Modern applications:
The WebAssembly & Serverless revolutions - in Action

think tecture

Christian Weyer
@christianweyer
CTO

Christian Weyer

- Founder & CTO at Thinktecture AG
- Personal focus on
  - Mobile & web-based application architectures
  - Interoperability, cross-device
  - Pragmatic end-to-end solutions
  - Cloud-native & serverless architectures

- Independent Microsoft Regional Director
- Microsoft MVP for ASP.NET (Architecture)
  ASPInsider, AzureInsider
- Google GDE for Web Technologies

- christian.weyer@thinktecture.com
- @christianweyer

think tecture

Modern applications?

think tecture

Cross-platform clients

think tecture

No app stores

think tecture

Any code on the client

think tecture

Data & business logic - available, scaleable, easy deployable

think tecture

No servers

think tecture

Any code on the server

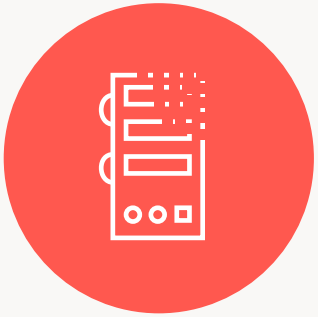
think tecture

... the language(s) you like ...

A different approach for Microservices

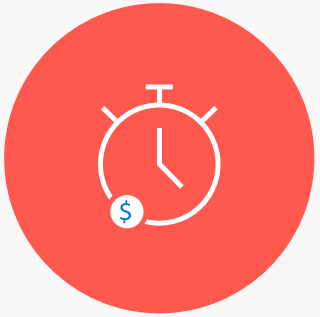
Serverless

What is Serverless, anyway?



Abstraction of servers

Event-driven & instant scale



Micro-billing

DevOps productivity

Focus on business logic

Faster time to market

Serverless in Azure

▪ Serverless Compute	→	Azure Functions
▪ Serverless Database	→	Azure Cosmos DB
▪ Serverless Realtime	→	Azure SignalR Service
▪ Serverless Events	→	Azure Event Grid
▪ Serverless Workflows	→	Azure Logic Apps Azure Durable Functions
▪ Serverless IoT	→	Azure IoT Hub
▪ Serverless Analytics	→	Azure Application Insights
▪ Serverless Containers	→	Azure Container Instances

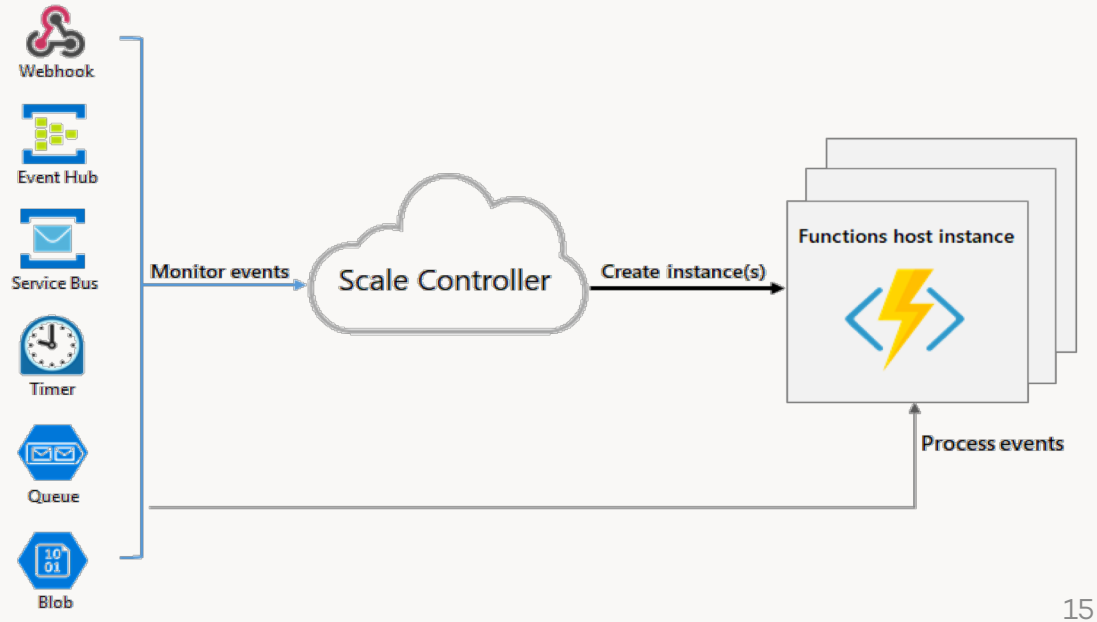
- *… and more …*
  https://azure.microsoft.com/en-us/overview/serverless-computing/

