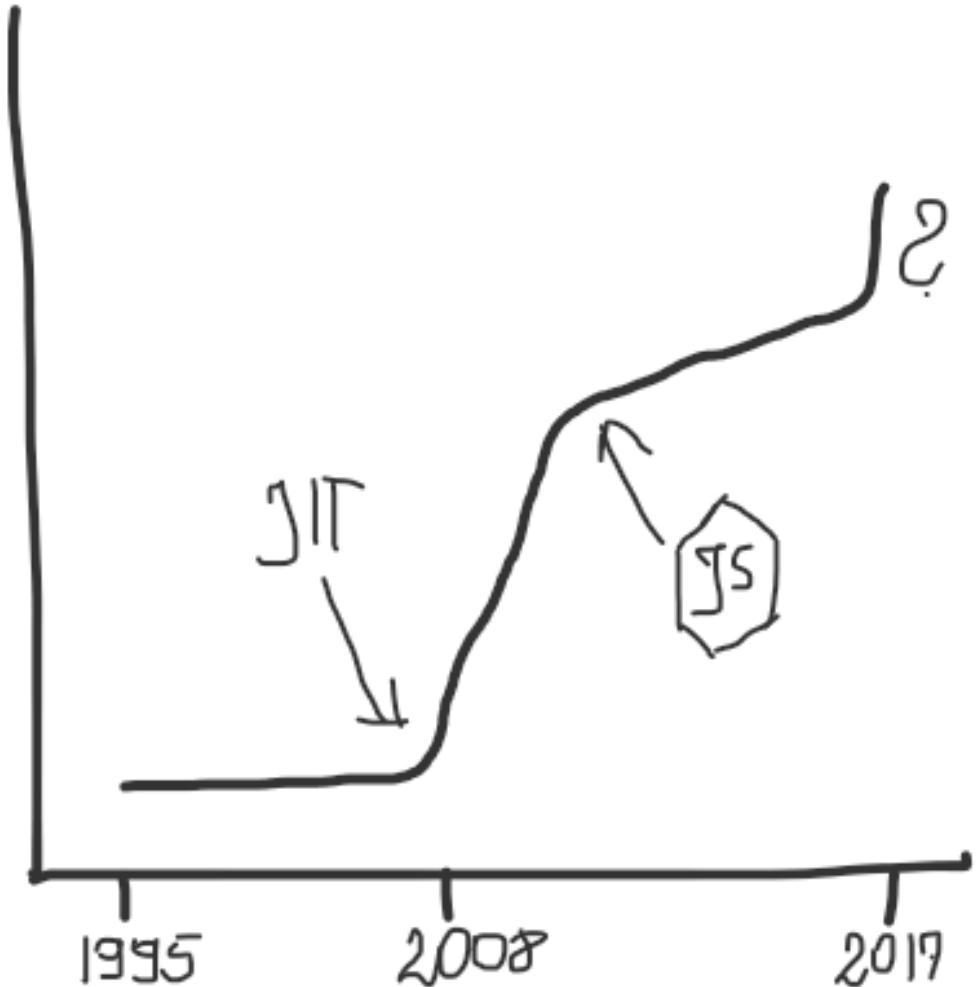
Silverlight [2007]

1996+

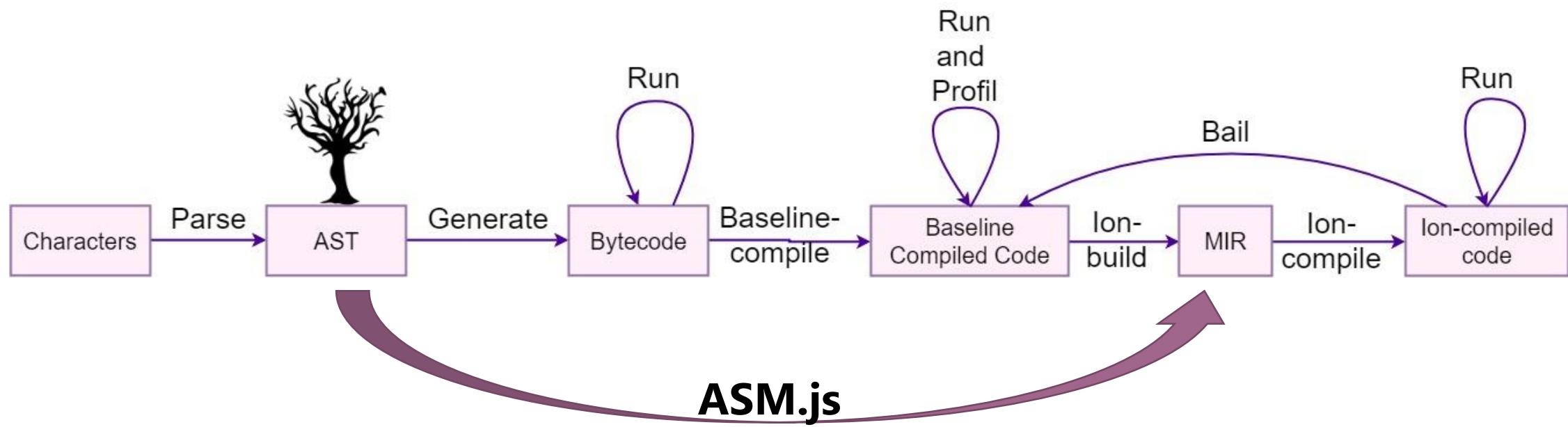
Browser performance wars (2008+)



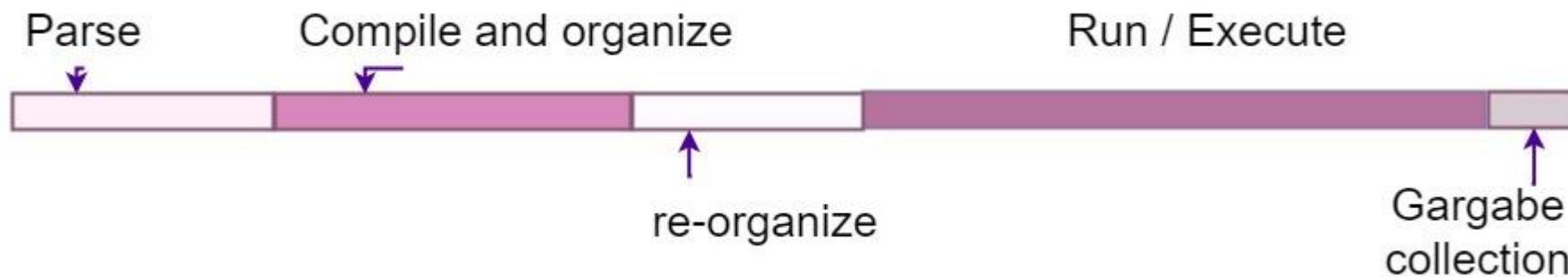
Browser performance wars (2008+)



Browser performance wars (2008+)



Browser performance wars (2008+)



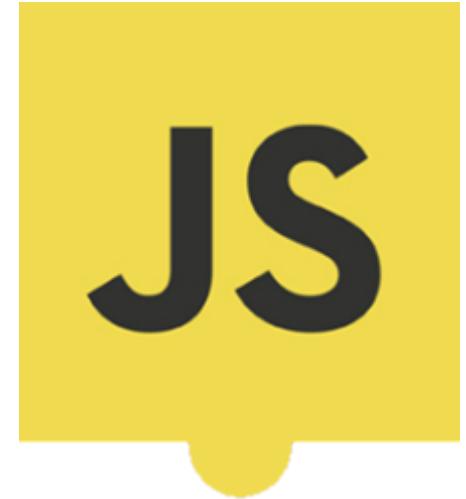
Todays JavaScript

Different way the same language

There is a LOT of JavaScript

JavaScript is the language of the Web

But it's not very good Assembly Language
(still human readable simple language)



Todays JavaScript



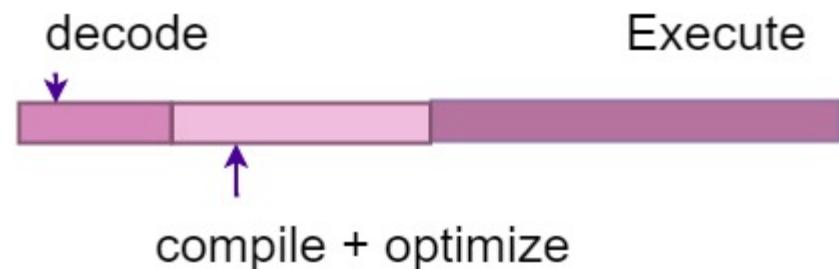
JS



WebAssembly What? Why?

A new low-level **binary** format **for the web** (WASM)

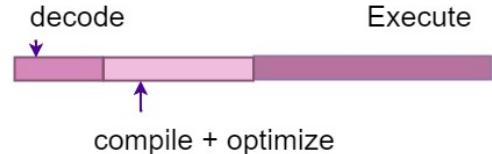
It's a bytecode for web / compilation target ➔ maximized performance



WebAssembly What? Why?

A new low-level **binary** format **for the web** (WASM)

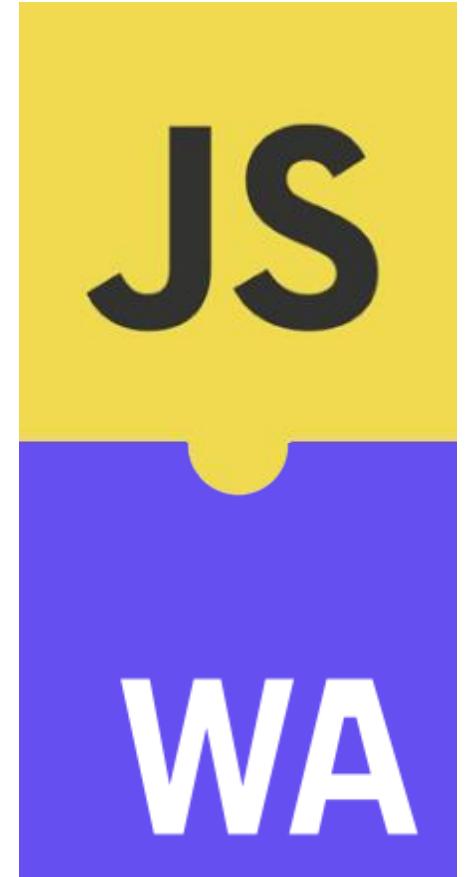
It's a bytecode for web / compilation target ➔ **maximized performance**



WebAssembly **is not designed to replace JS**, but to coexist

Sandboxed runtime in JS virtual machine

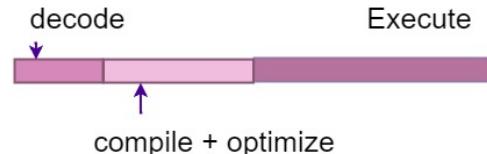
Security it runs locally in JS VM



WebAssembly What? Why?

A new low-level **binary** format **for the web** (WASM)

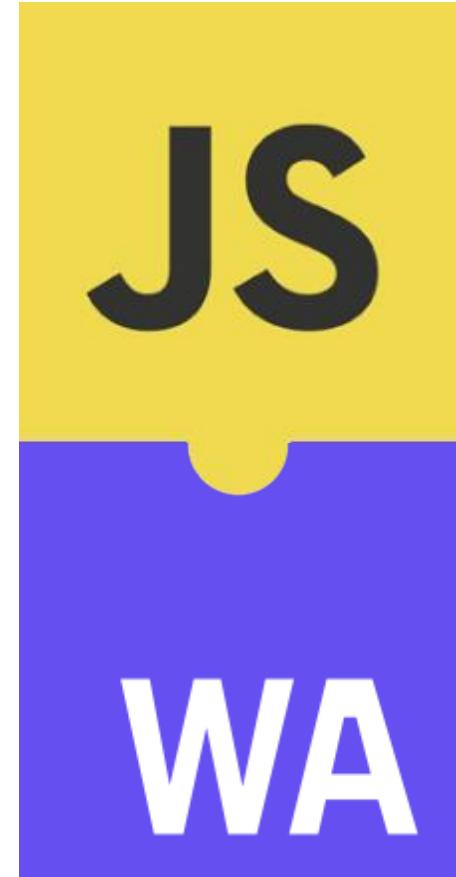
It's a bytecode for web / compilation target ➔ **maximized performance**



WebAssembly **is not designed to replace JS**, but to coexist

Sandboxed runtime in JS virtual machine

Security it runs locally in JS VM



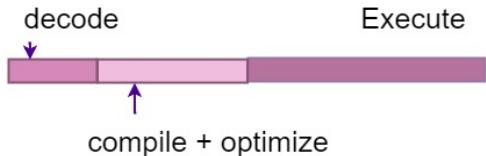
Supported in **all big browsers** –
W3C open specification



WebAssembly What? Why?

A new low-level **binary** format for the web (WASM)

It's a bytecode for web / compilation target ➔ maximized performance



WebAssembly is not designed to replace JS, but to coexist

Sandboxed runtime in JS virtual machine

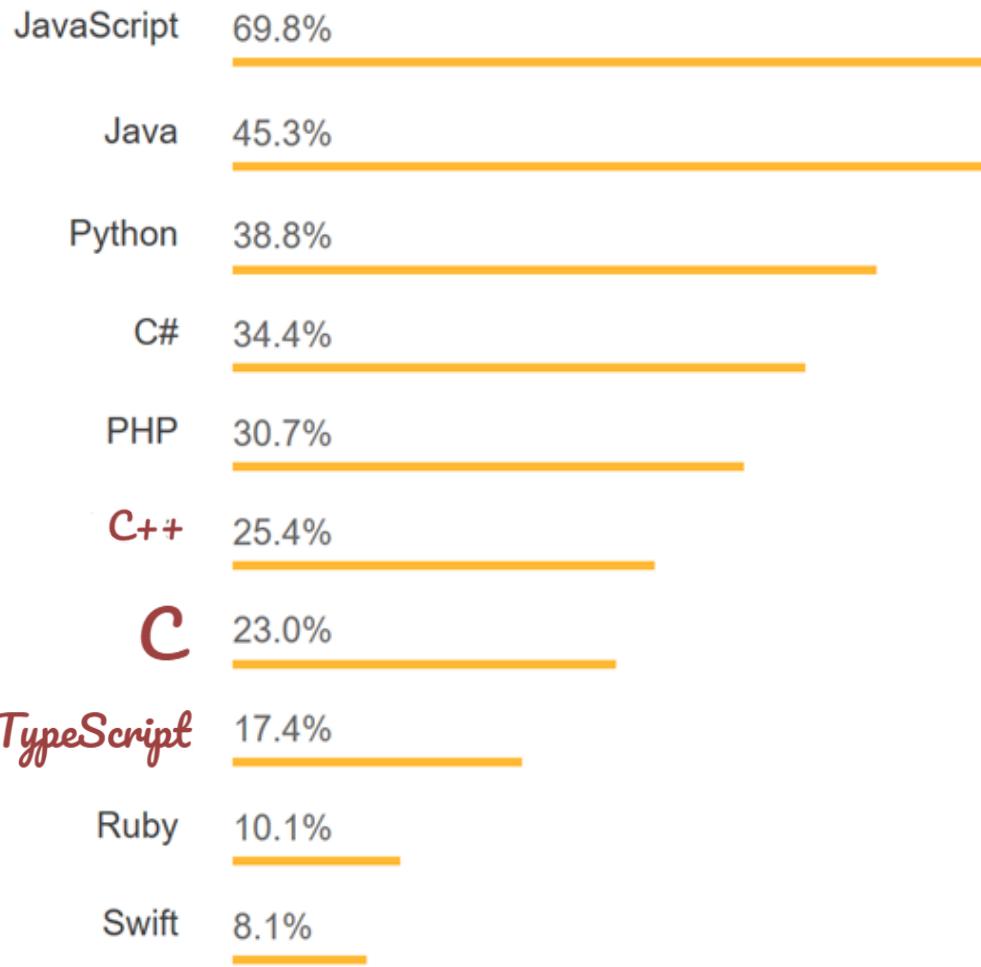
Security it runs locally in JSVM

Supported in **all** big browsers – W3C open specification

Compiled from other languages



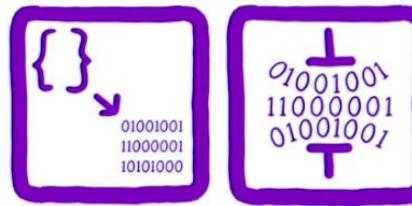
WASM opens possibilities for other languages



WebAssembly current state: MVP

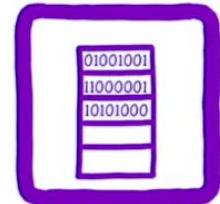
- Compilation target

Binary so compact



- 4 types, 67 instructions,
stack machine

- Linear memory



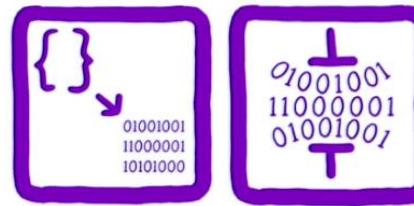
- Fast execution



WebAssembly current state: MVP

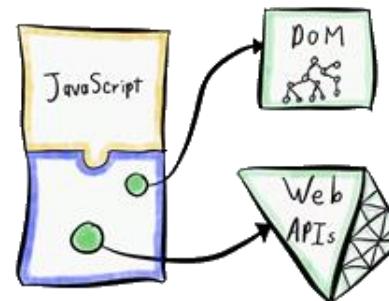
- Compilation target

Binary so compact

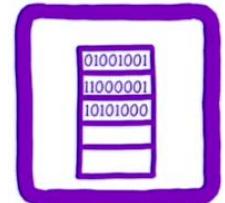


- 4 types, 67 instructions, stack machine

- No Web APIs, DOM access/manipulation



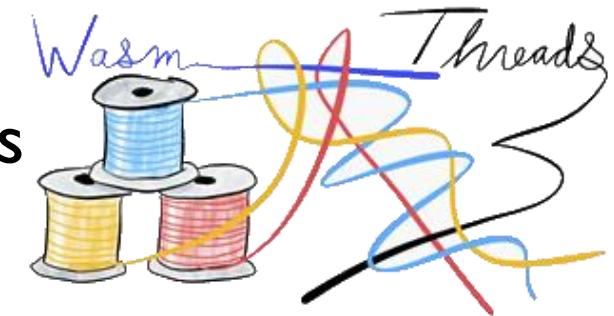
- Linear memory



- Fast execution or not so fast
not so fast load



- no Threads



WebAssembly How?



emsripten

```
$emcc -s WASM=1 -o example.js example.cpp
```

executes the **Emscripten** compiler

name our output file

Emscripten outputs WebAssembly - switch

input file

WebAssembly studio C/Rust/TypeScript

The image displays two side-by-side screenshots of the WebAssembly Studio web application. Both screenshots show a dark-themed interface with a header bar containing the 'WebAssembly Studio' logo, a search bar with the URL 'https://webassembly.studio', and various navigation and tool icons.

