



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS | San Diego

# Assembling the Web – A Tour of WebAssembly

**Jason Bock**  
Practice Lead  
Magenic

Level: Intermediate

Code Again for the First Time! 

Visual Studio **25**  
YEARS OF CODING INNOVATION

## Personal Info

- <http://www.magenic.com>
- <http://www.jasonbock.net>
- <https://www.twitter.com/jasonbock>
- <https://www.github.com/jasonbock>
- [jasonb@magenic.com](mailto:jasonb@magenic.com)



# Downloads

<https://github.com/JasonBock/Presentations/blob/master/Assembling%20the%20Web%20-%20A%20Tour%20of%20WebAssembly.pptx>

<https://github.com/JasonBock/AssemblingTheWeb>



# Overview

- History
- Details
- Conclusion

Remember....

<https://github.com/JasonBock/Presentations/blob/master/Assembling%20the%20Web%20-%20A%20Tour%20of%20WebAssembly.pptx>

<https://github.com/JasonBock/AssemblingTheWeb>



# History



[http://foodnetwork.sndimg.com/content/dam/images/food/fullset/2008/3/5/0/NY0100\\_BBQ-Spaghetti.jpg#Spaghetti%20811x608](http://foodnetwork.sndimg.com/content/dam/images/food/fullset/2008/3/5/0/NY0100_BBQ-Spaghetti.jpg#Spaghetti%20811x608)



# History



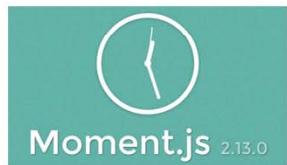
ES3 (1999) => ES5 (2009) => ES6 (2015) => ES7 (2016) => ES8 (2017)



# History



aurelia



# History



mongoDB



# History

## TIOBE Index for January 2015



### January Headline: JavaScript programming language of 2014!

After all these years, JavaScript has finally become TIOBE's language of the year. It was a close finish. Swift and R appeared to be the main candidates for the title but due to a deep fall of Objective-C this month, a lot of other languages took advantage of this and surpassed these two candidates at the last moment.

JavaScript has won the award because it appeared to be the biggest mover of 2014. JavaScript won 1.70% in one year time, followed by PL/SQL (+1.38%) and Perl (+1.33%). The JavaScript programming language has a long history and is always considered as the "ugly duckling" from a language design point of view. Nevertheless, JavaScript has become the standard browser language through the years. Boosted by the successes of JavaScript libraries and frameworks JQuery, Bootstrap, Node.js and GWT, JavaScript really deserves this award.

<http://www.tiobe.com>



# History



<http://www.globalnerdy.com/wordpress/wp-content/uploads/2014/08/javascript-and-the-good-parts.jpg>



# History

true == "1" => TRUE

true === "1" => NOT TRUE

<http://dorey.github.io/JavaScript-Equality-Table/>



# History

```
1 == "1"  
null == undefined  
[0] == false  
0 == ""
```



# History

```
console.log('4' - 4); → 0  
console.log('4' + 4); → 44
```



# History

**Why is the method called `includes` and not `contains`?**

The latter was the initial choice, but that broke code on the web (MooTools adds this method to `Array.prototype`).

<http://www.2ality.com/2016/02/array-prototype-includes.html>



# History



<https://twitter.com/ThePracticalDev/status/766858028859523072/photo/1>



History

TypeScript

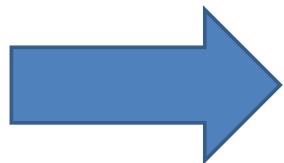


## History

TypeScript



C#



JS



## History

Lisp: Code is data

Haskell: Data is code

Ruby: Strings are code

JavaScript: Undefined is not a function

"I'm in love with the Rust type system  
and going back to JavaScript/ES6 just  
isn't enough."