Azure Functions

Serverless Compute

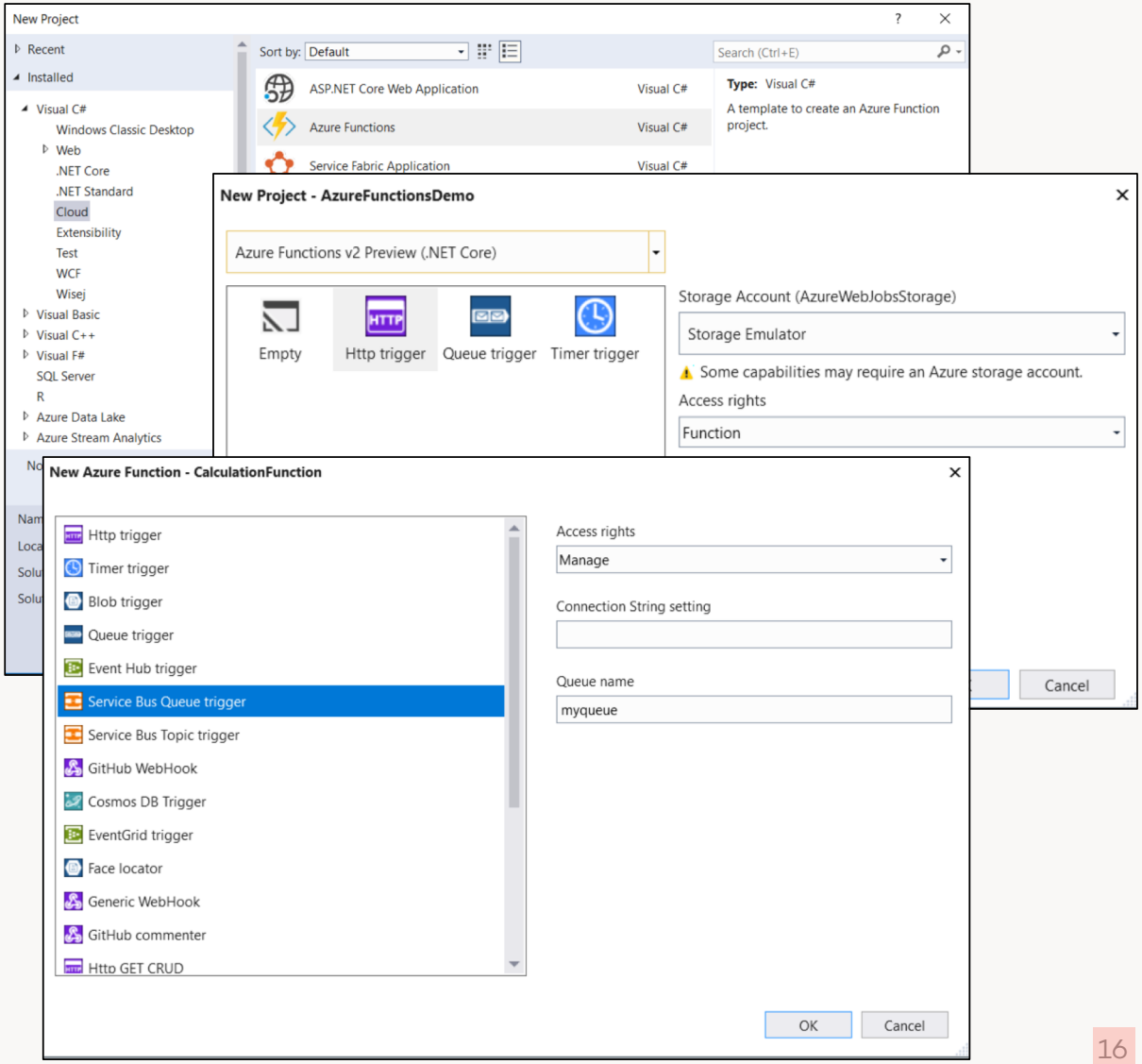
Azure Functions: Functions-as-a-Service

- Azure Functions is code being triggered by an event
- Basic principles enable powerful use cases
  - Triggers
  - Bindings
- Two ways of operation
  - Cost-optimized, automatic scale (aka Consumption Plan)
  - Always-on, via Azure App Service
- V2 Runtime built on .NET Core
  - Available for self-hosting



