



ROCK YOUR CODE  
TOUR 2017

CHICAGO

# Assembling the Web – A Tour of WebAssembly

Jason Bock  
Practice Lead,  
Magenic

## 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://www.slideshare.net/JasonBock2/assembling-the-web-a-tour-of-webassembly>

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



# Overview

- History
- Details
- Conclusion

Remember....

<https://www.slideshare.net/JasonBock2/assembling-the-web-a-tour-of-webassembly>

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



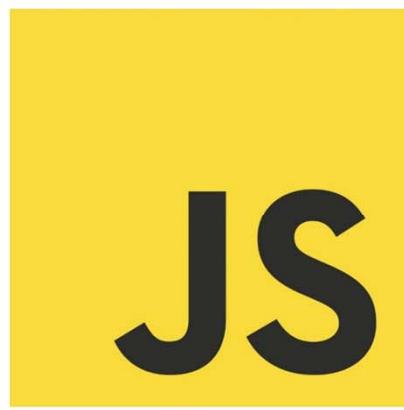
# History



[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)

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# History



ES3 (1999) => ES5 (2009) => ES6 (2015) => ES7 (WIP)

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## History



aurelia



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## History



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# 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>

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# History



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

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# 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>

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

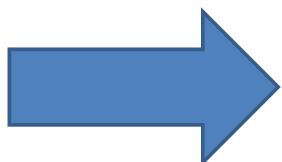
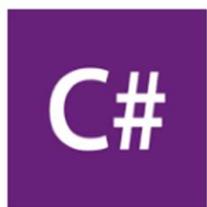
# History

TypeScript

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## History

TypeScript



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## 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>

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## History



<http://memegenerator.net>

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

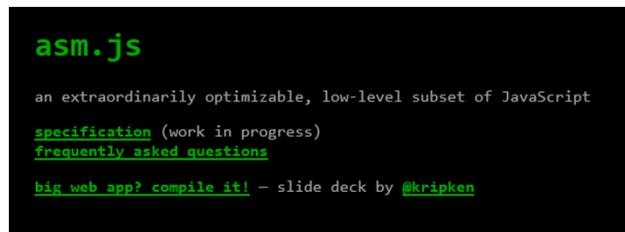
## History



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

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

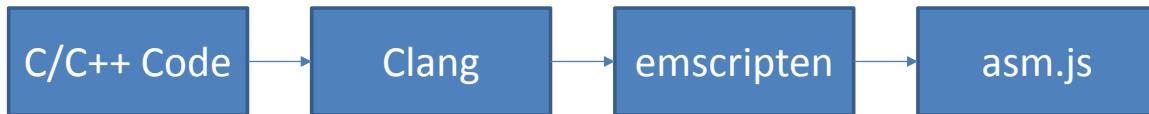
# History



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

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# History



<http://asmjs.org>

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# History



<https://youtu.be/nB9Pm9Mr7xo>

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Details



# WEBASSEMBLY

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

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# 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.

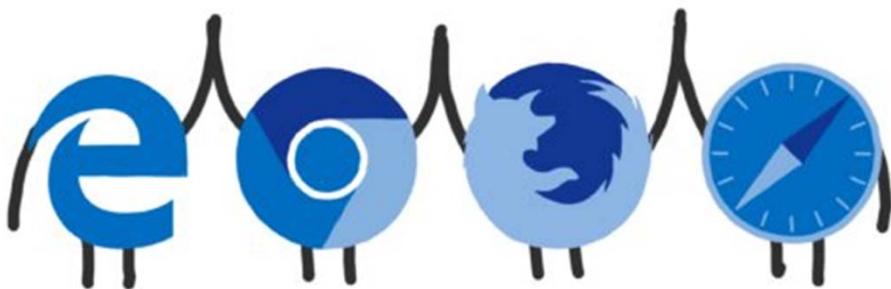
WebAssembly is currently being designed as an open standard by a [W3C Community Group](#) that includes representatives from all major browsers. Expect the contents of this website and its associated design repositories to be in flux: everything is still under discussion and subject to change.

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

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



# Details



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



## Details

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



## Details

- Chrome (Canary) – available
- Firefox – available
- Edge – Can be enabled via the Experimental JavaScript Features flag. See this link for details: <https://blogs.windows.com/msedgedev/2017/04/20/improved-javascript-performance-webassembly-shared-memory/>
- Safari – Coming in 11