<https://twitter.com/mattoflambda/status/727290638102876160>  
<https://github.com/rust-lang/rust/issues/33205>



# History



```
<script type="text/javascript">
<script type="text/javascript">
<script type="text/javascript">
```

```
</script>
<script>
</script>
```

**DOESN'T CARE WHAT YOU  
THINK**

memegenerator.net

<http://memegenerator.net>



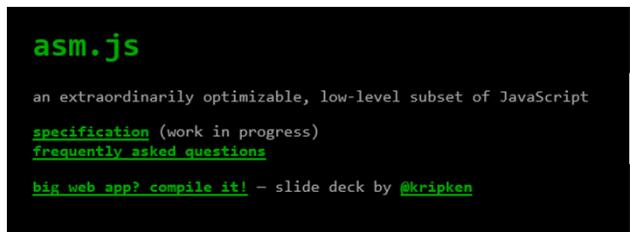
# History



<http://blog.cmaresources.org/wp-content/uploads/2012/05/speed2.jpg>



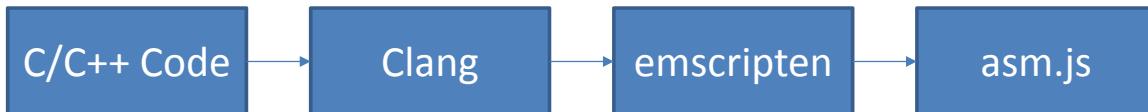
# History



<http://asmjs.org>  
<http://kripken.github.io/emscripten-site/>



# History



<http://asmjs.org>



# History



<https://youtu.be/nB9Pm9Mr7xo>



# Details



# WEBASSEMBLY

<https://github.com/carlosbaraza/web-assembly-logo>



## Details

### WebAssembly

[Overview](#) [Demo](#) [Design](#) [Specification](#) [Community Group](#)

WebAssembly or *wasm* is a new portable, size- and load-time-efficient format suitable for compilation to the web.

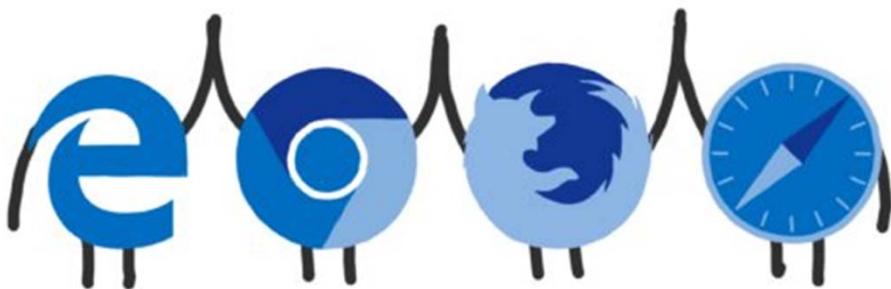
WebAssembly is currently being designed as an open standard by a [W3C Community Group](#) that includes representatives from all major browsers. Expect the contents of this website and its associated design repositories to be in flux: everything is still under discussion and subject to change.

<http://webassembly.github.io/>

<http://webassembly.github.io/>



## Details



<https://twitter.com/linclark/status/836609222733348864/photo/1>



## Details

- Efficient and fast
- Safe
- Open and debuggable
- Part of the open web platform



## Details

Value Types	Description
i32	32-bit integer
i64	64-bit integer
f32	32-bit floating point
f64	64-bit floating point



# Details

Operand	Description
i32.load8_s	Load 1 byte and sign-extend i8 to i32
i32.load16_u	Load 2 bytes and zero-extend i16 to i32
f32.load	Load 4 bytes as f32
i32.store16	Wrap i32 to i16 and store 2 bytes
i64.store16	Wrap i64 to i16 and store 2 bytes
f64.store	(No conversion) store 8 bytes



# Details

Operand	Description
get_local	Read the current value of a local variable
set_local	Set the current value of a local variable
tee_local	Like set_local, but also returns the set value
get_global	Get the current value of a global variable
set_global	Set the current value of a global variable

