International
JavaScript Conference

SPA Revolution with WebAssembly and Blazor

Rainer Stropek | software architects
@rstropek

International
JavaScript Conference

Samples:
https://github.com/software-architects/learn-blazor
https://learn-blazor.com

Blazor Introduction

ASP.NET Browser

Blazor

Rainer Stropek

software architects gmbh

Web http://www.timecockpit.com
Mail rainer@timecockpit.com
Twitter @rstropek

time cockpit
Saves the day.

https://commons.wikimedia.org/wiki/File:Images_200px-ISO_C%2B%2B_Logo_svg.png

https://commons.wikimedia.org/wiki/File:Csharp_Logo.png

https://commons.wikimedia.org/wiki/File:Wikimedia_Foundation_Servers-8055_14.jpg

https://en.wikipedia.org/wiki/File:Google_Chrome_icon_(September_2014).svg
https://commons.wikimedia.org/wiki/File:Antu_firefox.svg
https://commons.wikimedia.org/wiki/File:Microsoft_Edge_logo.svg

Microsoft®
Silverlight™

https://commons.wikimedia.org/wiki/File:Silverlight_Logotyp.jpg

https://de.wikipedia.org/wiki/Datei:Unofficial_JavaScript_logo_2.svg

https://de.wikipedia.org/wiki/Datei:AngularJS_logo.svg

https://de.wikipedia.org/wiki/Datei:Node.js_logo.svg

https://de.wikipedia.org/wiki/Datei:Angular_full_color_logo.svg

WebAssembly

http://webassembly.org/

Binary instruction format
for a stack-based VM
For Browser and beyond

Portable compilation target
for high-level languages
like C/C++/Rust

Open Standard

Why?
Performance
Safe

https://commons.wikimedia.org/wiki/File:Web_Assembly_Logo.svg

Some Facts about WASM

Very different from .NET's IL

- Much simpler
- Linear memory
- No GC

Cannot access the DOM = no UI

- (…yet)

JavaScript interop exists

- WASM calls JS
- JS calls into WASM

WASM and the CLR

C++ can be compiled into WASM

The .NET CLR is written in C++

Can the .NET CLR run on WASM?

Yes, it can – with mono

Blazor

Built on the Mono WASM Runtime

ASP.NET Razor Template Syntax

The .NET Core you know and love…

Demo

Demos

Anatomy of a Blazor app

JS Interop

Data Binding

Router

RESTful Web APIs

Anatomy of a Blazor App

dotnet command line

dotnet new blazor
dotnet build

Add to a new solution

dotnet new sln
dotnet sln add BlazorDemo.csproj

Publish Solution

dotnet publish -c Release -o out
Review content of *out* folder

VS2017

Open VS2017 and show how to create Blazor app there
Show Blazor language service extension
Open project in VS2017 (*start BlazorDemo.sln*)