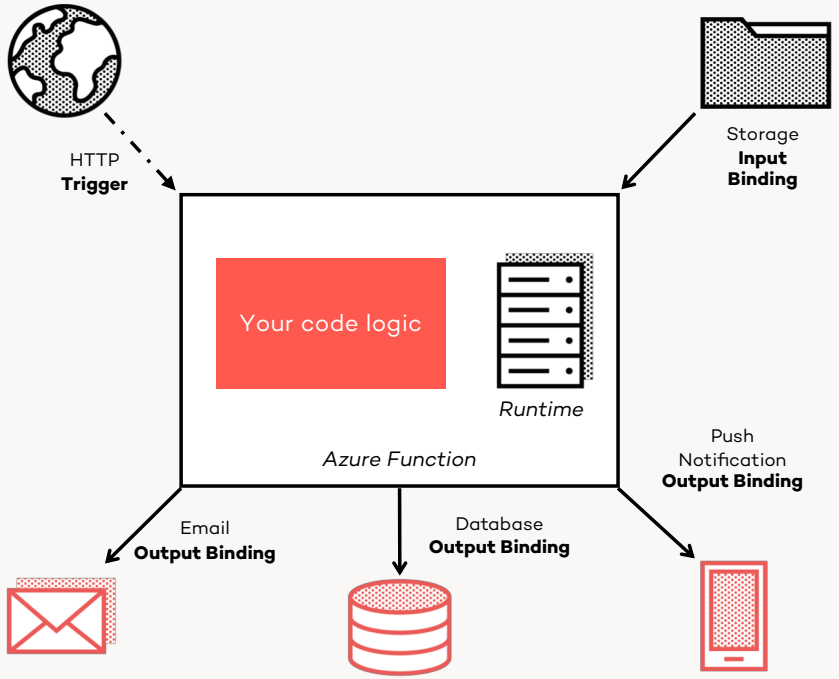
Developing Functions

- In portal, via CLI, or in IDE
- Local tooling & runtime available
  - Debugging locally or remote
- Multiple language bindings supported
  - C# & C# Script (CSX)
  - F#
  - JavaScript, TypeScript (node.js)
  - Java (*preview*)



Triggers & Bindings

- Incoming event triggers function
- Input bindings enable easy access to data from various data sources
- Output bindings offer easy access to outbound data sinks



Azure Functions are not just a replacement for your Web APIs.

They are an event-driven code execution hub (triggers) enabling easy integration with data sources (input bindings) and data sinks (output bindings) - and one type of event can be HTTPS requests.