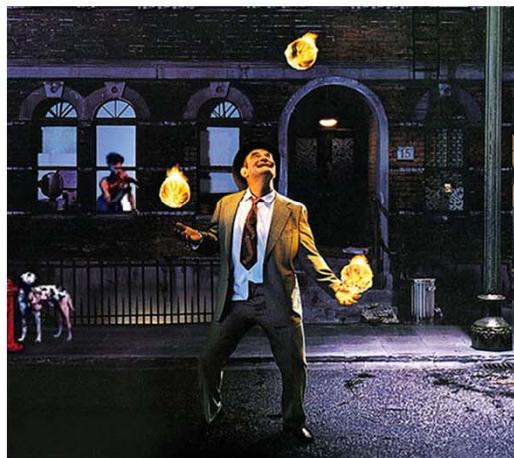


# Details

Operand	Description
loop	A block with an additional label at the beginning which may be used to form loops
br_if	Conditionally branch to a given label in an enclosing construct
return	Return zero or more values from this function



# Details



<http://www.2112.net/powerwindows/downloads/smartphone960x854/Rush-HoldYourFire2.jpg>



Assembling the Web – A Tour of WebAssembly

## DEMO: TARGETING WASM



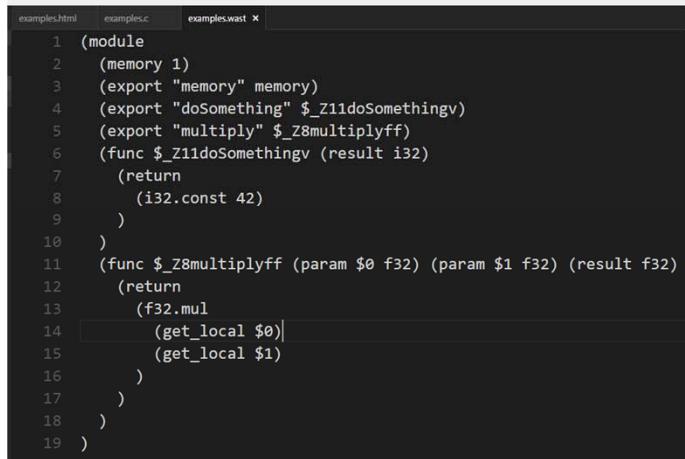
## Demo: Targeting wasm

A screenshot of a code editor showing two tabs: "examples.html" and "examples.c". The "examples.c" tab is active, displaying the following C code:

```
1 // Use http://mbebenita.github.io/WasmExplainer
2 int doSomething() {
3     return 42;
4 }
5
6 float multiply(float x, float y) {
7     return x * y;
8 }
```

The "examples.wast" tab is visible at the top but contains no visible text.

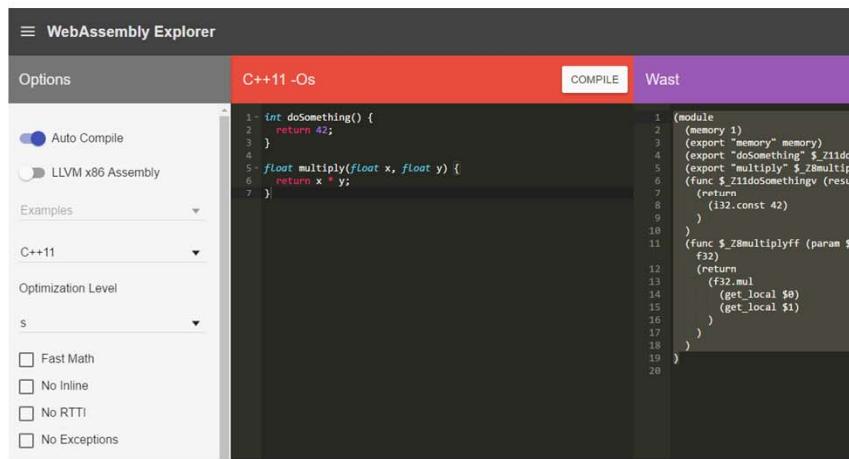
# Demo: Targeting wasm



```
examples.html examples.c examples.wast x
1 (module
2   (memory 1)
3   (export "memory" memory)
4   (export "doSomething" $Z11doSomething)
5   (export "multiply" $Z8multiplyff)
6   (func $Z11doSomethingv (result i32)
7     (return
8       (i32.const 42)
9     )
10   )
11   (func $Z8multiplyff (param $0 f32) (param $1 f32) (result f32)
12     (return
13       (f32.mul
14         (get_local $0)
15         (get_local $1)
16       )
17     )
18   )
19 )
```