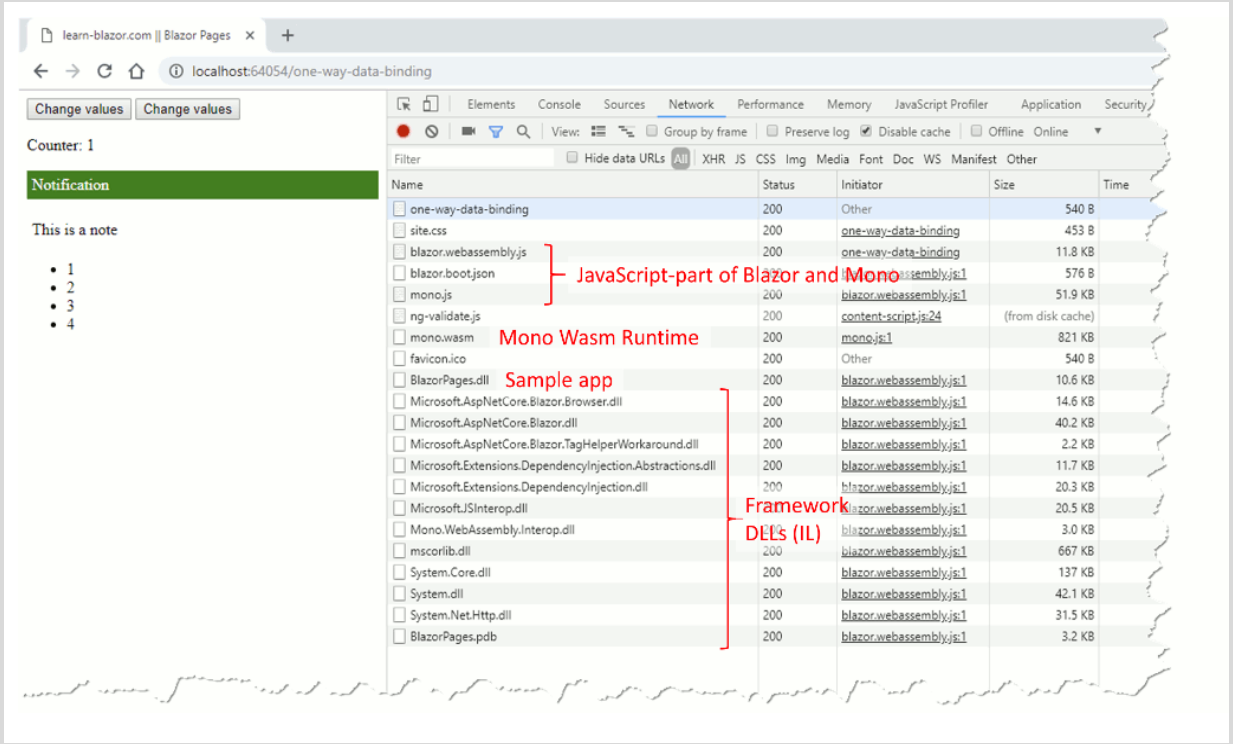
Running a SPA Blazor App

dotnet command line

- *dotnet blazor serve*
- F5 in Visual Studio – show *.csproj*
- Look at Network tab in Chrome Dev Tools

Static hosting

- Prove SPA nature by hosting app in *Chrome Dev Web Server* (*chrome://apps*)
- Speak about rewrite rules



Anatomy

of a Blazor App

Loading

- HTML, CSS, JS
- WASM (Mono)
- .NET DLLs

Hosting in ASP.NET Core

RestApi Sample

- Show and discuss *Startup.cs*
- *Microsoft.AspNetCore.Blazor.Server* in *UseBlazor<T>*
- Show and discuss shared library (*Shift+F12*)

Publish and discuss result

- *dotnet publish -c Release -o out*
- Run hosted app in Docker container: *docker run -v C:\Code\GitHub\learn-blazor\samples\RestApi\RestApi.Server\out:/app -w /app -p 8081:5000 microsoft/dotnet:2.1.4-aspnetcore-runtime-alpine dotnet RestApi.Server.dll*

Razor Walkthrough

Razor

- *Counter.cshtml* Razor file
- Show generated C# file *Counter.g.cs* → Razor becomes C#

BlazorComponent

- Speak about Components-based architecture
- Show *DynamicRenderTree* in *BlazorPages* app

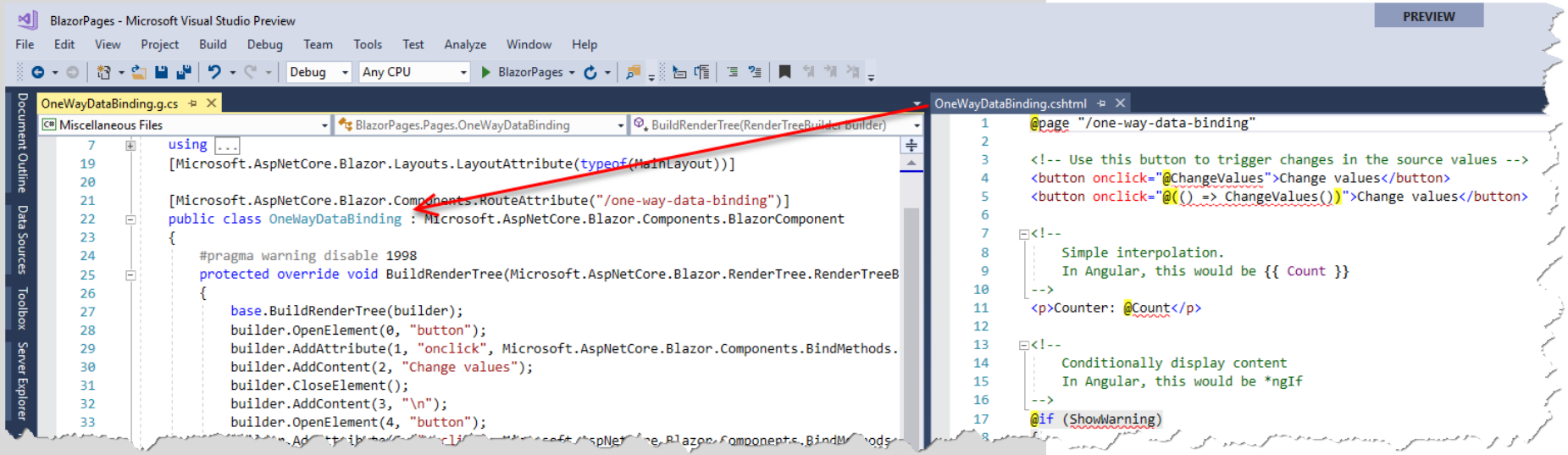
Blazor templates quick tour (*BlazorPages* sample)