Azure Cosmos DB

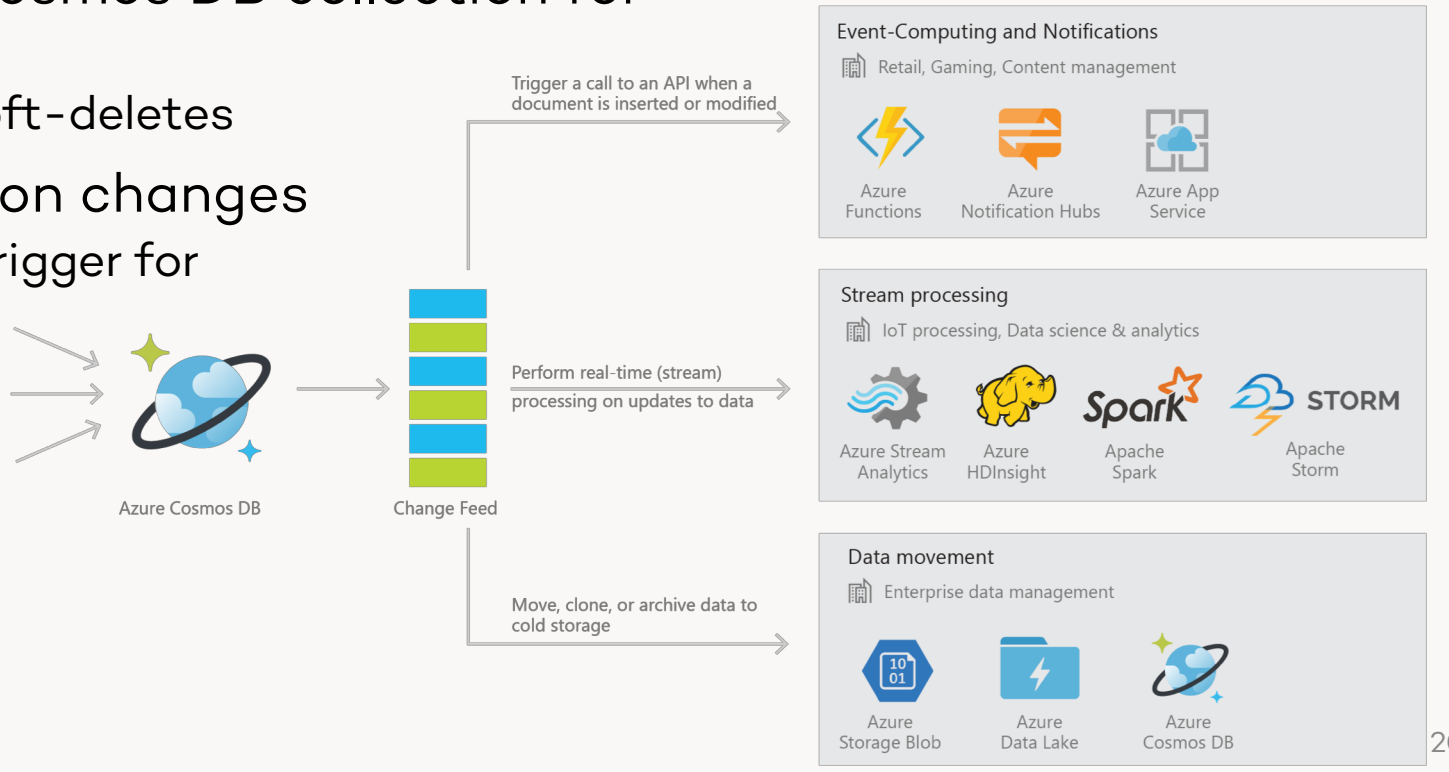
Serverless Database

Cosmos DB: A Database with Many Faces

- Globally distributed, elastically scalable
  - Fully managed by Azure
  - But not yet fully automatically scaling
- Multi-model database
  - Key-value, document, graph ➔ NoSQL
- Multiple APIs
  - DocumentDB, with SQL & JS APIs
  - MongoDB API
  - Cassandra API
  - Table Storage API
  - Graph Database with Gremlin API

Cosmos DB Change Feed

- Building event-driven architectures with Cosmos DB
- Listening to Azure Cosmos DB collection for any changes
  - Inserts, updates, soft-deletes
- Subscribe to collection changes
  - Can be used as a trigger for Functions

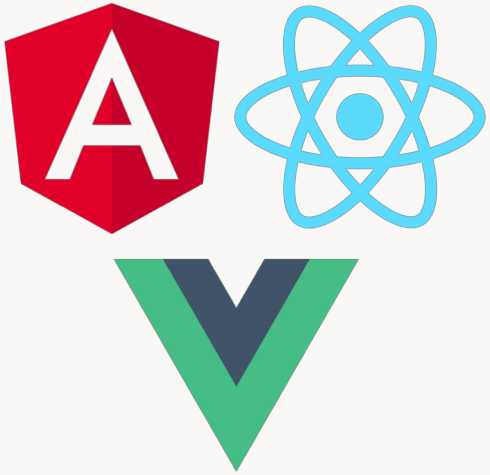


State of the art

Modern Applications

Modern Applications & the Web

- Single Page Applications (SPAs) are the new smart clients



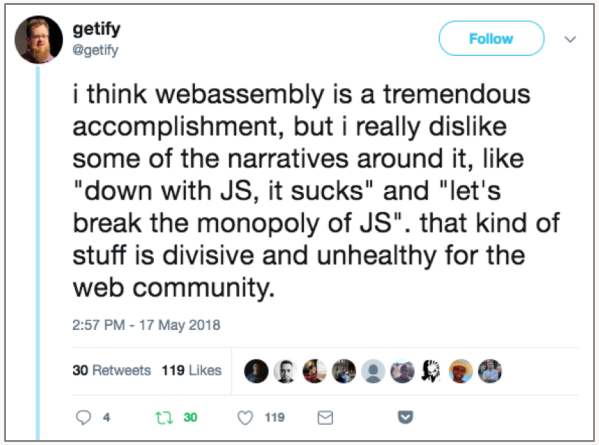
Your native code in the browser

WebAssembly

WebAssembly (WASM)