# Demo: Targeting wasm



WebAssembly Explorer

Options C++11 -Os COMPILER Wast

Auto Compile (selected)

LLVM x86 Assembly

Examples C++11 Optimization Level S

Fast Math No Inline No RTTI No Exceptions

```
int doSomething() {
    return 42;
}

float multiply(float x, float y) {
    return x * y;
}
```

```
1 (module
2   (memory 1)
3   (export "memory" memory)
4   (export "doSomething" $Z11doSomething)
5   (export "multiply" $Z8multiplyff)
6   (func $Z11doSomethingv (result i32)
7     (return
8       (i32.const 42)
9     )
10   )
11   (func $Z8multiplyff (param $0 f32) (param $1 f32) (result f32)
12     (return
13       (f32.mul
14         (get_local $0)
15         (get_local $1)
16       )
17     )
18   )
19 )
```



# Demo: Targeting wasm

```
<script>
// https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
var wasmExports;

fetch('examples.wasm')
.then(response =>
  response.arrayBuffer())
.then(buffer => {
  let codeBytes = new Uint8Array(buffer);
  try {
    WebAssembly.compile(codeBytes)
    .then(module => {
      let instance = new WebAssembly.Instance(module);
      wasmExports = instance.exports;
      console.log('wasm is loaded');
    })
  } catch (e) {
    alert("Error: " + e);
  }
});
</script>
```



# Demo: Targeting wasm

```
<button onclick='multiplyClick()>Multiply!</button>
<script>
  function multiplyClick() {
    alert(wasmExports.multiply(2, 21));
  }
</script>
```



# Demo: Targeting wasm



```
<!-- In the function
<script type="text/javascript">
  import * as wasmModule from './wasm';
  alert(wasmModule.add(1, 2));
</script>
```

Details

Sean T. Larkin [Follow](#)  
@Webpack Core team & AngularCLI team. Program Manager @Microsoft @EdgeDevTools. Web Performance & Accessibility Advocate.

Jul 31 · 4 min read

## webpack awarded \$125,000 from MOSS Program

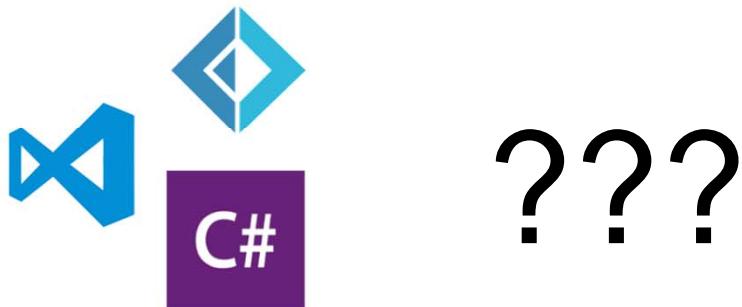
Implementing first-class support for WebAssembly

m';