Left Screenshot (build.ts):

- File Structure:** Shows files: README.md, build.ts (selected), package.json, main.html, main.js, and main.rs.
- Code Editor:** Displays the content of build.ts:

```
1 import * as gulp from "gulp";
2 import { Service, project } from "@wasm/studio-util"
3
4 gulp.task("build", async () => {
5   const options = { lto: true, opt_level: 's', debug: false }
6   const data = await Service.compileFile(project.getMainFile())
7   const outWasm = project.newFile("out/main.wasm",
8     outWasm.setData(data);
9   );
10
11 gulp.task("default", ["build"], async () => {});
```
- Output:** Shows 'Output (1)' tab with the number 1.
- Problems:** Shows 'Problems (0)' tab.

Right Screenshot (main.ts):

- File Structure:** Shows files: README.md, assembly (selected), main.ts, tsconfig.json, gulpfile.js, package.json, setup.js, main.html, and main.js.
- Code Editor:** Displays the content of main.ts:

```
1 @external("env", "sayHello")
2 declare function sayHello(): void;
3
4 declare namespace console {
5   function logi(value: i32): void;
6   function logf(value: f64): void;
7 }
8
9 sayHello();
10
11 export function add(x: i32, y: i32): i32 {
12   return x + y;
13 }
14
15 console.logi(add(1, 2));
```
- Output:** Shows 'Output (2)' tab with the text 'Loading AssemblyScript compiler ...' and the number 2.
- Problems:** Shows 'Problems (0)' tab.

Demo

<http://mbebenita.github.io/WasmExplorer/>

<https://wasdk.github.io/WasmFiddle/>

<https://webassembly.studio/>



WasmExplorer

The screenshot shows the WasmExplorer interface with the following components:

- C++11 -Os**: The input code area containing the following C++ code:

```
1 float add(float a, float b){  
2     return a+b;  
3 }
```
- COMPILE**: A button to compile the C++ code.
- Wat**: The WebAssembly text representation of the compiled code, starting with:

```
1 (module  
2   (table 0 anyfunc)  
3   (memory $0 1)  
4   (export "memory" (memory $0))  
5   (export "_Z3addff" (func $_Z3addff))  
6   (func $_Z3addff (; 0 ;) (param $0 f32) (param  
7       $1 f32) (result f32)  
8       (f32.add  
9           (get_local $0)  
10          (get_local $1)  
11      )  
12  )  
13 )
```
- ASSEMBLE**: A button to assemble the code.
- DOWNLOAD**: A button to download the code.
- Firefox x86 Assembly**: A section showing the assembly output for Firefox x86, starting with:

```
wasm-function[0]:  
    sub rsp, 8  
    addss xmm0, xmm1  
    nop  
    add rsp, 8  
    ret
```

With corresponding assembly mnemonics and hex addresses.

<https://bit.ly/2JkqFYt>

C main.c

JS main.js

WA main.wasm



C main.c

```
1  fetch('../out/main.wasm').then(response =>
2    response.arrayBuffer()
3  )
4  .then(bytes => WebAssembly.instantiate(bytes)).then(result=> {
5    instance= result.instance;
6    document.getElementById("container").textContent = "Result: " +
7      + instance.exports.add(19.19, 23.23);
8  }).catch(console.error);
9
```

```
1  #include <stdio.h>
2  #include <sys/uio.h>
3  #include <math.h>
4  #define WASM_EXPORT __attribute__((visibility("default")))
5
6  WASM_EXPORT
7  int main() {
8    return 42;
9  }
10
11 WASM_EXPORT
12 double add(float a, float b){
13   float f=a+b;
14   float num = floor(100*f)/100;
15   return f;
16 }
17
18
```

≡ Output (50)

```
41 [info]: Task build is completed
42 [info]: Task build is running...
43 [info]: Task build is completed
44 [info]: Task build is running...
45 [info]: Task build is completed
46 Downloading Project ...
47 Project Zip CREATED
48 [info]: Task build is running...
49 [info]: Task build is completed
50
```

Result: 42.41999816894531

C main.c

[Save](#)

```
1 #include <stdio.h>
2 #include <sys/uio.h>
3 #include <math.h>
4 #define WASM_EXPORT __attribute__((visibility("default")))
5
6 WASM_EXPORT
7 int main() {
8     return 42;
9 }
10
11 WASM_EXPORT
12 double add(float a, float b){
13     float f=a+b;
14     float num = floor(100*f)/100;
15     return f;
16 }
```

999816894531

C main.c

JS main.js

WA main.wasm

Save

C main.c

Save

README.md
build.ts
package.json
src
main.c
main.html
main.js
out
main.wasm

```
1 fetch('../out/main.wasm').then(response =>
2   response.arrayBuffer()
3 )
4 .then(bytes => WebAssembly.instantiate(bytes)).then(result=> {
5   instance= result.instance;
6   document.getElementById("container").textContent = "Result: "
7   + instance.exports.add(19.19, 23.23);
8 }).catch(console.error);
9
```

Result: 42.41999816894531

Output (50)

```
41 [info]: Task build is com
42 [info]: Task build is run
43 [info]: Task build is com
44 [info]: Task build is run
45 [info]: Task build is com
46 Downloading Project ...
47 Project Zip CREATED
48 [info]: Task build is running...
49 [info]: Task build is completed
50
```

Result: 42.41999816894531

C main.c

```
1 fetch
2   res
3 }
4 .then
5   ins
6   doc
7   + i
8 }).ca
9 }
```

main.c

```
#include <stdio.h>
#include <sys/uio.h>
#include <math.h>
#define WASM_EXPORT __attribute__((visibility("default")))

WASM_EXPORT
int main() {
    return 42;
}

WASM_EXPORT
double add(float a, float b){
    float f=a+b;
    float num = floor(100*f)/100;
    return f;
}
```

README.md

TS build.ts

{ package.json

src

C main.c

main.html

JS main.js

out

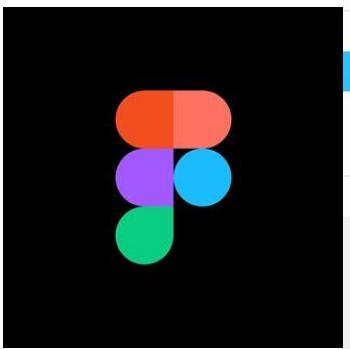
main.wasm

≡ Output

```
41 [info
42 [info
43 [info
44 [info
45 [info
46 Downloading...
47 Project...
48 [info
49 [info
50 ]
```

Result: 42.41999816894531

WASM in real world



Figma

<https://www.figma.com> # App Screens/Profile 9

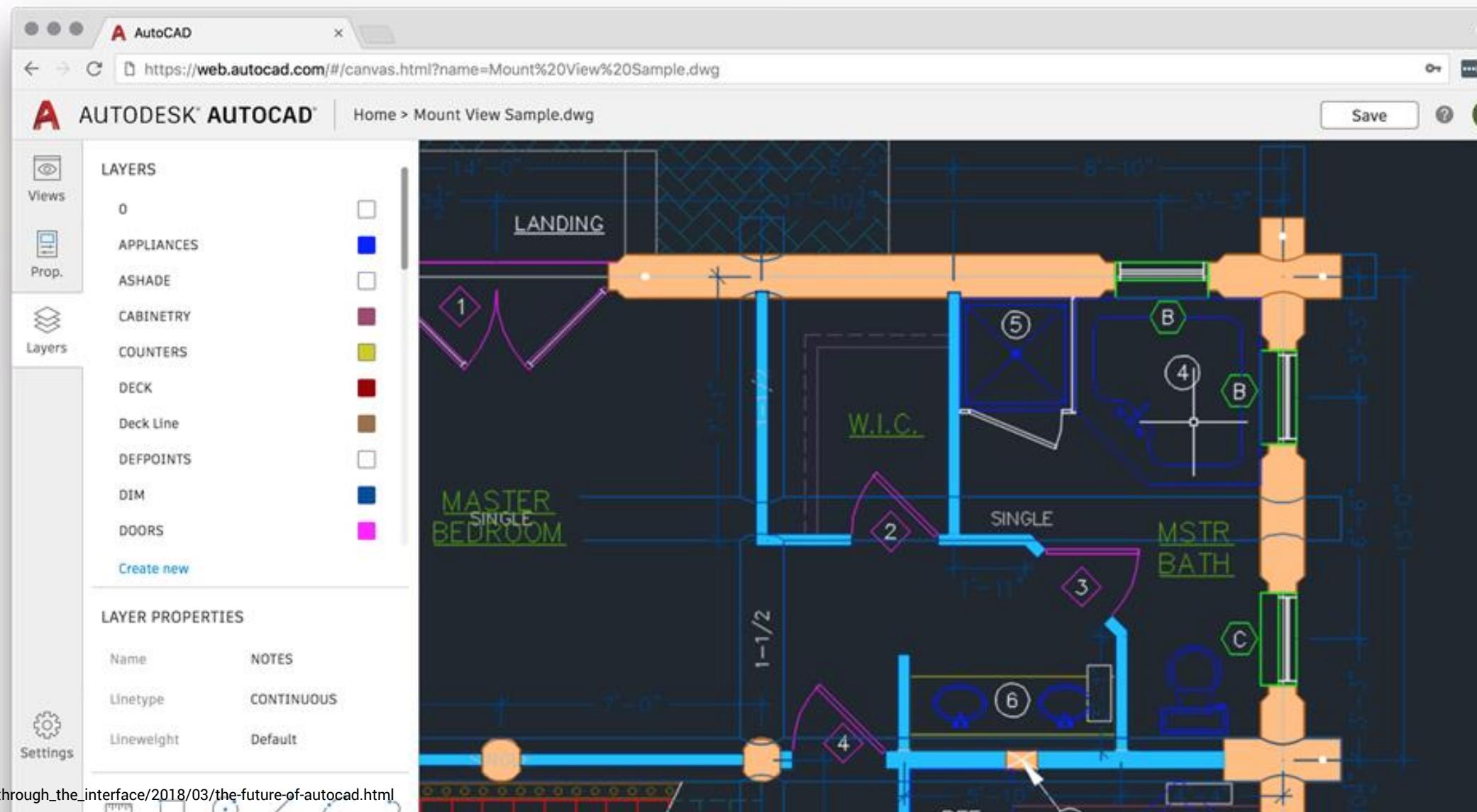
<https://www.youtube.com/watch?v=KfIglLChfk>

A collage of screenshots from various Figma projects and interfaces, demonstrating the use of WebAssembly (WASM) in real-world applications. The images include:

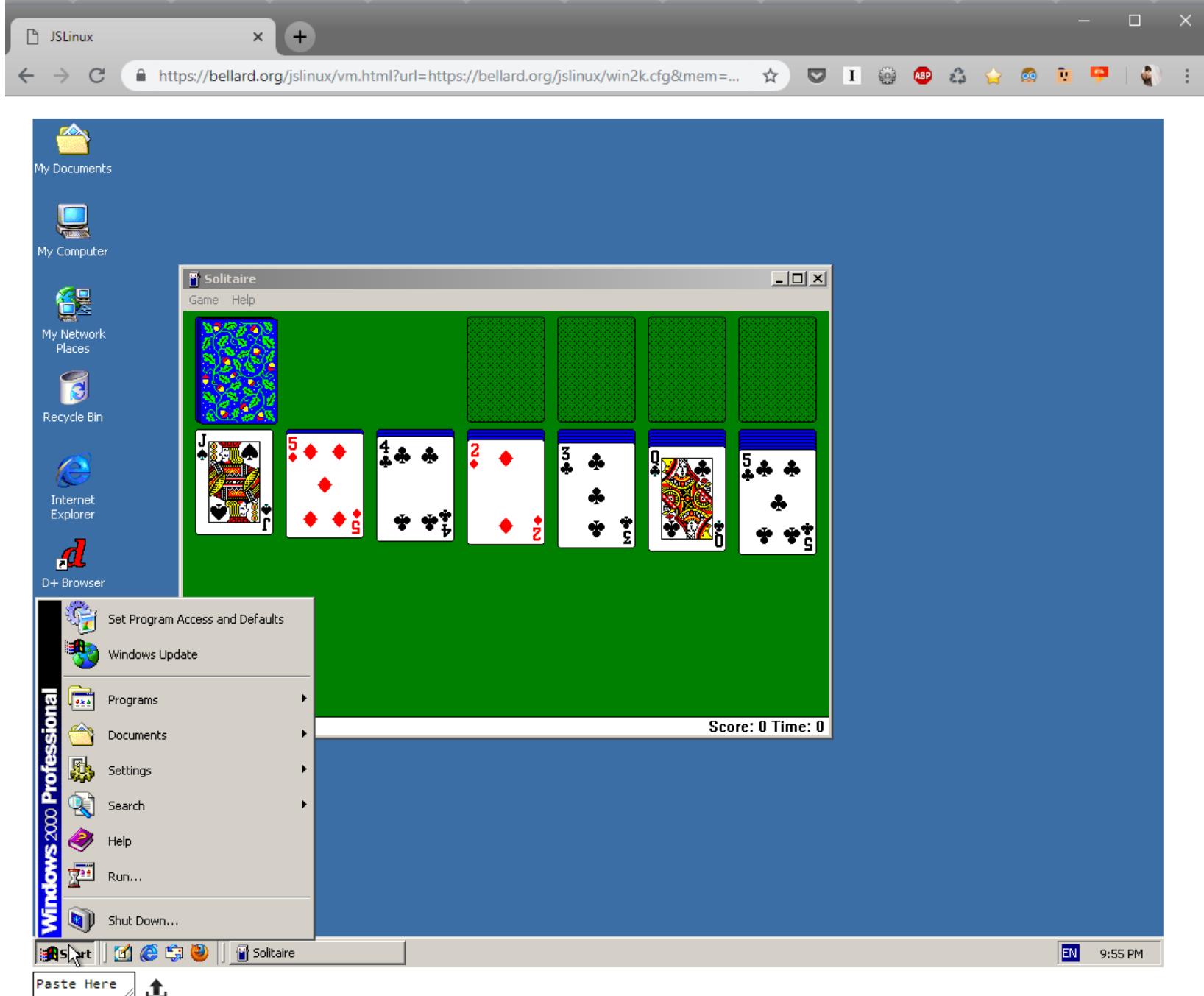
- A mobile navigation interface titled "Mobile templates" showing a map of San Francisco with a route from "Your location" to "Rockefeller Center, San Francisco".
- A search results page for "Events in San Francisco, California" listing events like "1. The History of Fonts" and "2. The Whiteboy Allstars / Mike Coach Band".
- A "Calendar" interface showing a weekly view with various events and tasks.
- A "Taskapp" interface with a grid of tasks and events.
- A "Masonry cards" interface showing cards for "Awesome Things" and a stool.
- A "Switter" interface showing tweets from users like Maria Varucci and Peolis Shushko.
- A "Trello cards" interface showing cards for "Completed Projects" and "Current Projects".
- A "Trello variant" interface showing a different layout for the Trello cards.
- A "Trello masonry" interface showing a masonry-style layout for Trello cards.
- A "Airbnb" interface showing a search bar and a list of nearby experiences.