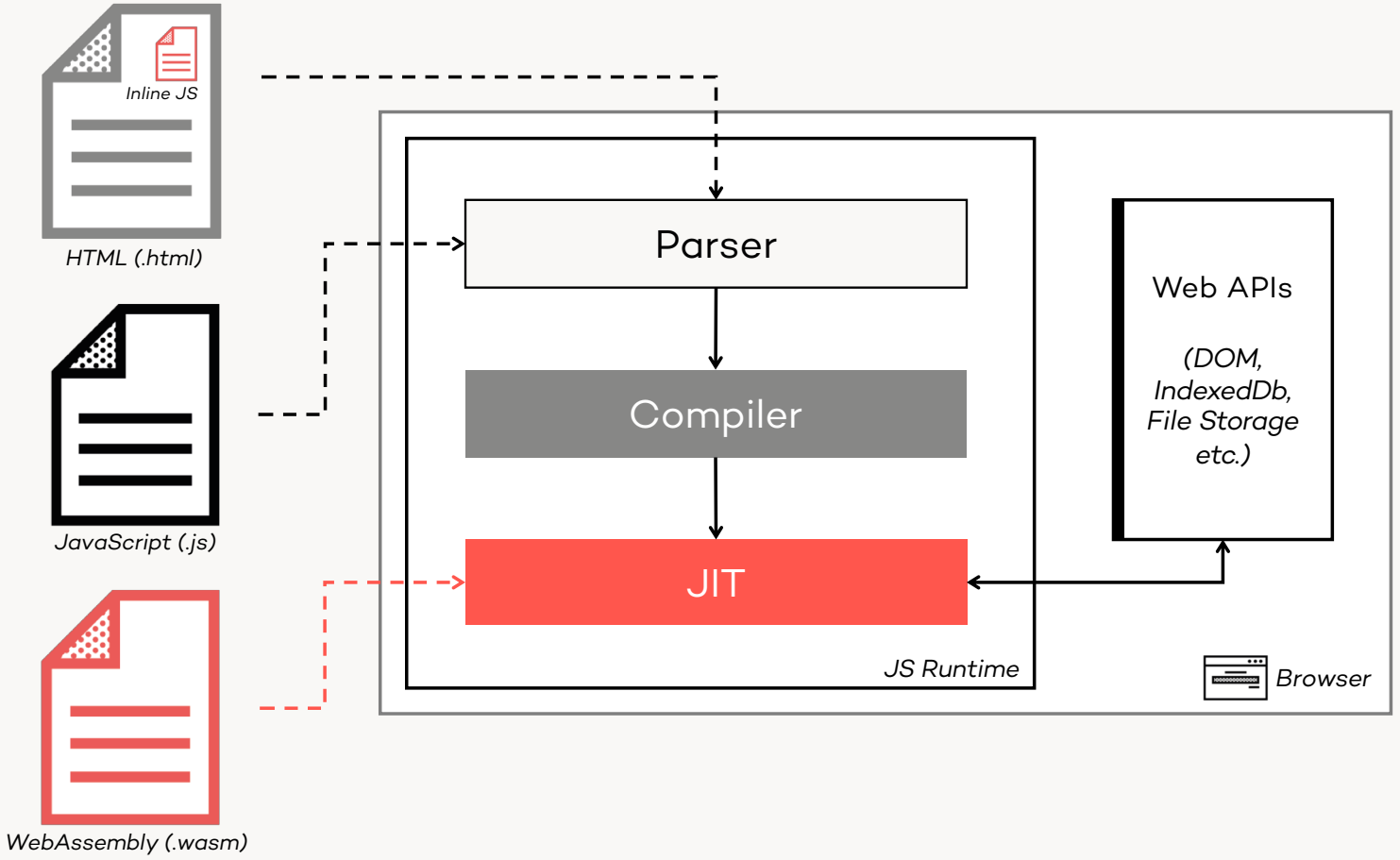
- Low-level bytecode for the Web (beyond asm.js)
  - Bring any language into the browser
    - Currently targeted at C/C++ (and Rust)
  - Achieve superior performance in certain cases