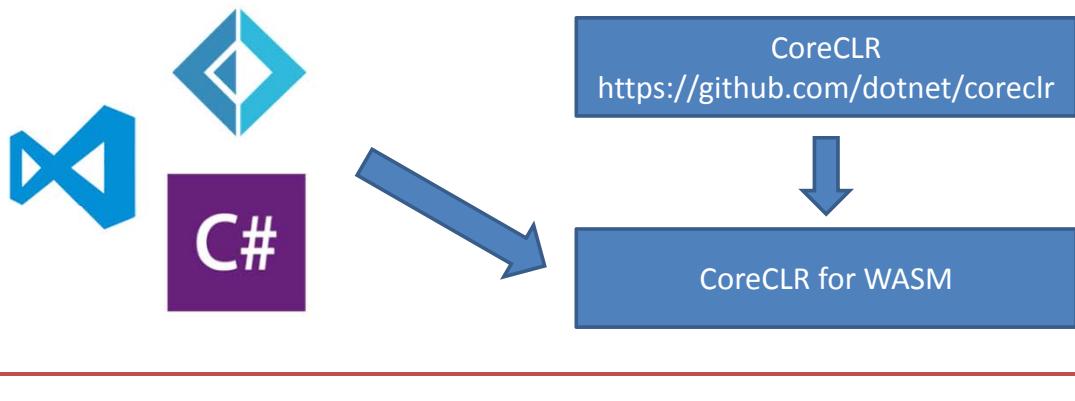
WA  
WEBASSEMBLY



# Details



# Details



## Details

“C# and Java require ... GC and stack manipulation primitives. That's also on the roadmap, but after MVP, threads, and dynamic linking.”

“We want to use the rich type information that has been produced by the TypeScript team to produce strongly typed APIs that can be consumed by C# on WebAssembly.”

<http://stackoverflow.com/questions/31994034/webassembly-javascript-and-other-languages>  
[https://www.reddit.com/r/programmerchat/comments/4dxpcp/i\\_am\\_miguel\\_de\\_icaza\\_i\\_started\\_xamarin\\_mono\\_gnome/d1v9xyd](https://www.reddit.com/r/programmerchat/comments/4dxpcp/i_am_miguel_de_icaza_i_started_xamarin_mono_gnome/d1v9xyd)



## Details

### WebAssembly for .NET

nuget v0.2.4-preview

A library able to create, read, modify, write and execute WebAssembly (WASM) files from .NET-based applications. Execution does not use an interpreter. WASM Instructions are mapped to their .NET equivalents at runtime by the .NET JIT compiler.

A preview is available via NuGet at <https://www.nuget.org/packages/WebAssembly>. No API surface area is planned; the next release will be 1.0 with full support for the WebAssembly “MVP”.

#### Getting Started

The API is unstable--this means the names and structure of everything can change--but it is `WebAssembly.Module`.

- Read and write WASM binary files via `ReadFromBinary()` and `WriteToBinary()`.
- Create a new WASM binary file from scratch: create a new `WebAssembly.Module` instance.
- `WebAssembly.Module` reflects the binary format in a very pure form: nearly anything in the file is covered. As the binary format is optimized for size efficiency, it can be difficult to understand what each index space and labels mean. The best resource for understanding how things work is the `WebAssembly.Tests` project.
- `WebAssembly.Compile` converts WebAssembly binary files (WASM) to .NET via the run-time code generation features in `System.Reflection.Emit`. As it ultimately runs on the same CLR as C#, performance is equivalent.

```
class Program
{
    static void Main(string[] args)
    {
        var module = Compile.FromBinary<dynamic>(
            "collatz.wasm");
        Console.WriteLine(
            module().Exports.collatz(5));
    }
}
```

<https://github.com/RyanLamansky/dotnet-wasmbly>



# Visual Studio Live! San Diego 2018

# Details

Server & Tools Blogs > Developer Tools Blogs > .NET Web Development and Tools Blog

Executive Bloggers Visual Studio Products DevOps Languages Features .NET

## .NET Web Development and Tools Blog

Your official information source from the .NET Web Development and Tools group at Microsoft.

### A new experiment: Browser-based web apps with .NET and Blazor

February 6, 2018 by Daniel Roth // 10 Comments

Share 904 Twitter 500 LinkedIn 284

Today I'm excited to announce a new experimental project from the ASP.NET team called Blazor. Blazor is an experimental web UI framework based on C#, Razor, and HTML that runs in the browser via WebAssembly. Blazor promises to greatly simplify the task of building fast and beautiful single-page applications that run in any browser. It does this by enabling developers to write .NET-based web apps that run client-side in web browsers using open web standards.