## 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

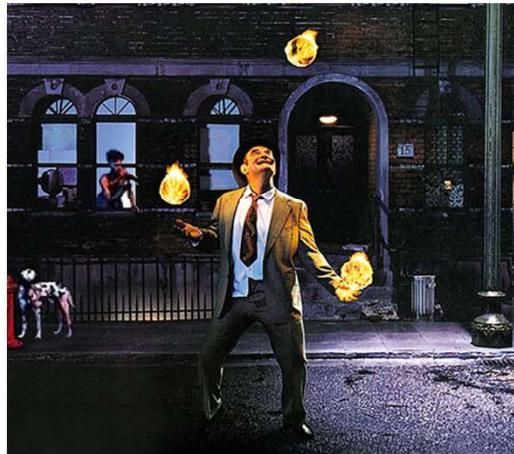


# Details

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>

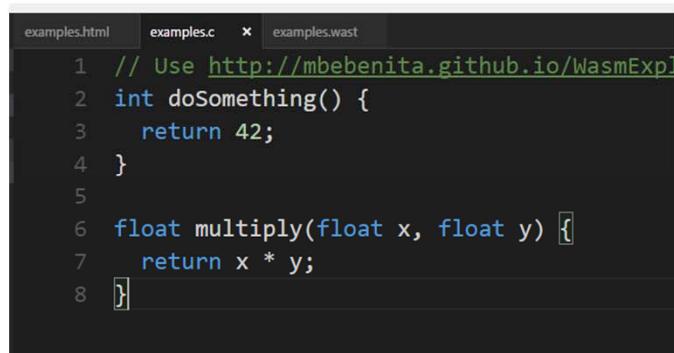
Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Assembling the Web – A Tour of WebAssembly

**DEMO: TARGETING WASM**

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Demo: Targeting wasm

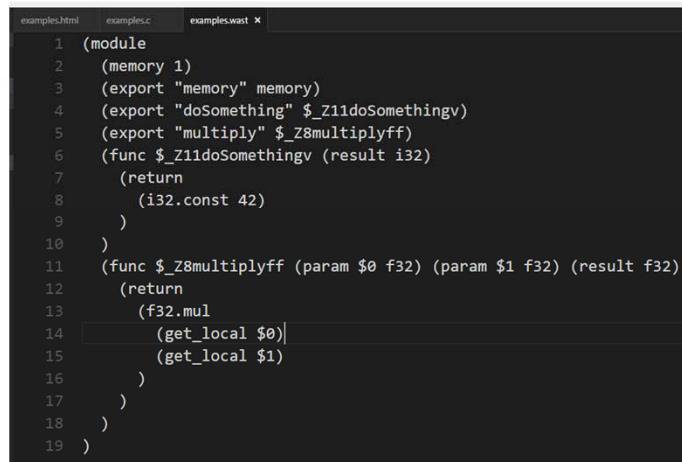


A screenshot of a code editor showing two tabs: "examples.html" and "examples.c". The "examples.c" tab is active, displaying the following C code:

```
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 }
```



## Demo: Targeting wasm



A screenshot of a code editor showing the "examples.wast" tab, which contains the WebAssembly (WAST) code generated from the "examples.c" file. The code defines a module with memory and exports for the "doSomething" and "multiply" functions.

```
1 (module
2   (memory 1)
3   (export "memory" memory)
4   (export "doSomething" $._Z11doSomethingv)
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

The screenshot shows the WebAssembly Explorer interface. On the left, there are options for Auto Compile, LLVM x86 Assembly, Examples (C++11 selected), Optimization Level (S selected), and compiler flags (Fast Math, No Inline, No RTTI, No Exceptions). The main area displays two panes: C++11 -Os (containing C++ code for doSomething() and multiply()) and Wast (containing the generated WebAssembly assembly code).

```

1 int doSomething() {
2     return 42;
3 }
4
5 float multiply(float x, float y) {
6     return x * y;
7 }

(module
  (memory 1)
  (export "doSomething" $ _Z11doSomething)
  (export "multiply" $ _Z8multiply)
  (func $ _Z11doSomething (result i32)
    (return (i32.const 42))
  )
  (func $ _Z8multiply (param $0 f32)
    (return
      (f32.mul
        (get_local $0)
        (get_local $1)
      )
    )
  )
)

```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# 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>

```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Demo: Targeting wasm

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



## Demo: Targeting wasm

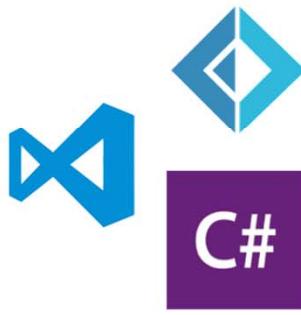


## Details

```
<!-- In the future... -->
<script type="module">
    import * as wasmMethods from 'examples.wasm';
    alert(wasmMethods.multiply(2, 21));
</script>
```



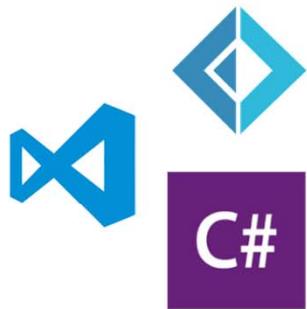
## Details



???



## Details



CoreCLR  
<https://github.com/dotnet/coreclr>

CoreCLR for WASM

JavaScript VM (?)