WASM in real world

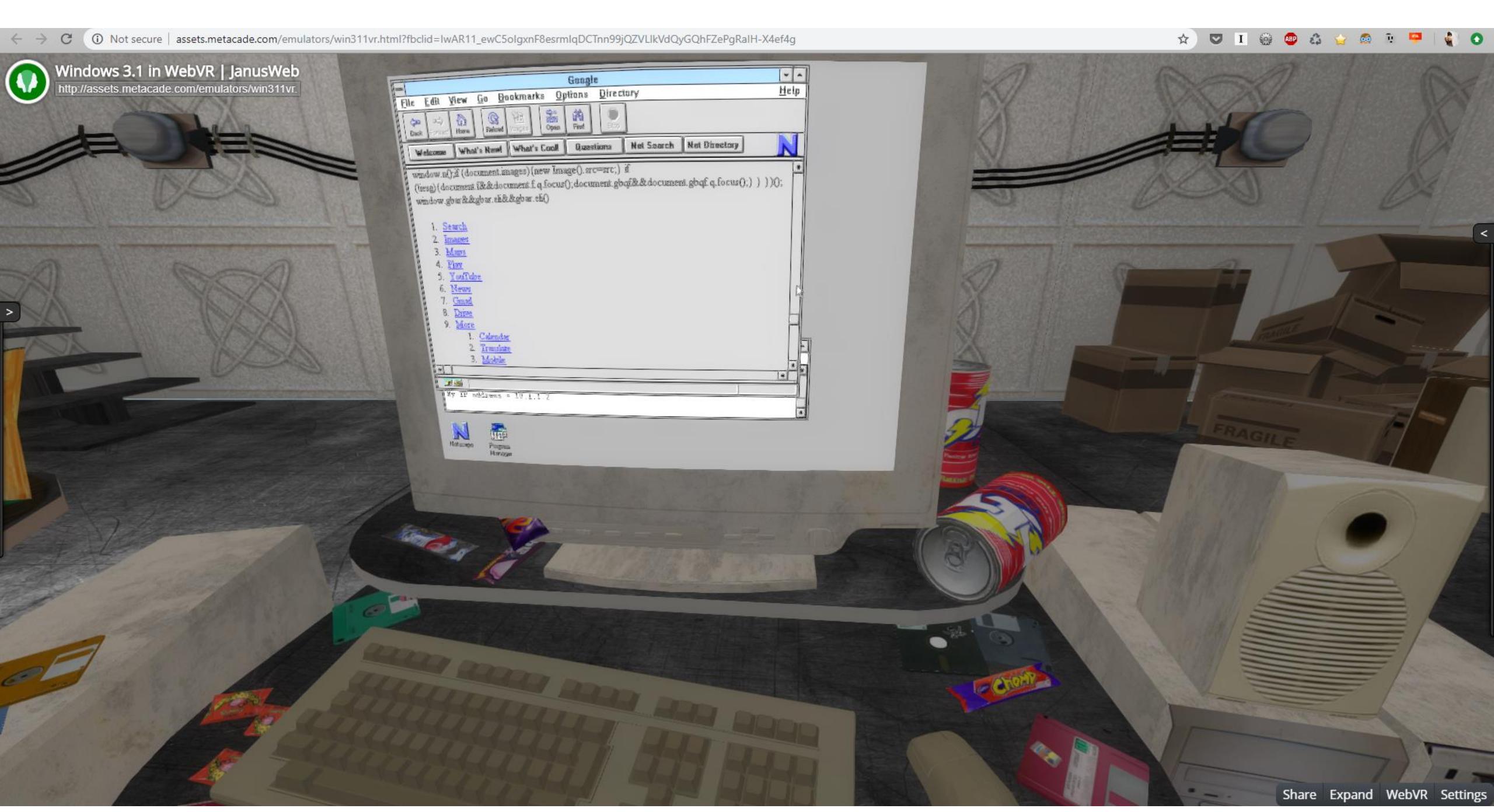
A



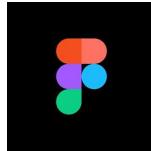




<https://bellard.org/jslinux/vm.html?url=https://bellard.org/jslinux/win2k.cfg&mem=192&graphic=1&w=1024&h=768>



- WASM in real world



Figma

<https://www.figma.com/>

<https://www.youtube.com/watch?v=KfIglLChfks>



https://s3.amazonaws.com.mozilla-games/ZenGarden/EpicZenGarden.html?fbclid=IwAR0_uAensGfTjIMzp4wXgVoxZjquxFo_uu2YD8yDuleTpPohaXyiIDd82X8

<https://bellard.org/jslinux/vm.html?url=https://bellard.org/jslinux/win2k.cfg&mem=192&graphic=1&w=1024&h=768>

<https://aesalazar.github.io/AsteroidsWasm/>

<http://sqliteefcore-wasm.platform.uno/>

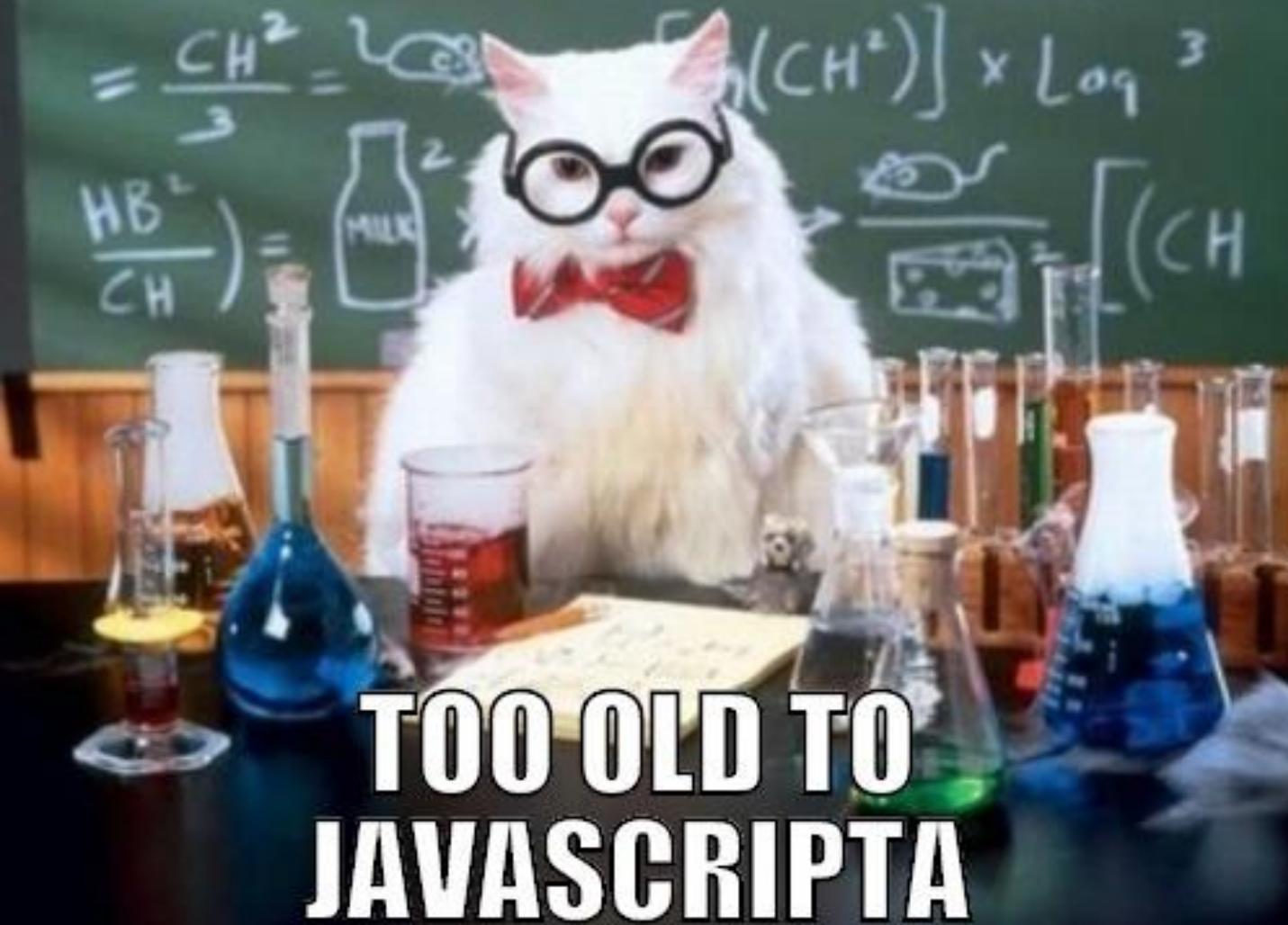
<https://raytracer-mono-act.platform.uno/>

http://www.continuation-labs.com/projects/d3wasm/?fbclid=IwAR2V9OqEDgu3bu-vMNlxZCUOm0HQAlv6ys-jcZGSMQY56saD8FYrHdVx_s

http://assets.metacade.com/emulators/win311vr.html?fbclid=IwARI1I_ewC5olgxnF8esrmlqDCTnn99jQZVLlkVdQyGQhFZePgRalH-X4ef4g



TOO JOUNG TO DIE



TOO OLD TO JAVASCRIPTA



Why use .NET for browser apps?

Stable, mature, productive:

.NET Standard, MSBuild

Fast, scalable, reliable:

.NET Core for backend services

Modern languages:

Innovations in C#, F#, Razor

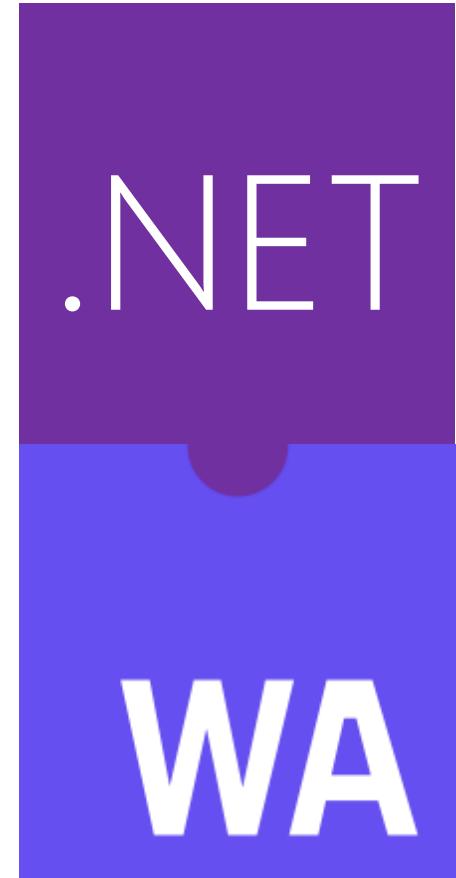
First-rate dev tools:

Visual Studio, IntelliSense

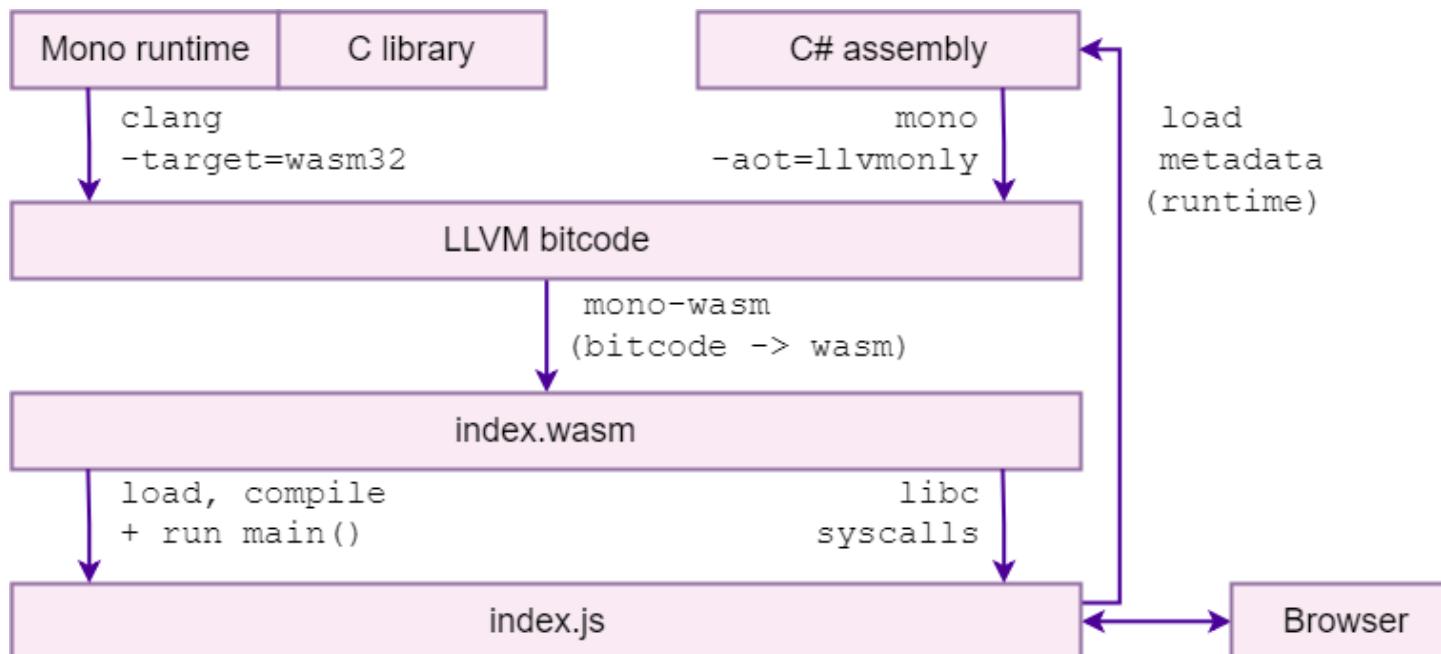
Solid foundation to build on

Leverage existing skills and knowledge

Because We can



C# in wasm ? Mono !



<https://github.com/migueldeicaza/mono-wasm>

https://www_mono-project.com/news/2018/09/11/csharp-jit/

<https://twitter.com/migueldeicaza/status/1039639597435641856>



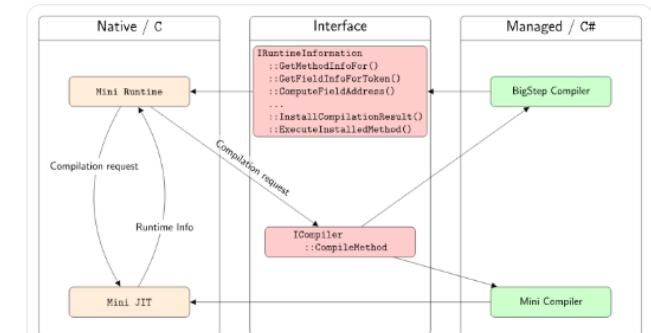
Miguel de Icaza
@migueldeicaza

Following

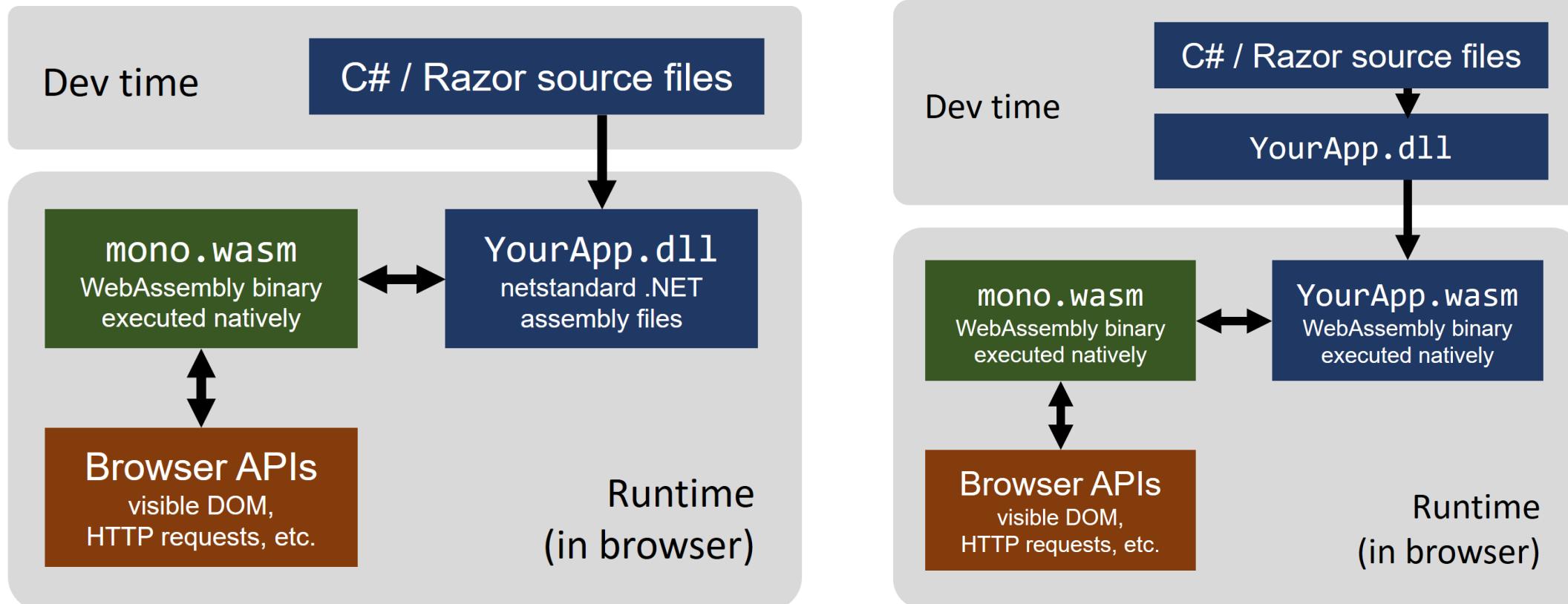
Mono now allows the code generation engine (for JIT and AOT) to be replaced. Not only replaced, but replaced with C# or F# code!