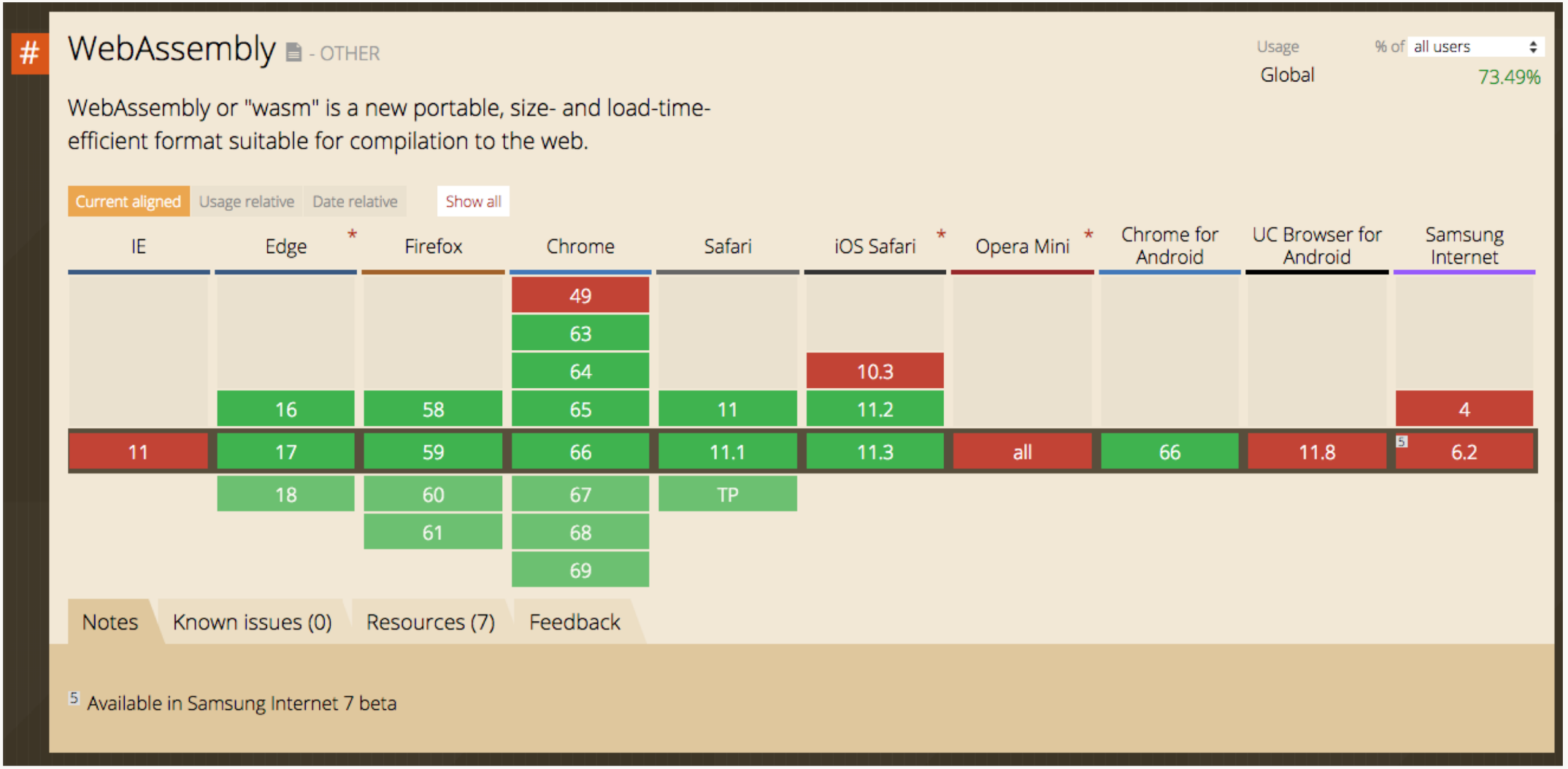
- Goals
  - Fast, efficient, portable
  - Readable and debuggable
  - Use existing sandboxing tools
  - Don't break the web



https://twitter.com/getify/status/997098808344547328

WASM & JS Architecture *(simplified)*





https://caniuse.com/#search=WebAssembly

WASM – A Simple Sample

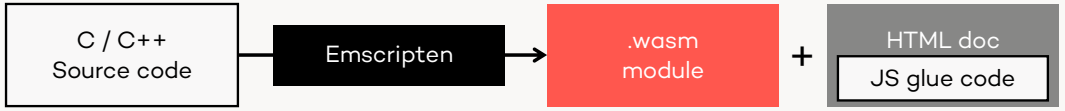
```
(module
  (func $add (param $lhs i32) (param $rhs i32) (result i32)
    get_local $lhs
    get_local $rhs
    i32.add)
  (export "add" (func $add))
)
```