<https://blogs.msdn.microsoft.com/webdev/2018/02/06/blazor-experimental-project/>

Visual Studio 25 YEARS OF CODING INNOVATION

# Details

This repository Search Pull requests Issues Marketplace Explore

aspx / Blazor Watch 202 Uns

Code Issues 44 Pull requests 5 Projects 1 Wiki Insights

An experimental web UI framework using C#/Razor and HTML, running in the browser via WebAssembly

199 commits 4 branches 0 releases

Branch: dev New pull request Create new file Upload files Find

d-dizhevsky and SteveSandersonMS replaced backslashes with common slashes Latest

samples replaced backslashes with common slashes

src Further renderer refactoring

test Further renderer refactoring

.nifianore Simple F2F test for static site

<https://github.com/aspnet/blazor>

Visual Studio 25 YEARS OF CODING INNOVATION

Assembling the Web – A Tour of WebAssembly

## DEMO: C# AND WASM EXPERIMENTS



## Conclusion

Greg Reimer, Web Developer

“We won't need to learn wasm any more than C++ programmers need to learn assembly. In the same vein, to remain competitive as developers, we'll need to learn how it fits in our toolchains, which means getting nice and cozy with it. In other words, we'll all be *generating* wasm, but few of us will actually be writing or debugging it directly.”

<https://www.quora.com/What-does-WebAssembly-mean-for-those-who-are-currently-learning-web-development>



# Conclusion



[https://formula1.files.wordpress.com/2016/05/season2016\\_race5\\_wallpapers\\_09.jpg](https://formula1.files.wordpress.com/2016/05/season2016_race5_wallpapers_09.jpg)



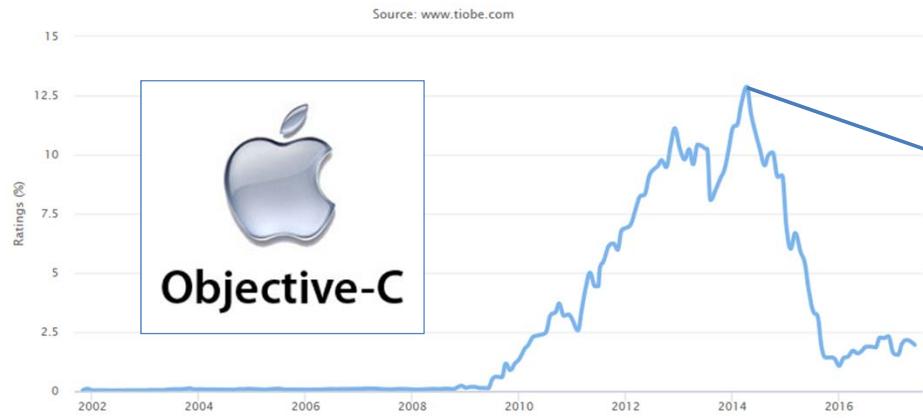
# Conclusion



<http://thebridge-can.com/wp-content/uploads/2014/09/Braveheart-Freedom-Slide.jpg>

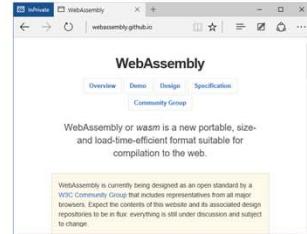
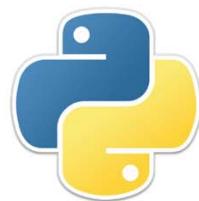


# Conclusion



Visual Studio 25  
YEARS OF CODING INNOVATION

# Conclusion



Visual Studio 25  
YEARS OF CODING INNOVATION

# Conclusion



Visual Studio **LIVE!** | San Diego

## Assembling the Web – A Tour of WebAssembly

Remember...

- <https://github.com/JasonBock/Presentations/blob/master/Assembling%20the%20Web%20-%20A%20Tour%20of%20WebAssembly.pptx>
- <https://github.com/JasonBock/AssemblingTheWeb>
- References in the notes on this slide

**Jason Bock**  
**Practice Lead**  
**Magenic**

Level: Intermediate