No more low-level, error-prone, unsafe C code!

Details here: mono-project.com/news/2018/09/11/csharp-jit/ ...



Interpreter mode vs AOT compiled



XAML for desktop app



XAML standard is ready

XAML for desktop app



XAML standard is ready and dead

XAML for desktop app

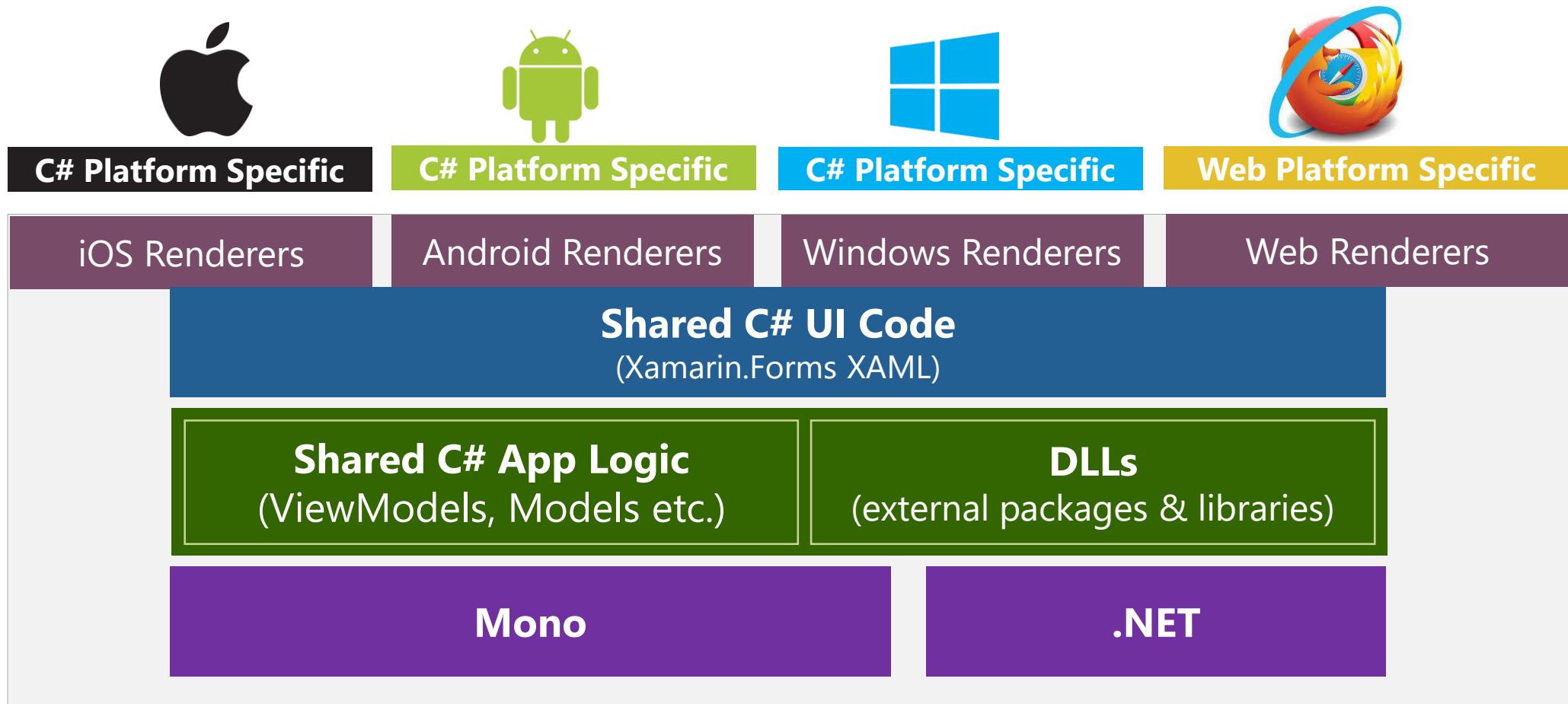


XAML standard is ready and dead

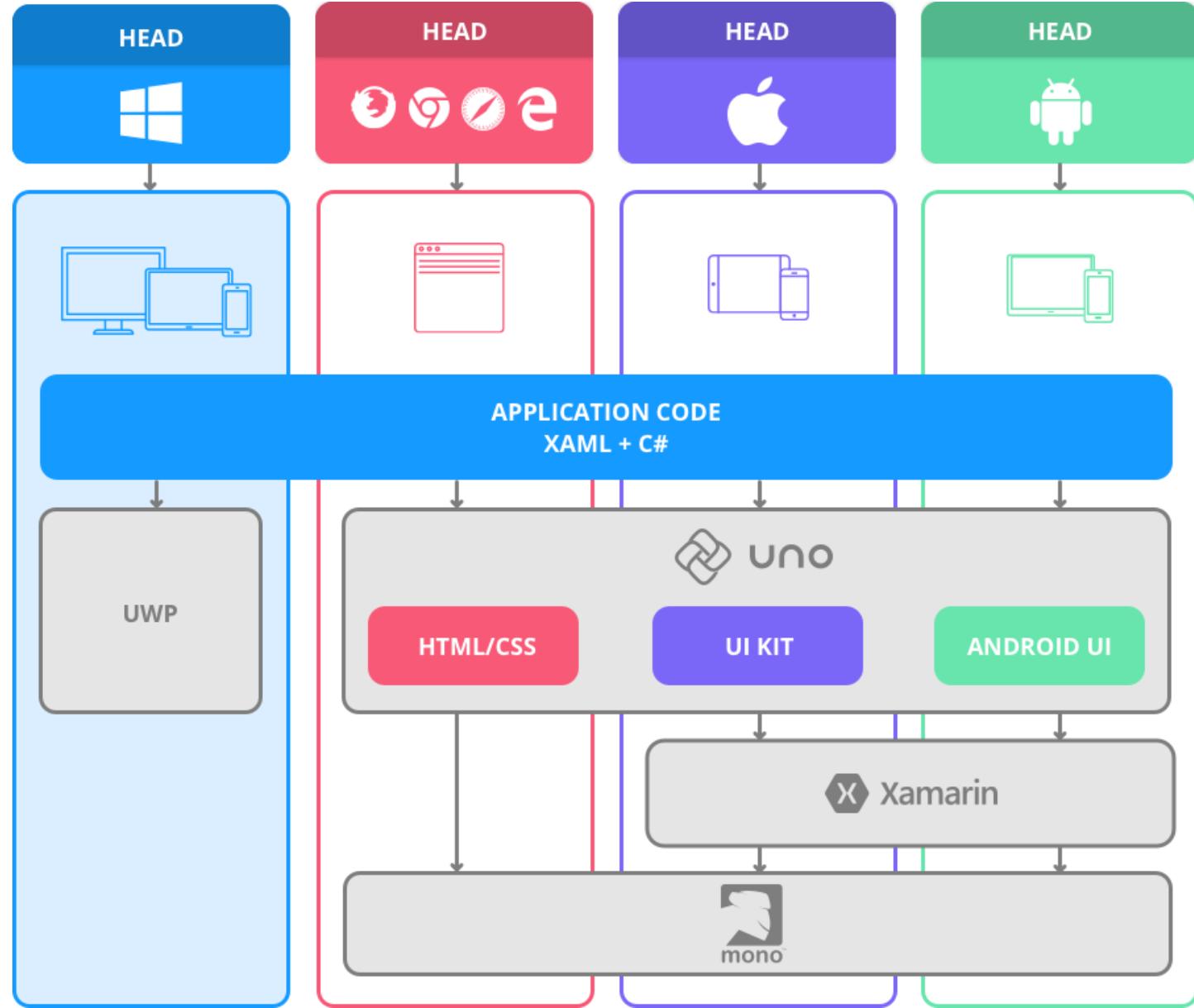
- Xamarin.Forms
- UWP
- Silverlight
- WPF

}

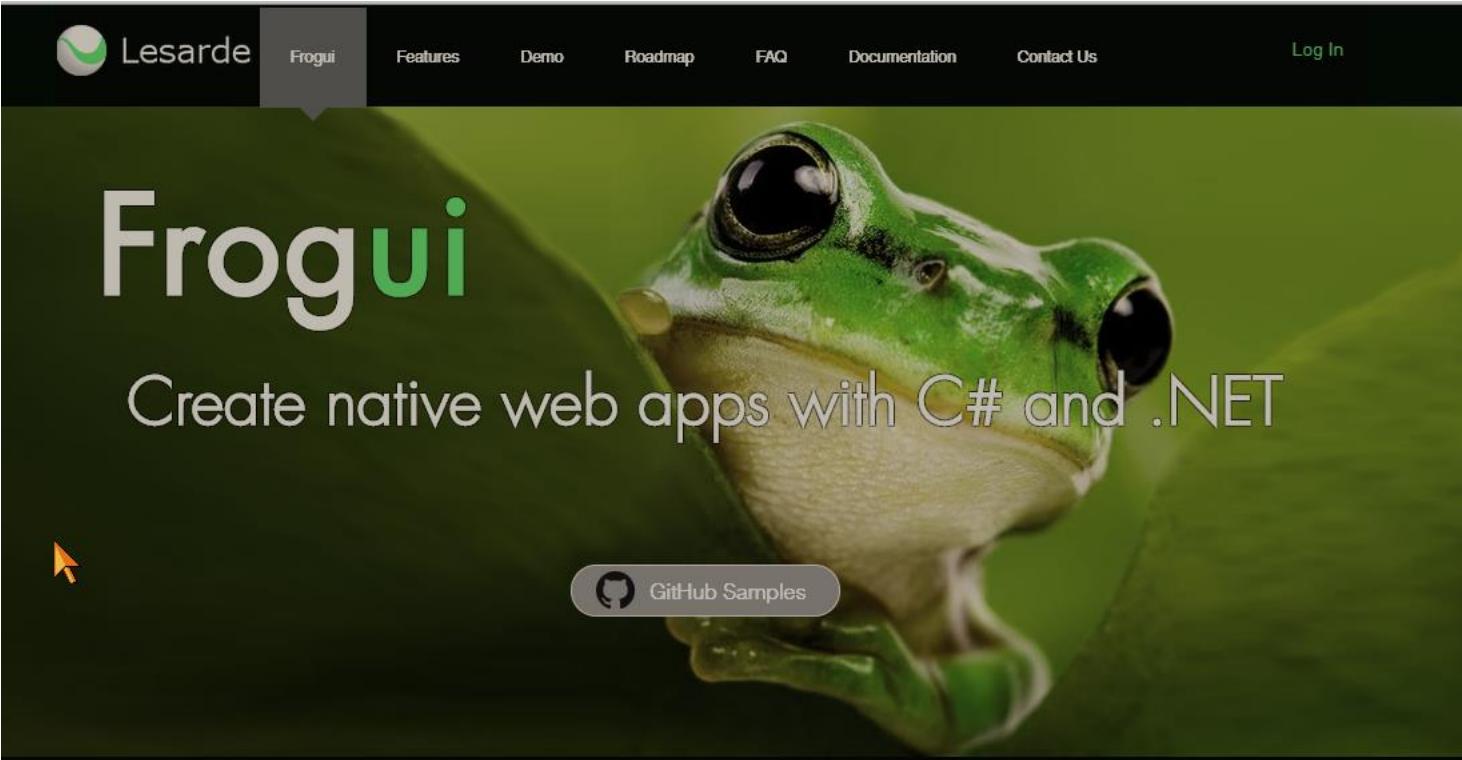
Ooui



UNO



FrogUI



The screenshot shows the homepage of the FrogUI website. At the top, there's a dark navigation bar with the 'Lesarde' logo, a 'Frogui' button (which is highlighted in grey), and links for 'Features', 'Demo', 'Roadmap', 'FAQ', 'Documentation', 'Contact Us', and 'Log In'. The main content area features a large, close-up image of a green tree frog. Overlaid on the left side of the image is the word 'Frogui' in a large, white, sans-serif font. Below it, the text 'Create native web apps with C# and .NET' is displayed in a smaller, white, sans-serif font. In the bottom-left corner of the image, there's a small orange cursor icon pointing towards a 'GitHub Samples' button. This button has a dark grey rounded rectangle with a white outline, containing a GitHub logo icon and the text 'GitHub Samples'. The bottom portion of the page has a black background with white text. It starts with a paragraph about Frogui being a UI framework for building native client web apps using C#, .NET, and Visual Studio. It then says the API is based on WPF and Silverlight but modernized for modern browsers.

Frogui is a revolutionary UI framework that allows developers familiar with C#, .NET and Visual Studio to build *native* client web apps using their existing skills, with no need to learn or use JavaScript or any other web technology.

The API is strongly based on WPF and Silverlight so will feel immediately familiar, but has been modernized to leverage the immense power of modern browsers.

<https://www.lesarde.com/>



Blazor? Razor components ?

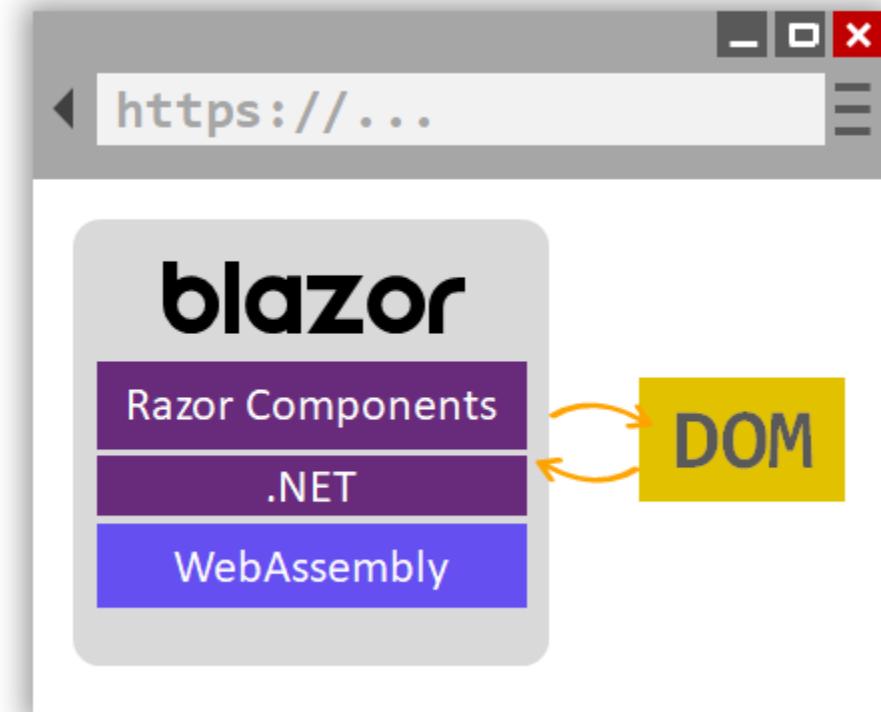
Blazor

For information about using Blazor, see [blazor.net](#).

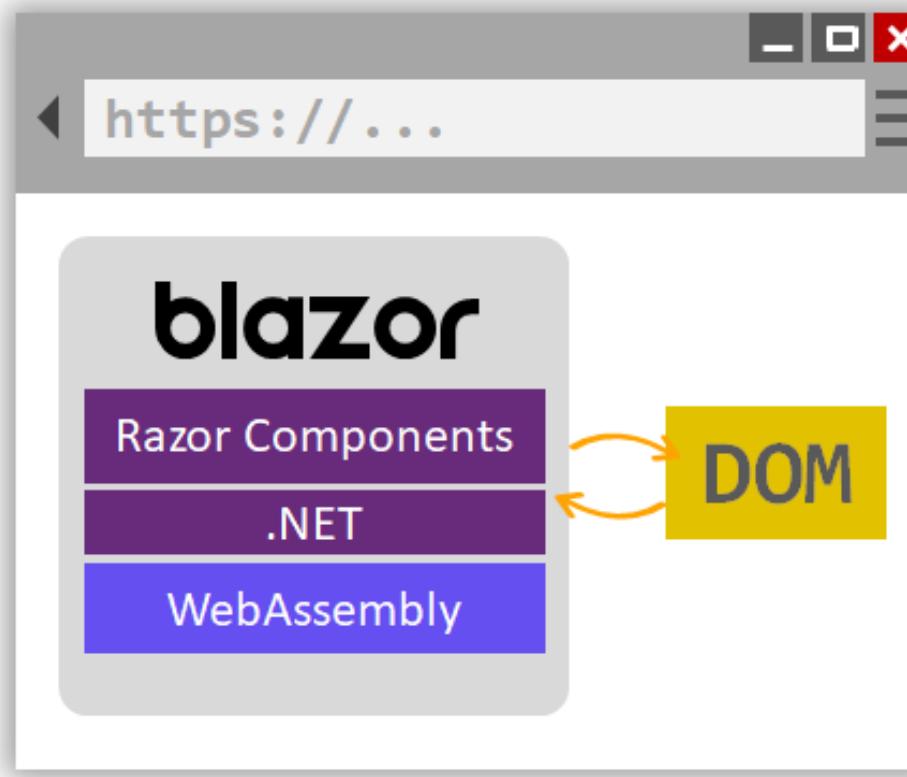
Sources moved

Almost all the sources for Blazor and the Razor Components programming model have moved [here](#) in the central ASP.NET Core repo. We are also in the process of migrating open issues from here to there.