- Binary (.wasm) & text (.wat) format

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style>
    body {
      background-color: rgb(255, 255, 255);
    }
  </style>
</head>
<body>
  <span id="container"></span>
  <script src="./main.js"></script>
</body>
</html>
```

```
fetch('../out/main.wasm').then(response =>
  response.arrayBuffer()
).then(bytes => WebAssembly.instantiate(bytes)).then(results => {
  instance = results.instance;
  document.getElementById("container").innerText = instance.exports.add(1,1);
});
```

WASM Tooling

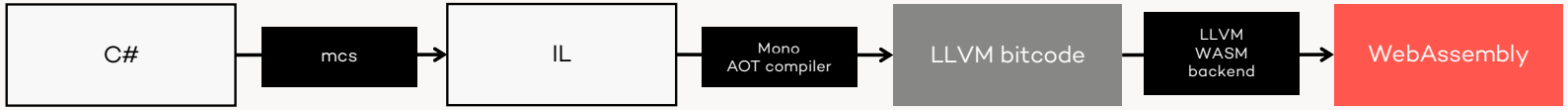


- Baremetal: WebAssembly Studio
  - Playing around with WASM in the browser

- C/C++ tools & compilers
  - Binaryen: compiler and toolchain infrastructure library for WebAssembly, written in C++
  - LLVM: *The* modern C/C++ compiler toolchain
  - Emscripten: LLVM-to-JavaScript-or-WASM compiler
- Tools cross-compiling from major languages
  - Cheerp (from C++)
  - TeaVM (from Java)
  - Rust
  - mono-wasm (from C#)

.NET & WASM

- Mono is Microsoft's strategy for cross-platform client applications
  - Xamarin is based on Mono, Unity as well
  - Support for .NET Standard
- Mono team is working on tooling to compile Mono/.NET code to WASM



- Aka 'mono-wasm'
  - JIT – for fast development cycles, including live reload
  - AOT – for optimized runtime deployment & execution

An experiment based on WASM

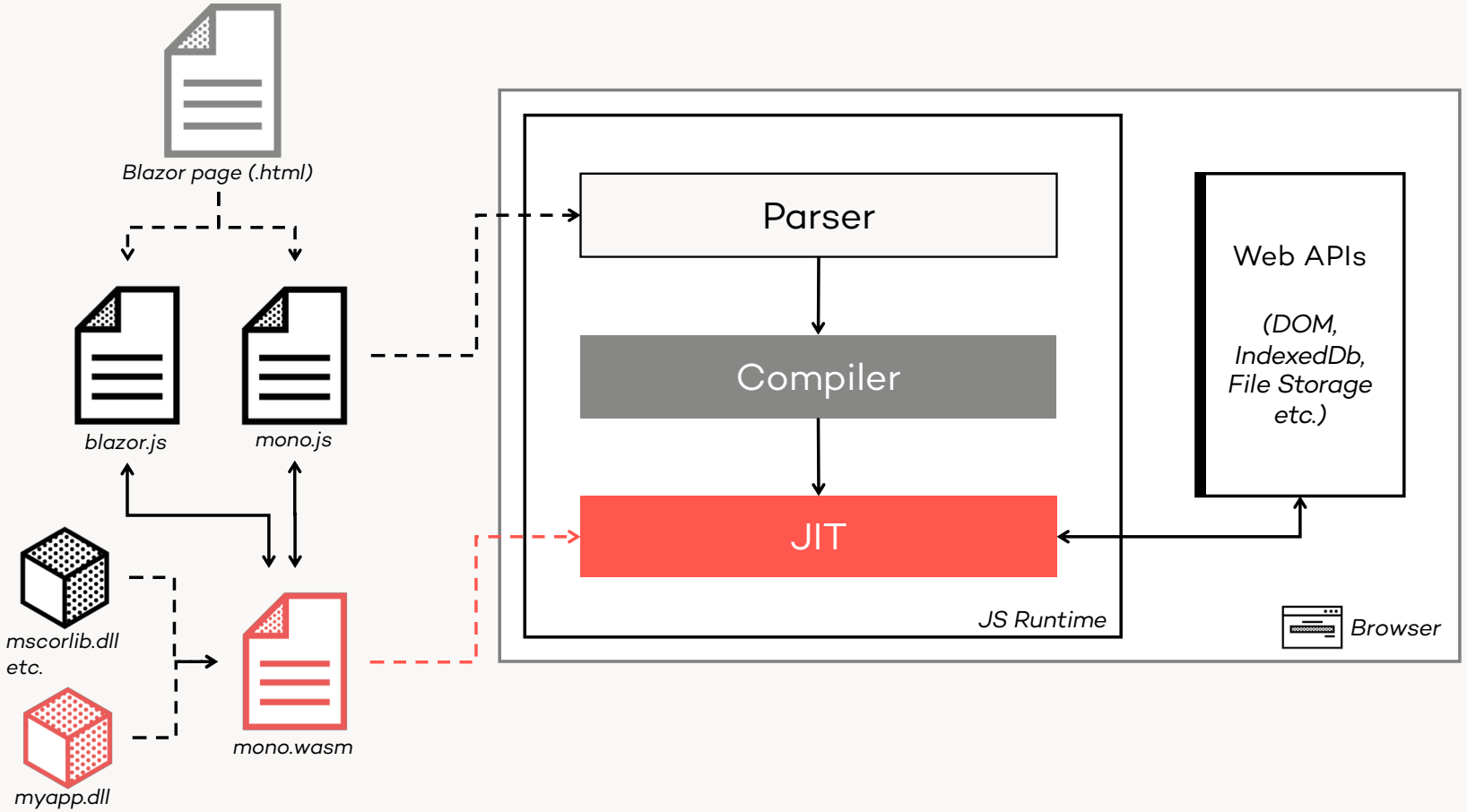
ASP.NET Core Blazor

Blazor – WASM in action

- Blazor is an experimental Single Page Application (SPA) framework based on .NET
- Blazor projects run 100% in the browser *
  - On top of a WASM build of the Mono .NET runtime *
  - Compiled .NET assemblies are loaded into the browser and JIT compiled *
- Uses Razor / .cshtml templates for rendering HTML / CSS