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## 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)

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Details

The screenshot shows a Microsoft Visual Studio interface. On the left, the Solution Explorer displays a project named 'MySuperThing' with files like Layout.cshtml, NavMenu.cshtml, Counter.cshtml, FetchData.cshtml, Index.cshtml, Program.cs, and TodoList.cshtml. In the center, the code editor shows the TodoList.cshtml file with the following content:

```

<ul>
    <li>New thing</li>
</ul>

<input @bind(nextItem) placeholder="Type here..." />
<button @onclick(AddItem)>Add item</button>

@functions {
    string nextItem;
    List<TodoItem> TodoItems = new List<TodoItem>();

    void AddItem()
    {
        TodoItems.Add(new TodoItem { Text = "New thing" });
    }
}

```

On the right, a browser window shows the application running at [localhost:65278/TodoList](http://localhost:65278/TodoList), displaying a simple todo list with one item: 'New thing'. A button labeled 'Add item' is present.

<https://www.youtube.com/watch?v=MiLAE6HMri0>

Visual Studio LIVE!  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Details

The screenshot shows the Chrome DevTools debugger's Sources tab. It is currently viewing the RazorComponent.cs file. A breakpoint is set on line 77. The code in the file is:

```

public static string GetViewClassName(string rootDir, string cshtmlFilename)
{
    cshtmlFilename = cshtmlFilename.Replace('/', Path.DirectorySeparatorChar);

    if (!rootDir.EndsWith(Path.DirectorySeparatorChar.ToString()))
    {
        rootDir += Path.DirectorySeparatorChar;
    }
}

```

The debugger's sidebar indicates that the execution is "Paused on breakpoint". The call stack shows the following frames:

- GetViewClassName (RazorComponent.cs:77)
- SendDebuggerMessage (Blazor.Runtime.js:20)
- \_invokeJsFunc (dnajs:1)
- Module.\_\$lInterop\_CallDotNet (dnajs:1)
- ccallFunc (dnajs:1)

<https://twitter.com/stevensanderson/status/893553691747332096/photo/1>

Visual Studio LIVE!  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Details



<http://www.mono-project.com/news/2017/08/09/hello-webassembly>  
<http://www.kumpera.com/wasm/driver.html/>



## 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 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's going on without looking at the raw bytes. The best resource for understanding how things work is the [WebAssembly Tests](#).
- `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-wabassembler>



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)

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Conclusion

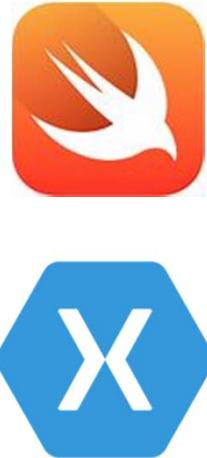
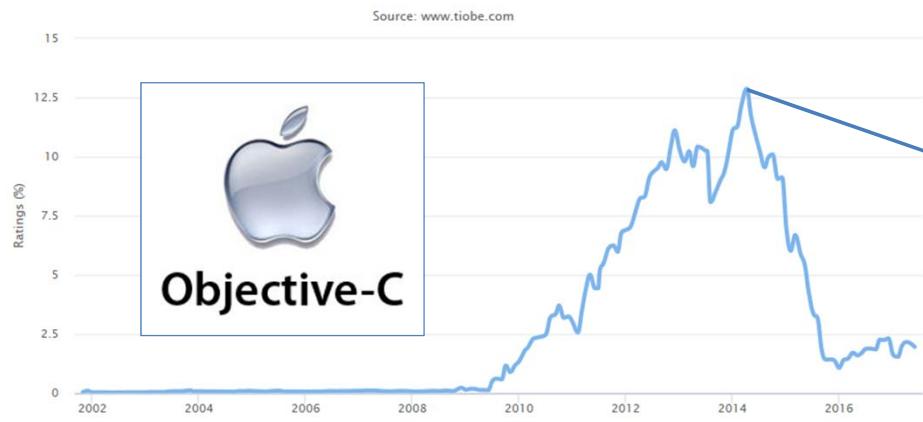


FREEDOM

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

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

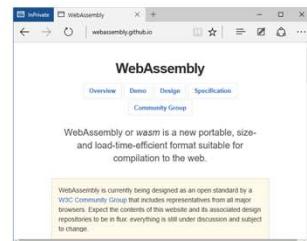
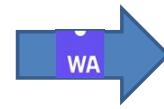
# Conclusion



<https://www.tiobe.com/tiobe-index/objective-c/>

Visual Studio LIVE!  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Conclusion



Visual Studio LIVE!  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Conclusion



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

**ROCK YOUR CODE TOUR 2017 | CHICAGO**

## Assembling the Web – A Tour of WebAssembly

Remember...

- <https://docs.com/jason-bock/9318/assembling-the-web-a-tour-of-webassembly>
- <https://github.com/JasonBock/AssemblingTheWeb>
- References in the notes on this slide

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