This is in preparation for shipping Razor Components as a built-in feature of ASP.NET Core 3.0. Note: *client-side Blazor remains experimental while we continue to work on making the WebAssembly runtime complete.*



Blazor? Razor components? It's complicated



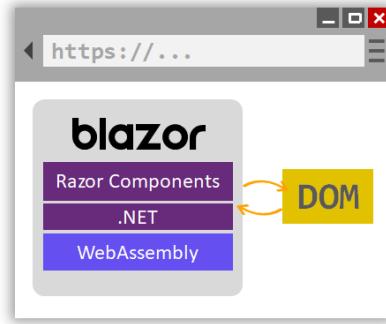
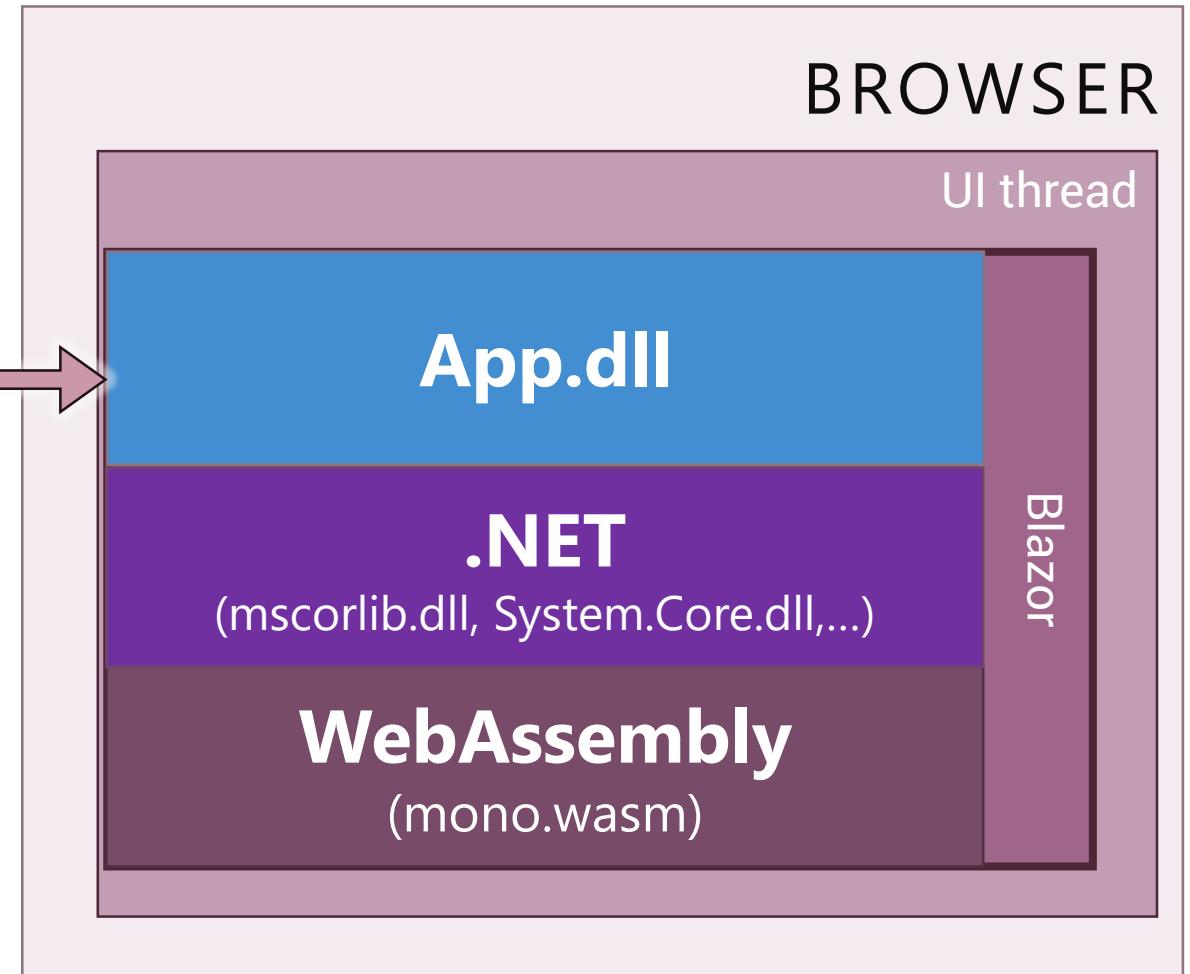
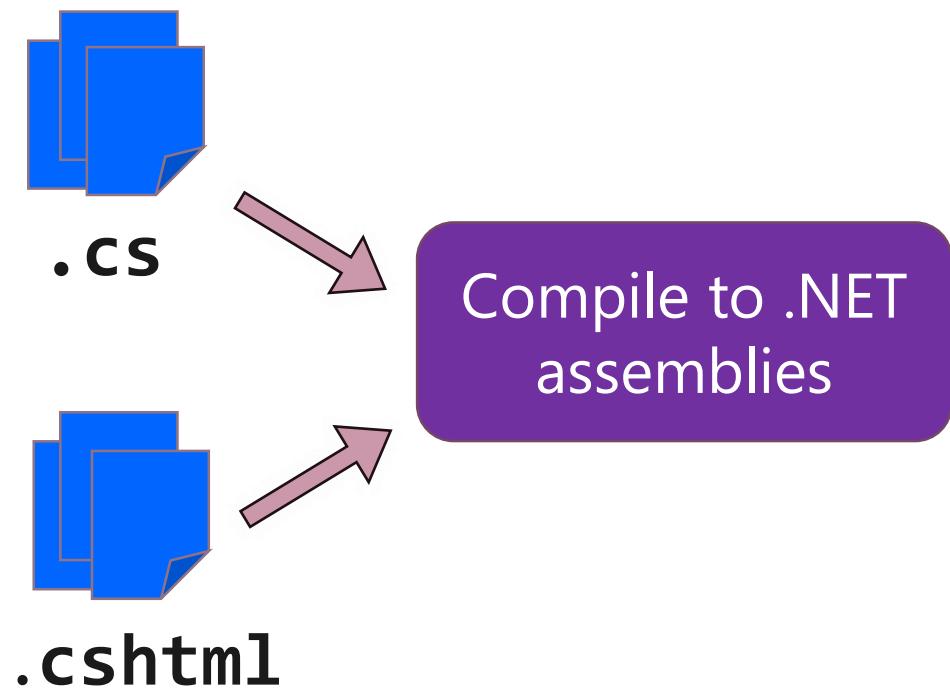
This is in preparation for shipping Razor Components as a built-in feature of ASP.NET Core 3.0. Note: client-side Blazor remains experimental while we continue to work on making the WebAssembly runtime complete.

<https://github.com/aspnet/AspNetCore/issues/8931>

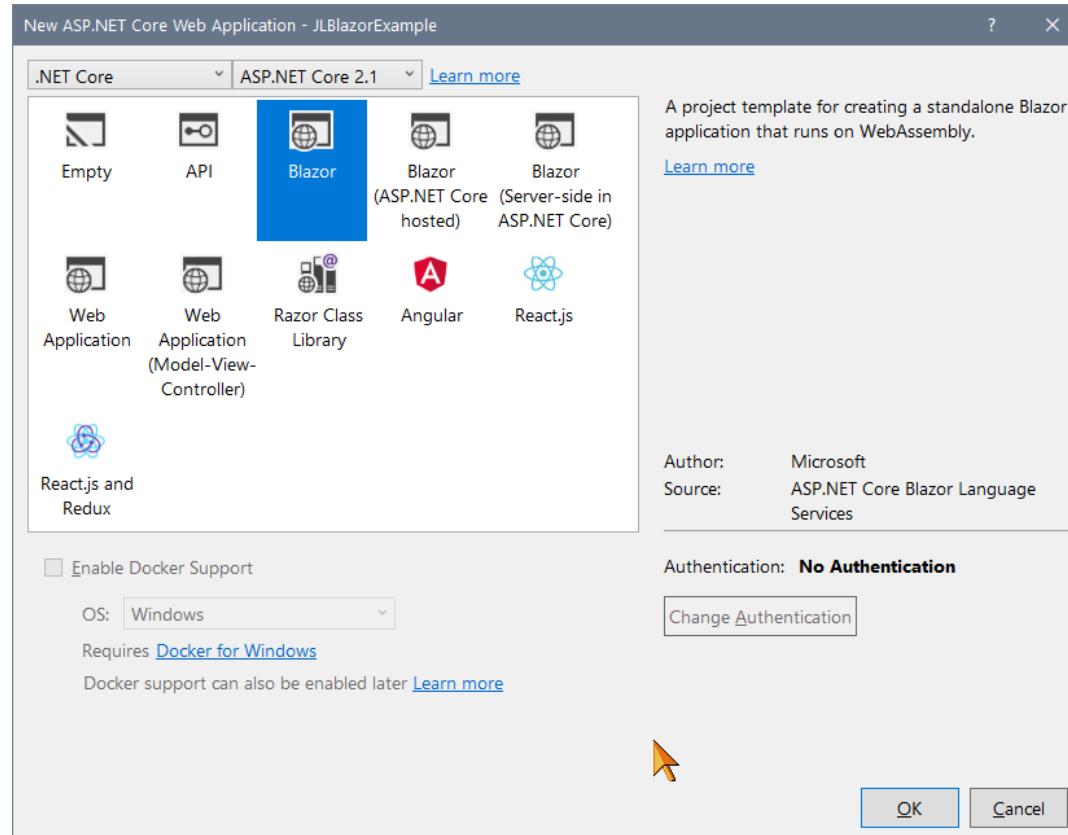
Blazor new project

Blazor (hosted in ASP.NET server)	blazorhosted	[C#]
Blazor Library	blazorlib	[C#]
Blazor (standalone)	blazor	[C#]
Razor Class Library	razorclasslib	[C#]
ASP.NET Core Web App (Razor Components)	razorcomponents	[C#]