- Uses C# for writing application logic

** as of July 23, 2018*

Blazor Architecture*



* as of July 23, 2018

What is Blazor NOT?

- It is NOT about running any .NET Code just in the browser
  - We still have a sandbox
  - Various .NET Standard APIs will just not work in the browser
- It is NOT a way to port Razor / MVC / server-side web applications to an SPA
- It is NOT a way to bring XAML / WPF / Xamarin into the browser
  - You may want to look at the Ooui.WASM open source project for this

- It is NOT production-ready
  - Therefore, it is NOT suitable for real-world applications (yet?)

Modern applications:
The WebAssembly & Serverless revolutions - in Action

think tecture

https://github.com/thinktecture/dotnetrocks-tour-europe-2018

Christian Weyer
@christianweyer
CTO

Resources

- Azure Functions Dependency Injection (with AutoFac)
  - https://github.com/introtocomputerscience/azure-function-autofac-dependency-injection
- Processing 100,000 Events Per Second on Azure Functions
  - https://blogs.msdn.microsoft.com/appserviceteam/2017/09/19/processing-100000-events-per-second-on-azure-functions/
- Azure Functions – Significant Improvements in HTTP Trigger Scaling
  - https://www.azurefromthetrenches.com/azure-functions-significant-improvements-in-http-trigger-scaling/
- Azure Functions: Cold Starts in Numbers
  - https://mikhail.io/2018/04/azure-functions-cold-starts-in-numbers/

Resources

- WebAssembly Specification
  - http://webassembly.github.io/spec/core/index.html
- LVVM
  - https://llvm.org/
- Emscripten
  - http://kripken.github.io/emscripten-site/
- Mono & WebAssembly
  - http://www.mono-project.com/news/2018/01/16/mono-static-webassembly-compilation/
- ASP.NET Blazor
  - https://blazor.net/index.html