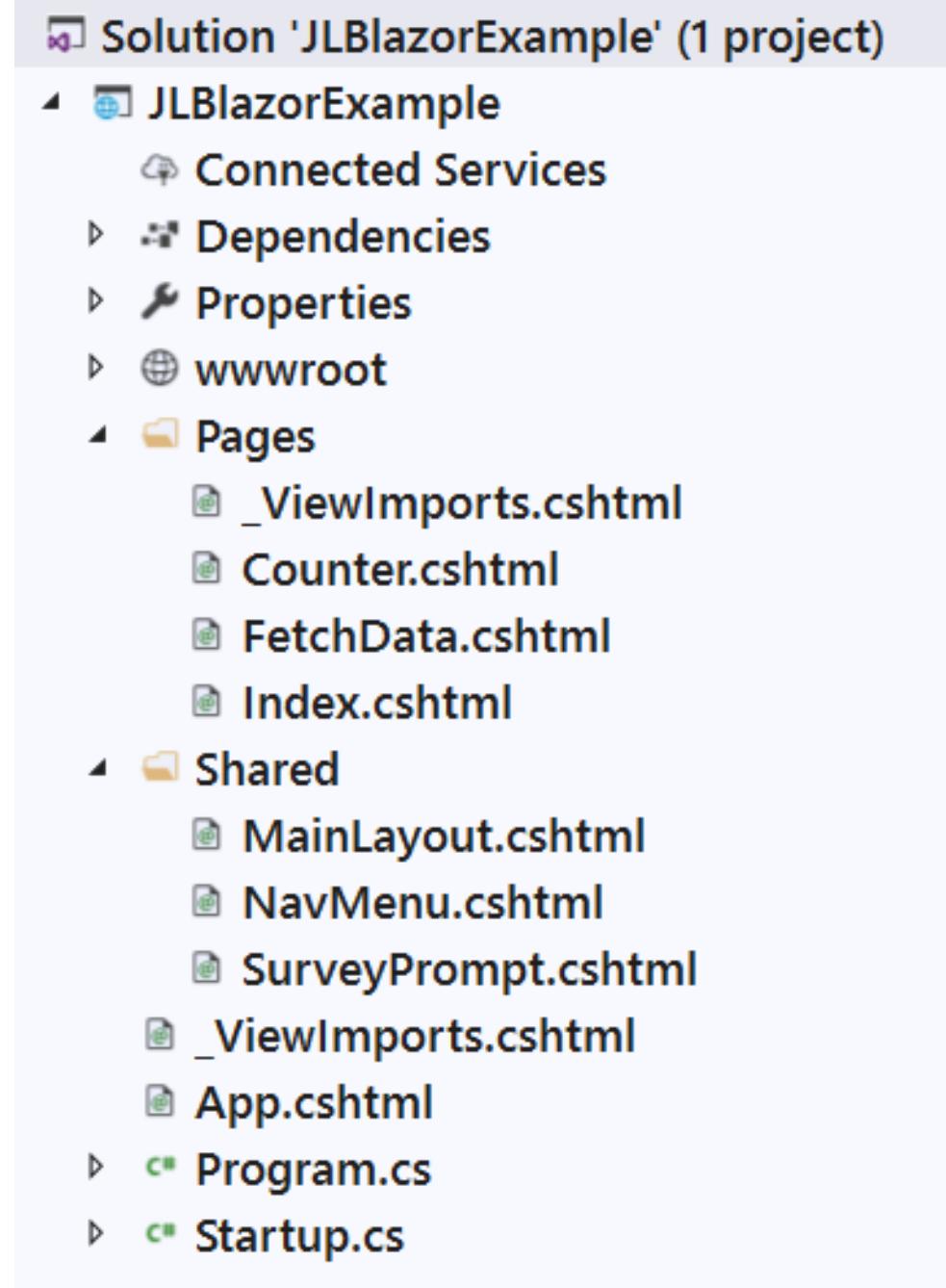
Blazor client-side



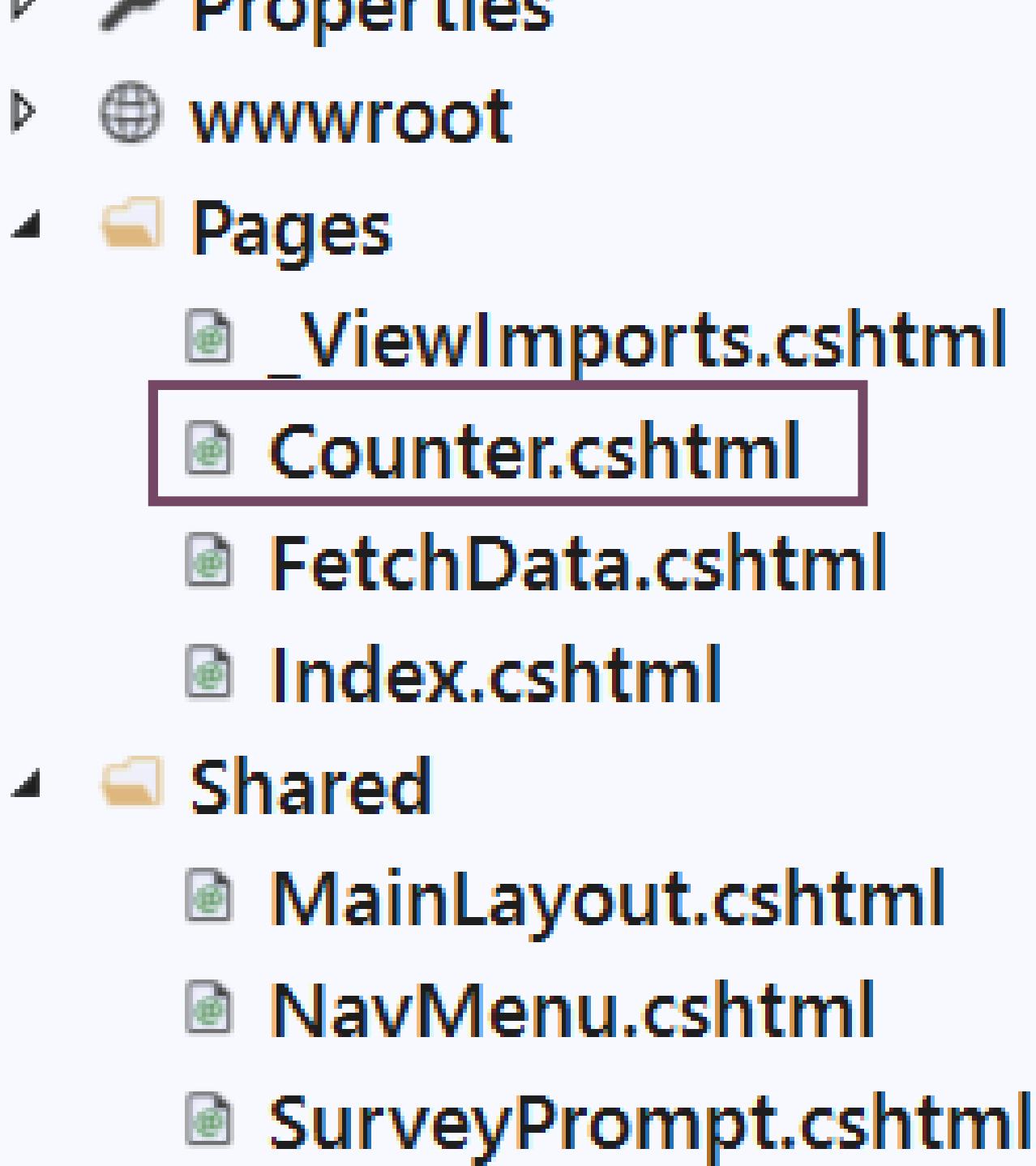
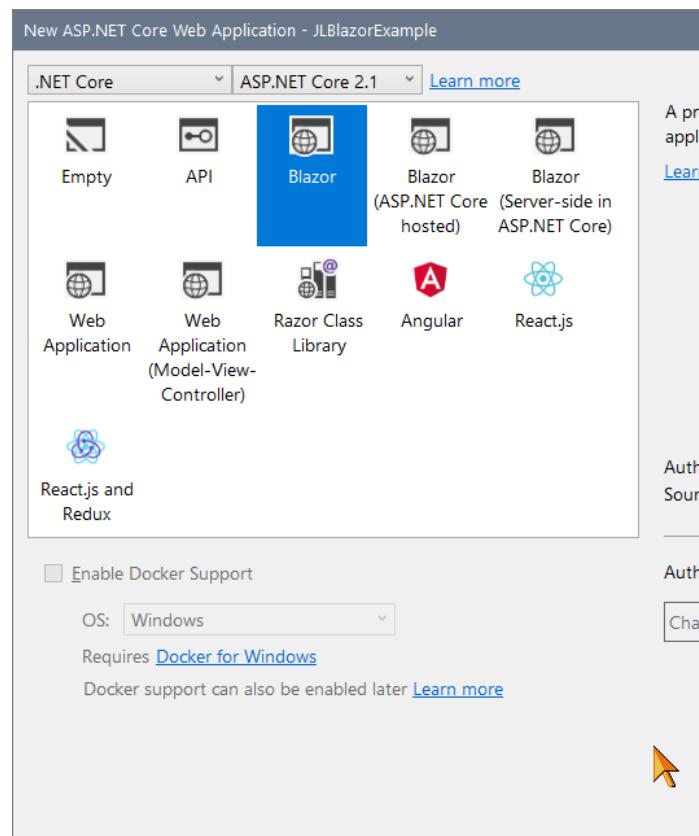
Blazor client-side



```
C:\Projects> dotnet new blazor
```



Blazor client





JLBlazorExample

Home

Counter

Fetch data

Hello, world!

Welcome to your new app.



How is Blazor working for you? Please take our [brief survey](#) and tell us what you think.

Blazor: Sample

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor: Routing

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor: Razor template

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor: C#

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor: One way data binding

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor: Two way data binding

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor: Event binding

```
@page "/counter"
@inject HttpClient

<h1>Counter</h1>
<input type="number" bind="IncrementAmount" />
<p>Current count: @currentCount</p>
<button class="btn btn-primary" onclick="@IncrementCount">Click me</button>

@functions {
    int currentCount = 0;
    int IncrementAmount { get; set; } = 1;

    void IncrementCount()
    {
        currentCount += IncrementAmount;
    }
}
```

Blazor sample in browser

The screenshot shows a browser window titled "JLBlazorExample" at the URL "localhost:64814". The page displays a "Hello, world!" message and a navigation menu with "Home" and "Counter" options. Below the page content is a Network tab from a developer tools interface, showing a list of requests and their details. The table includes columns for Stan (Status), Metoda (Method), Domena (Domain), Plik (File), Przycz... (Reason), Typ (Type), Przesłano (Sent), Rozmiar (Size), and various timing metrics. The requests listed include files like "/bootstrap.min.css", "site.css", "blazor.webassembly.js", and various DLL files. The "Sieć" (Network) tab is selected in the tabs bar above the table.

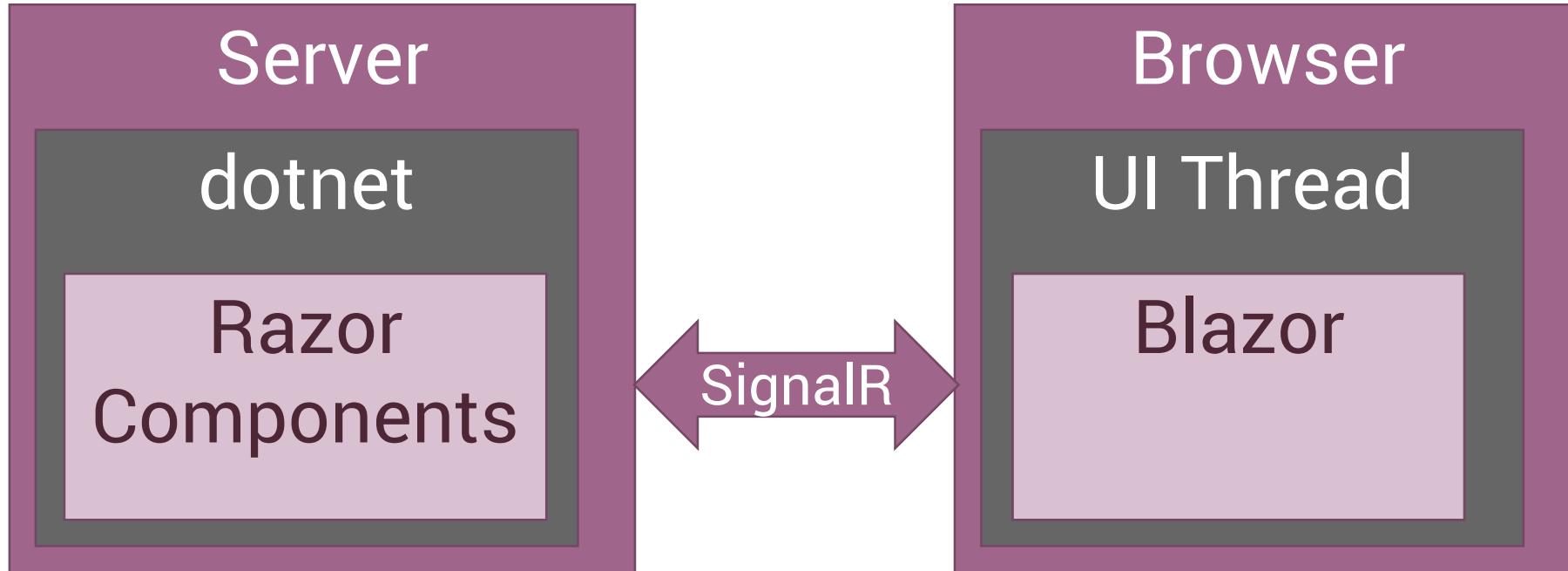
Stan	Metoda	Domena	Plik	Przycz...	Typ	Przesłano	Rozmiar	0 ms	320 ms	640 ms	960 ms	1,28 s	1,61 ^
304	GET	localhost:64814	/		document	html	0	426 B	5 ms				
304	GET	localhost:64814	bootstrap.min.css		stylesheet	css	0	141,49 KB	6 ms				
304	GET	localhost:64814	site.css		stylesheet	css	0	2,20 KB	7 ms				
304	GET	localhost:64814	blazor.webassembly.js		script	js	0	0 B	6 ms				
304	GET	localhost:64814	open-iconic-bootstrap.min.css		stylesheet	css	0	9,17 KB	4 ms				
304	GET	localhost:64814	blazor.boot.json		fetch	json	0	590 B	3 ms				
200	GET	localhost:64814	favicon.ico		img	html	0	426 B					
304	GET	localhost:64814	mono.js		script	js	0	0 B	3 ms				
304	GET	localhost:64814	mono.wasm		fetch	wasm	0	1,83 MB	3 ms				
200	GET	localhost:64814	JLBlazorExample.dll		xhr	octet-stream	7,19 KB	16 KB		6 ms			
304	GET	localhost:64814	JL.GameOfLife.Core.dll		xhr	octet-stream	0	12 KB		6 ms			
200	GET	localhost:64814	Microsoft.AspNetCore.Blazor.Browser.dll		xhr	octet-stream	14,76 KB	28,50 KB		7 ms			
200	GET	localhost:64814	Microsoft.AspNetCore.Blazor.dll		xhr	octet-stream	40,32 KB	91 KB		10 ms			
200	GET	localhost:64814	Microsoft.AspNetCore.Blazor.TagHelper... .xhtml		xhr	octet-stream	2,42 KB	5 KB		4 ms			
200	GET	localhost:64814	Microsoft.Extensions.DependencyInjection.Injecti... .xhtml		xhr	octet-stream	11,89 KB	26,50 KB		7 ms			
200	GET	localhost:64814	Microsoft.Extensions.DependencyInjection.Injecti... .xhtml		xhr	octet-stream	20,44 KB	42 KB		7 ms			
200	GET	localhost:64814	System.Net.Http.dll		xhr	octet-stream	31,82 KB	65,50 KB		8 ms			
304	GET	localhost:64814	JL.GameOfLife.Core.pdb		xhr	octet-stream	0	3,34 KB		4 ms			
200	GET	localhost:64814	JLBlazorExample.pdb		xhr	octet-stream	1,87 KB	2,36 KB					
200	GET	localhost:64814	Microsoft.JSInterop.dll		xhr	octet-stream	20,67 KB	40,50 KB					
200	GET	localhost:64814	Mono.WebAssembly.Interop.dll		xhr	octet-stream	3,15 KB	6 KB					
304	GET	localhost:64814	mscorlib.dll		xhr	octet-stream	0	1,56 MB					
200	GET	localhost:64814	System.Core.dll		xhr	octet-stream	137,22 KB	334 KB					

24 żądań | Przesłano: 4,29 MB / 291,75 KB | 1,31 s | DOMContentLoaded: 239 ms | load: 244 ms

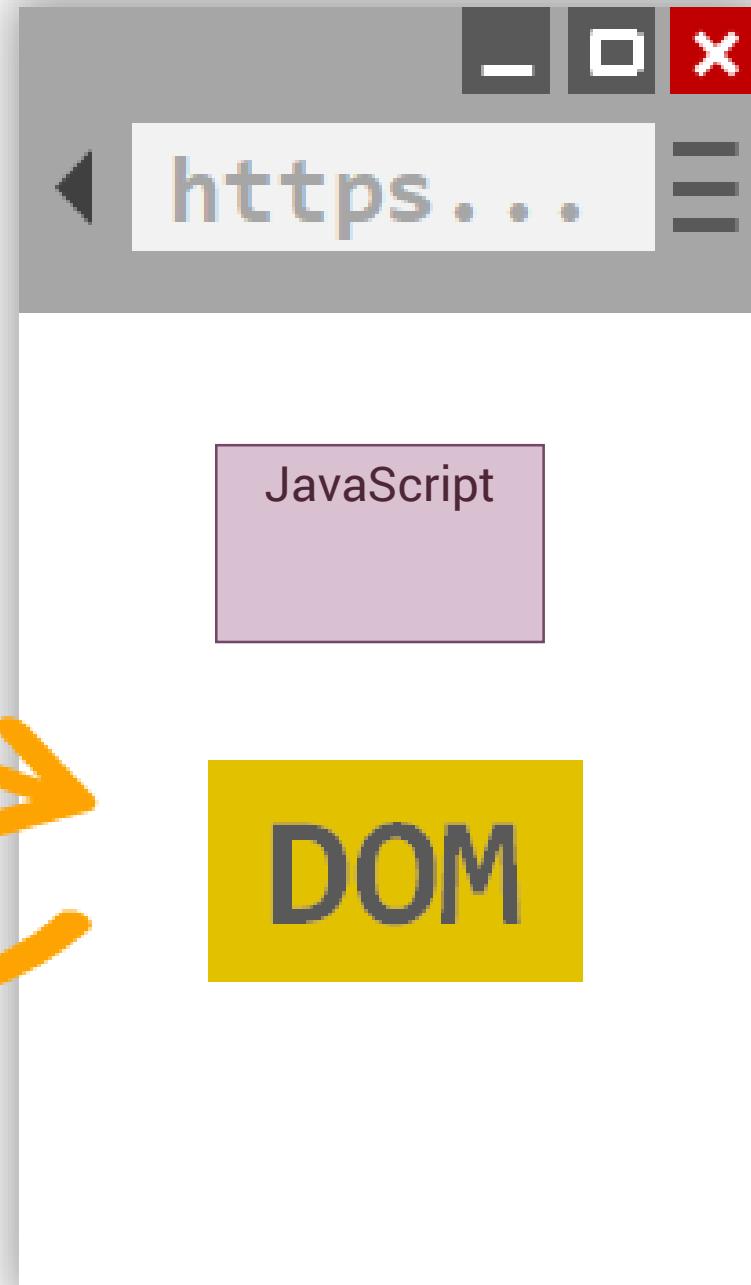
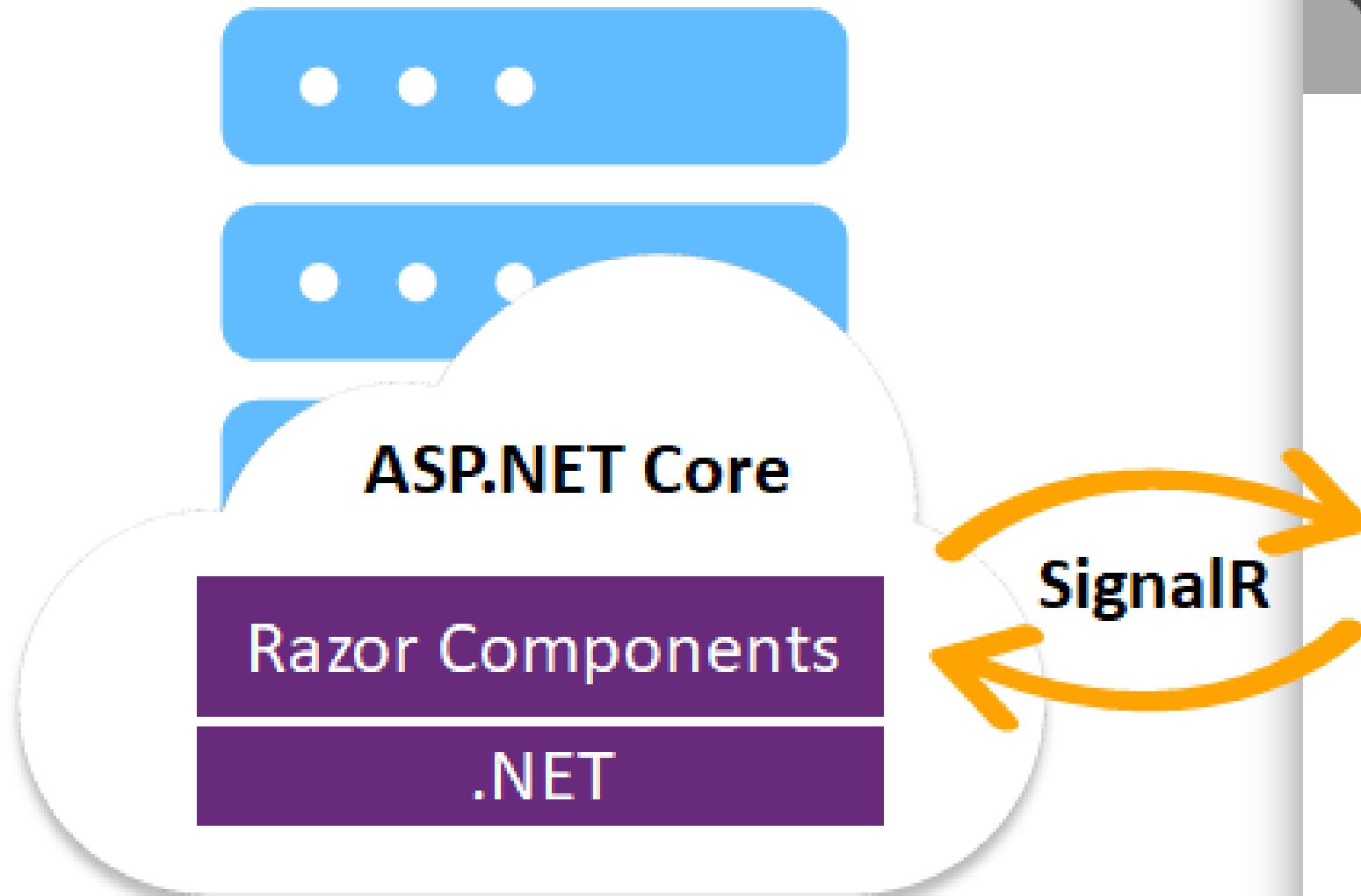


 bootstrap.min.css	stylesheet	css	0	141,49 KB
 site.css	stylesheet	css	0	2,20 KB
 blazor.webassembly.js	script	js	0	0 B
 open-iconic-bootstrap.min.css	stylesheet	css	0	9,17 KB
 blazor.boot.json	fetch	json	0	590 B
 favicon.ico	img	html	0	426 B
 mono.js	script	js	0	0 B
 mono.wasm	fetch	wasm	0	1,83 MB
 JLBlazorExample.dll	xhr	octet-stream	7,19 KB	16 KB
 JL.GameOfLife.Core.dll	xhr	octet-stream	0	12 KB
 Microsoft.AspNetCore.Blazor.Browser.dll	xhr	octet-stream	14,76 KB	28,50 KB
 Microsoft.AspNetCore.Blazor.dll	xhr	octet-stream	40,32 KB	91 KB
 Microsoft.AspNetCore.Blazor.TagHelper...	xhr	octet-stream	2,42 KB	5 KB

Let's switch elements places

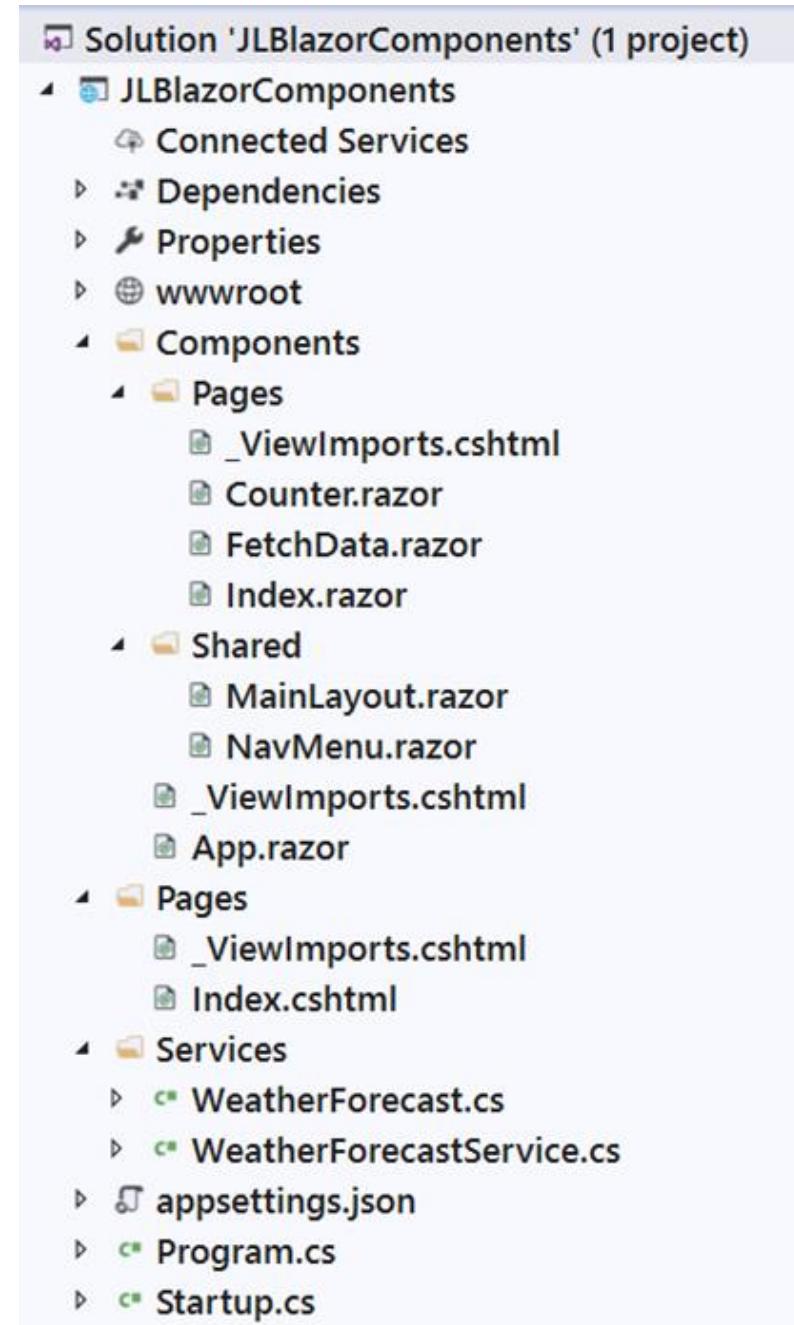


Blazor server-side

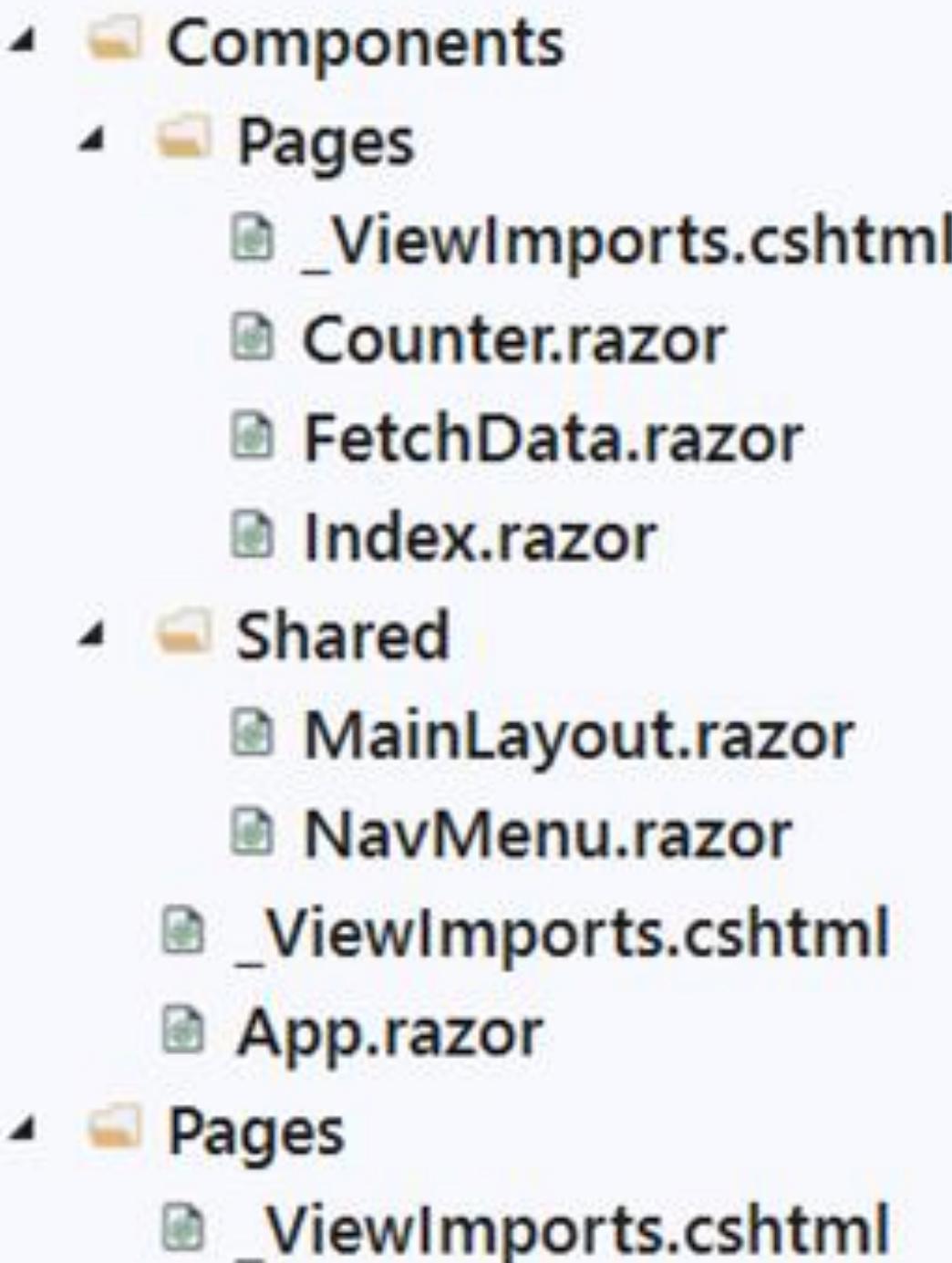


Blazor server-side

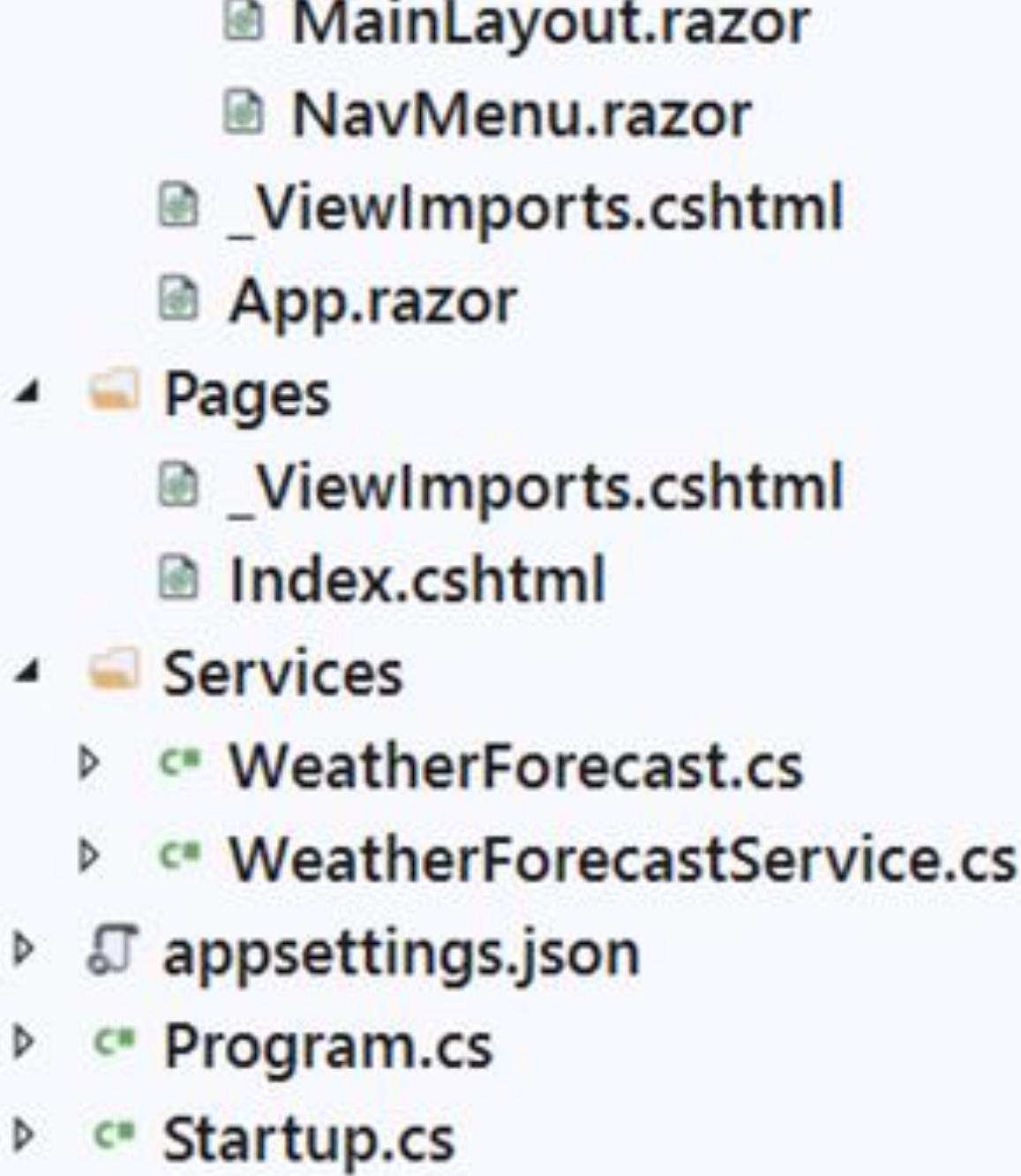
```
C:\Projects> dotnet new  
razorcomponents -n  
JLBlazorComponents
```



Blazor server-side

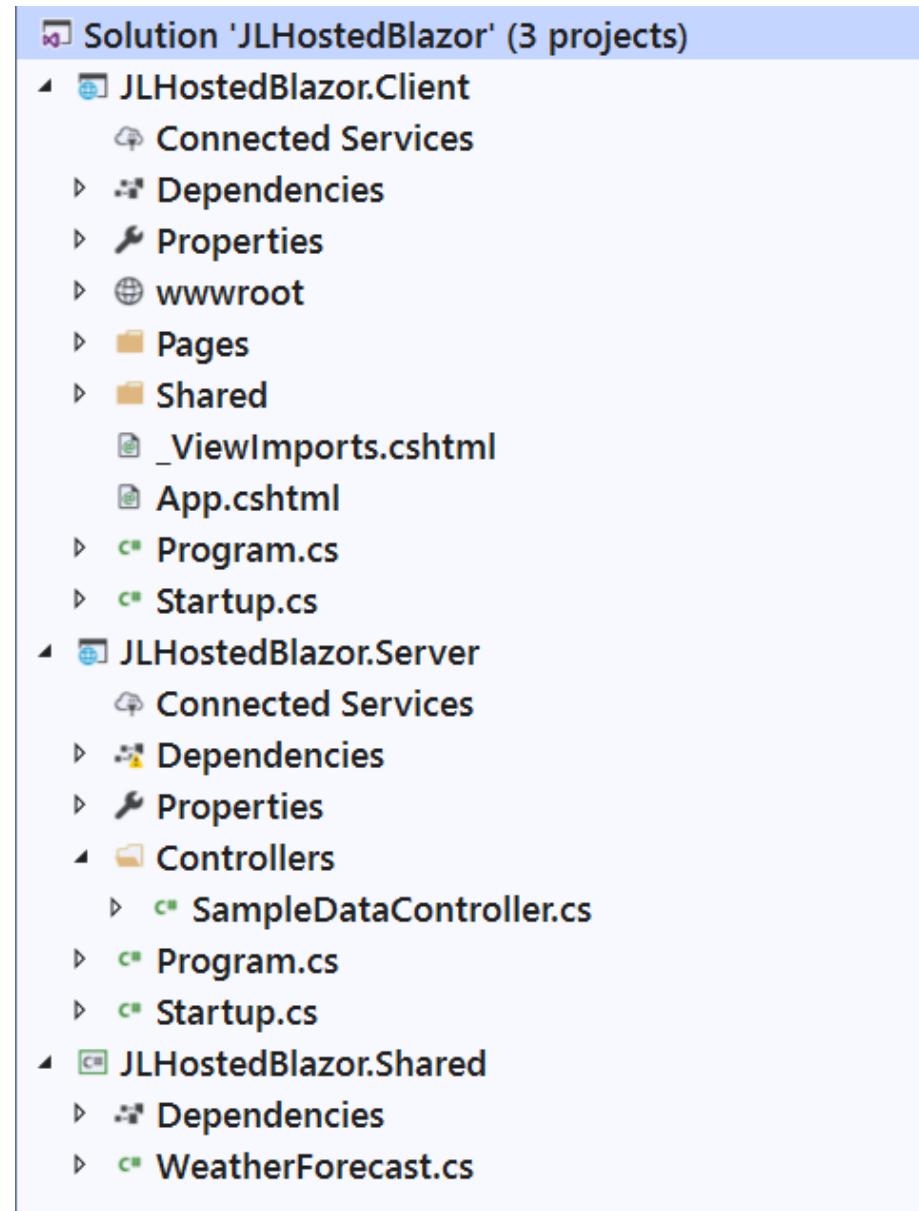


Blazor server-side

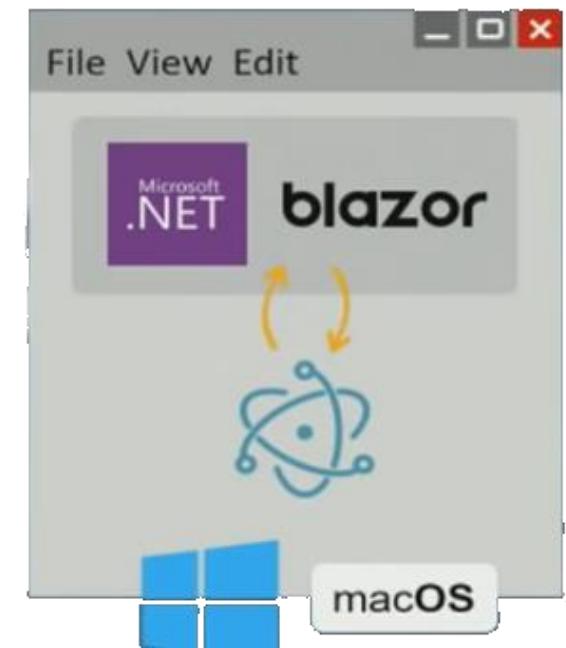
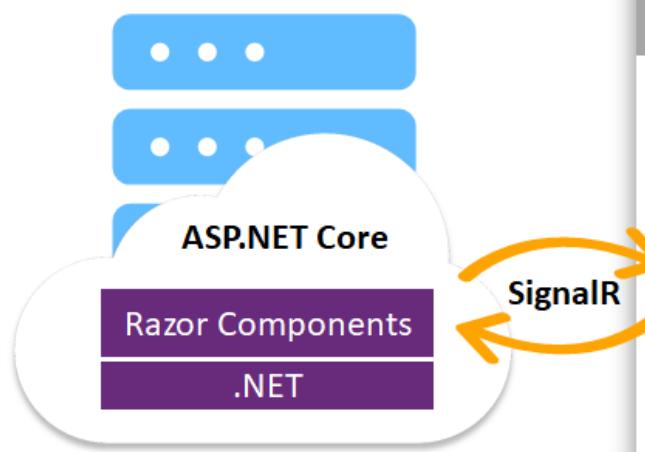
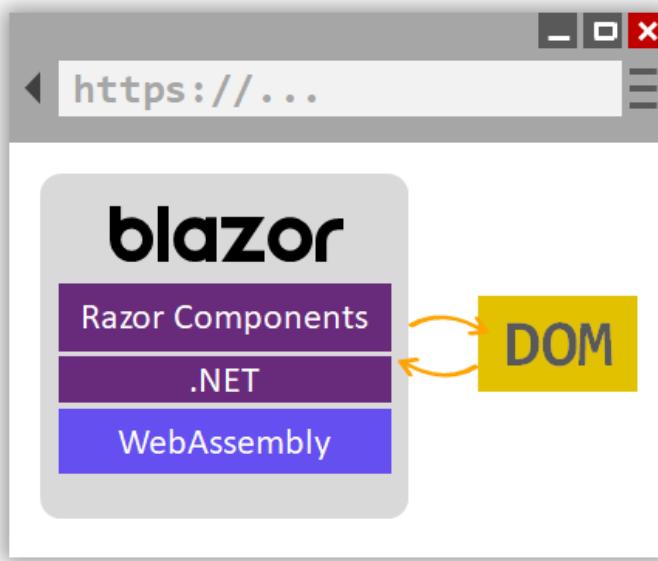


Blazor hosted

```
C:\Projects> dotnet new  
blazorhosted -n JLHostedBlazor
```



Blazor: possibilities



What's the future?



What WebAssembly are next to?

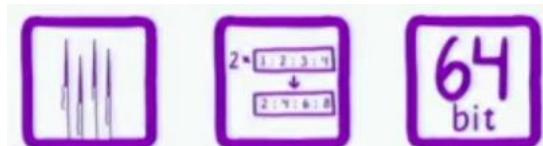
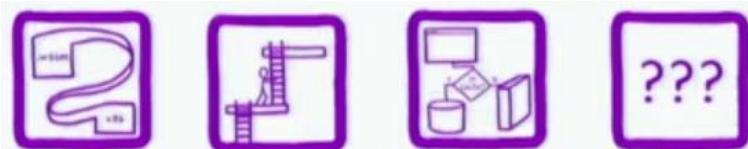


Loadtime improvements

- streaming compilation
- tiered compilation
- implicit HTTP caching
- other improvements

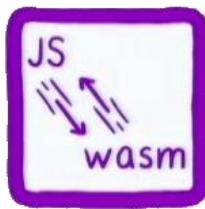
Making use of modern Hardware

- threading
- SIMD
- wasm 64bit



What WebAssembly are next to?

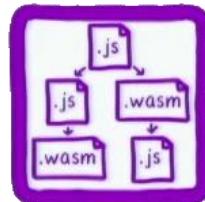
- fast calls
JS <~> wasm



- easy and fast
data exchange



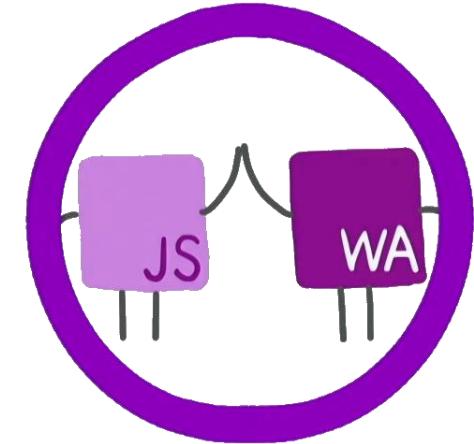
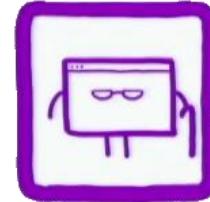
- ES module
integration



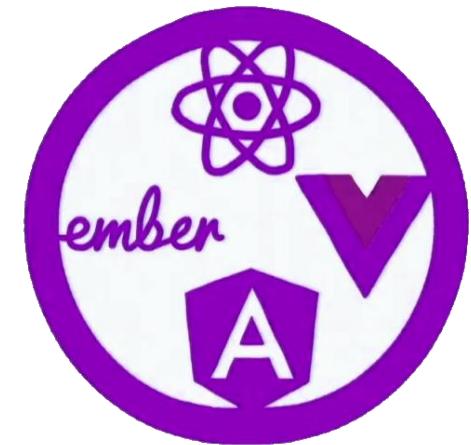
- toolchain integration
like npm or webpack



- backward compatibility

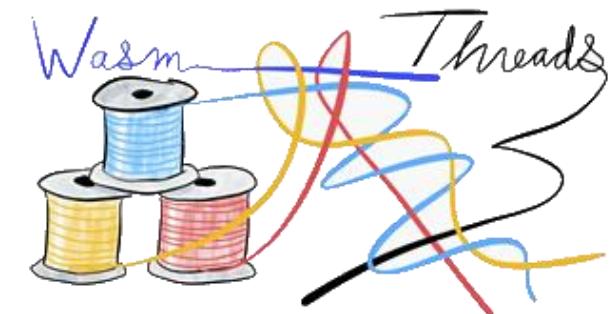
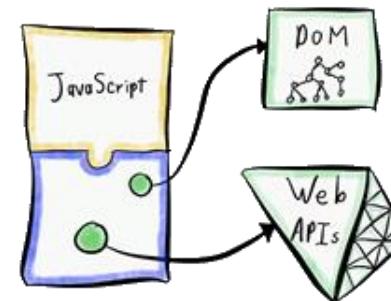


What WebAssembly are next to?

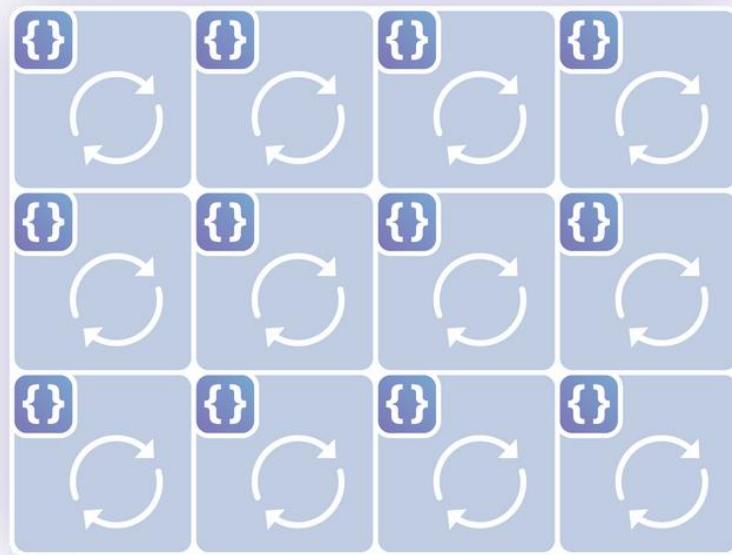


High level language features

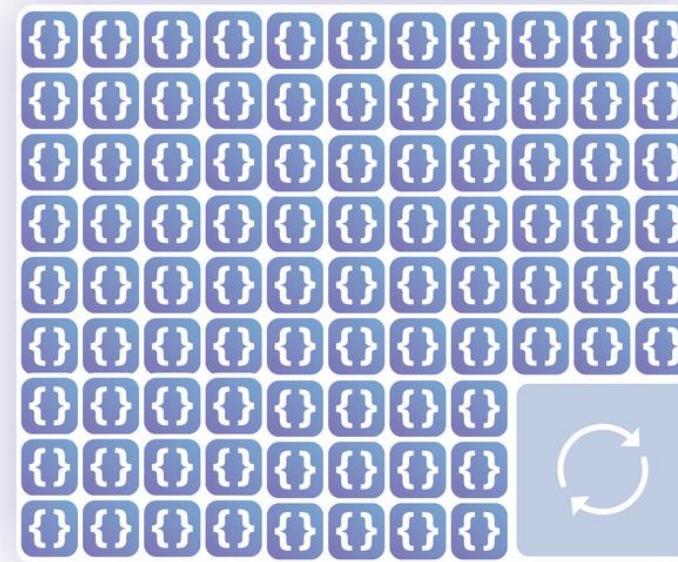
- GC integration
- Exception handling
- Debugging
- Tail calls



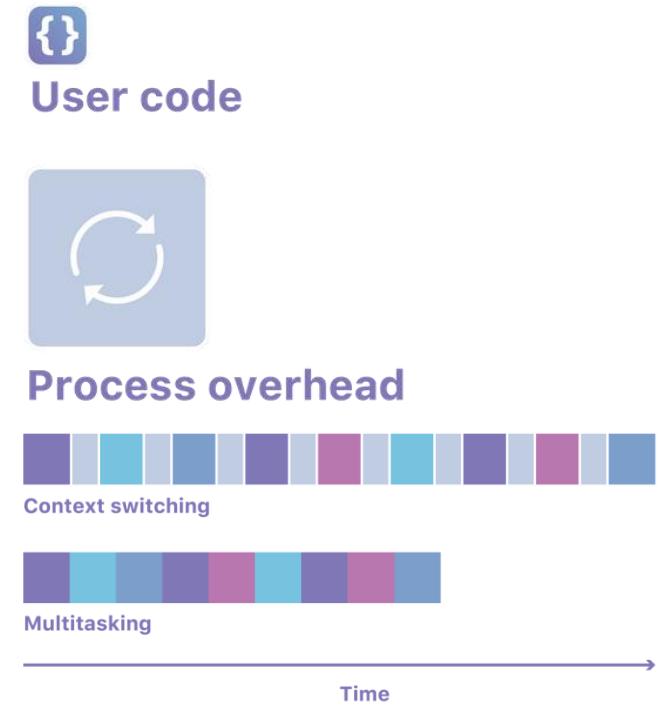
CloudFlare



Virtual machine



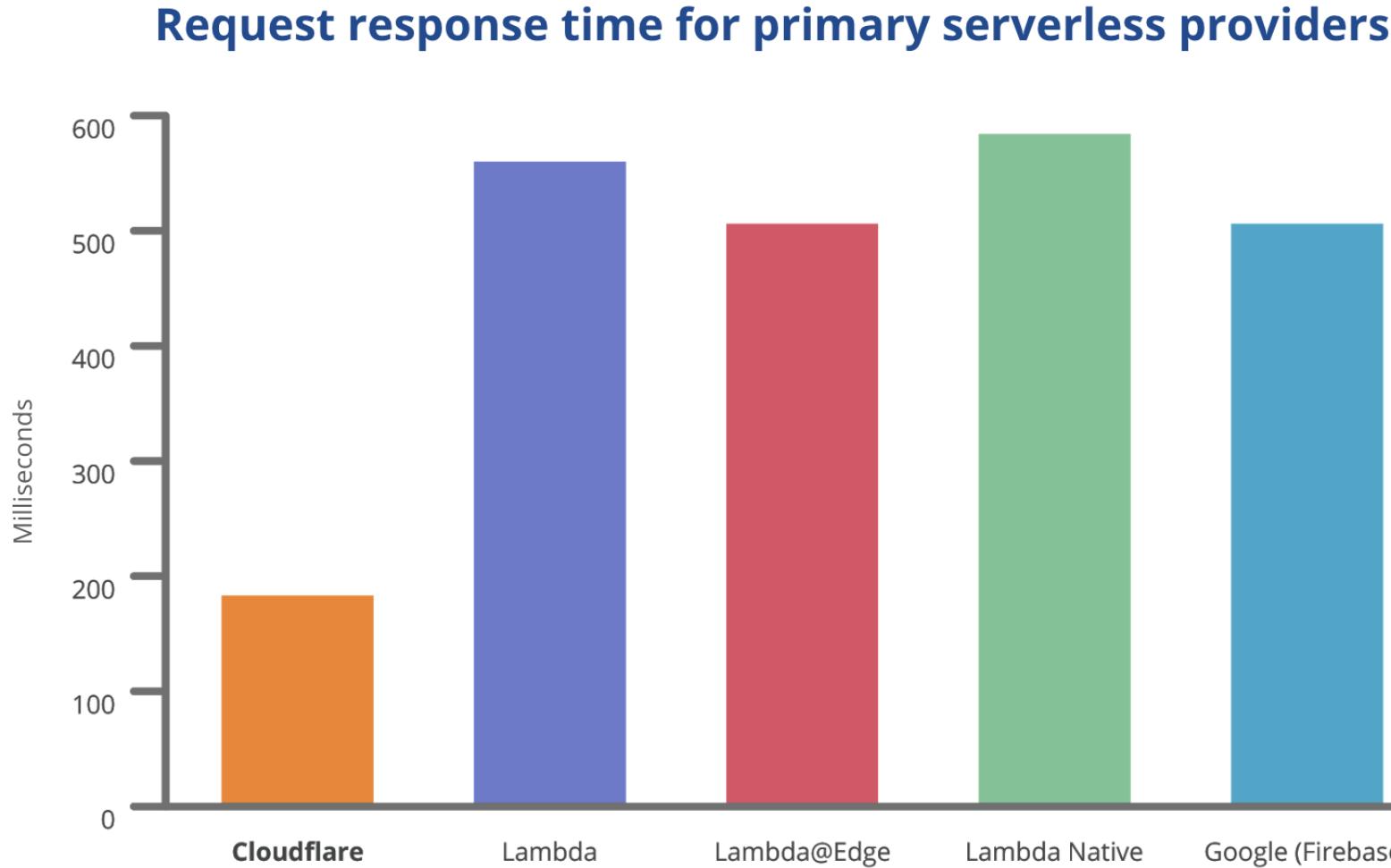
Isolate model



<https://blog.cloudflare.com/cloud-computing-without-containers/>

<https://www.youtube.com/watch?v=A9SydP1CcZU>

CloudFlare no more cold starts?



So WebAssembly?

Possibilities

Flexibility
Diversity

Cross-platform

Promise of bright tomorrow for todays projects

Questions??



Thank you ☺

Joanna Lamch

JLamch@gmail.com

JLamch.net

ProgramistkaKot.pl



Działanie pociąga za sobą koszty i ryzyko, ale o wiele mniejsze niż te, które wiążą się z wygodną bezczynnością



• Bibliografia

- <https://webassembly.org/docs/high-level-goals/>
- <https://www.smashingmagazine.com/2017/05/abridged-cartoon-introduction-webassembly/>
- https://www.youtube.com/watch?v=HktWin_LPf4&feature=youtu.be
- <https://www.youtube.com/watch?v=pBYqen3B2gc>
- <https://www.youtube.com/watch?v=BnYq7JapeDA>
<https://www.youtube.com/watch?v=kS29TT4wk44&feature=youtu.be>
- <https://github.com/mbasso/awesome-wasm>
- <https://github.com/migueldeicaza/mono-wasm>
- <https://superkotlin.com/kotlin-and-webassembly/>
- <https://medium.com/@mumarov/how-to-get-started-with-kotlin-native-and-web-assembly-baa2813f0d9>
- <https://github.com/DenisKolodin/yew>
- <https://www.mergeconflict.fm/89>
- <https://dotnetrocks.com/?show=1539>
- <https://dotnetrocks.com/?show=1540>
- <https://dotnetrocks.com/?show=1537>
- <https://www.hanselman.com/blog/NETAndWebAssemblyIsThisTheFutureOfTheFrontend.aspx>
<https://hacks.mozilla.org/2018/04/sneak-peek-at-webassembly-studio/>
- https://github.com/migueldeicaza/mono-wasm?WT.mc_id=-blog-scottha
- <https://blog.scottlogic.com/ceberhardt/>
- <https://blog.logrocket.com/working-with-the-blazor-javascript-interop-3c2a8d0eb56c>
<https://s3.amazonaws.com.mozilla-games/ZenGarden/EpicZenGarden.html>
<https://blog.logrocket.com/working-with-the-blazor-javascript-interop-3c2a8d0eb56c>



- Ankieta

Działanie pociąga za sobą koszty i ryzyko, ale o wiele mniejsze niż te, które wiążą się z wygodną bezczynnością.

- Będzie mi miło móc ulepszyć moją prezentację dzięki Twoim komentarzom, dlatego proszę Cię o wypełnienie ankiety, bądź kontakt mailowy.