- *OneWayDataBinding.cshtml*
- *TwoWayDataBinding.cshtml*
- *EventBinding.cshtml*
- *Initialization.cshtml*
- *ManualRefresh.cshtml*

Anatomy

of a Blazor App

Razor Code Generation



JavaScript Interop

Basics

Open *two-way-data-binding* in *BlazorPages* sample
Break on node removal at *You are an administrator*
Trigger node removal and speak about call stack

Coded JS Interop

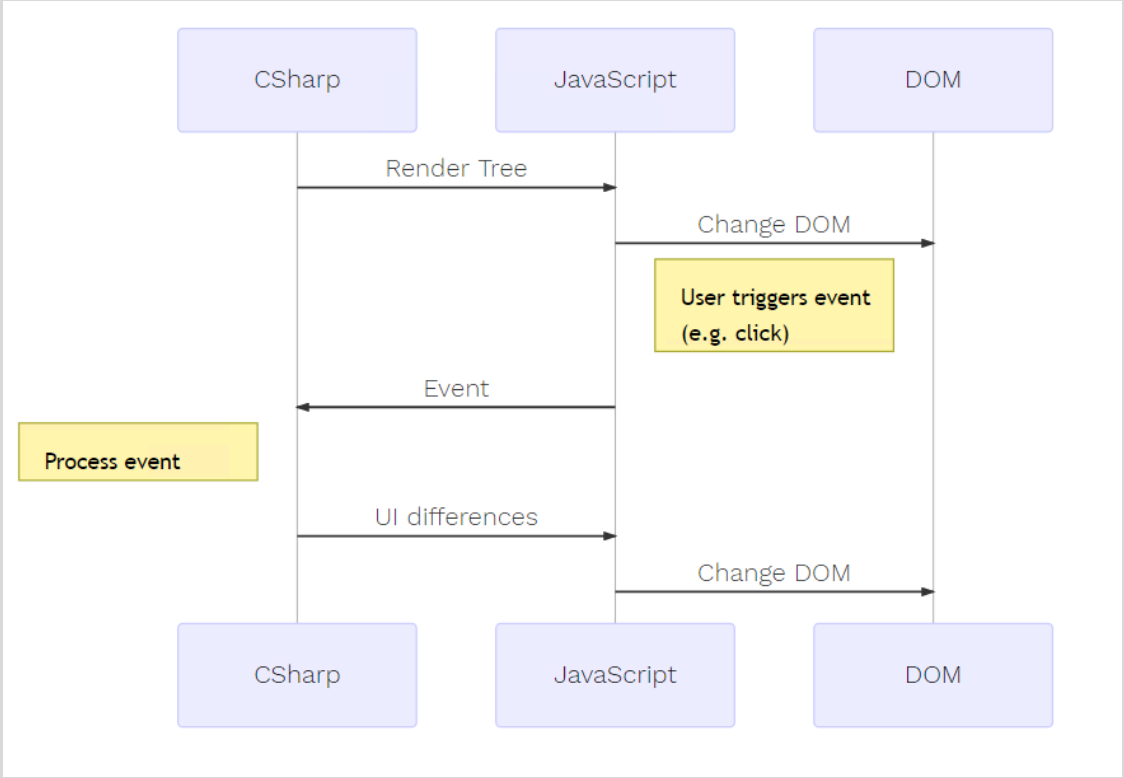
Open *interop-basics* in *RestApi* sample
Set breakpoint in *window.say*
Trigger breakpoint and speak about call stack

Open *auto-complete* in *RestApi* sample
Set breakpoints in *fillAutoComplete* and *select* callback
Trigger breakpoint and speak about call stack

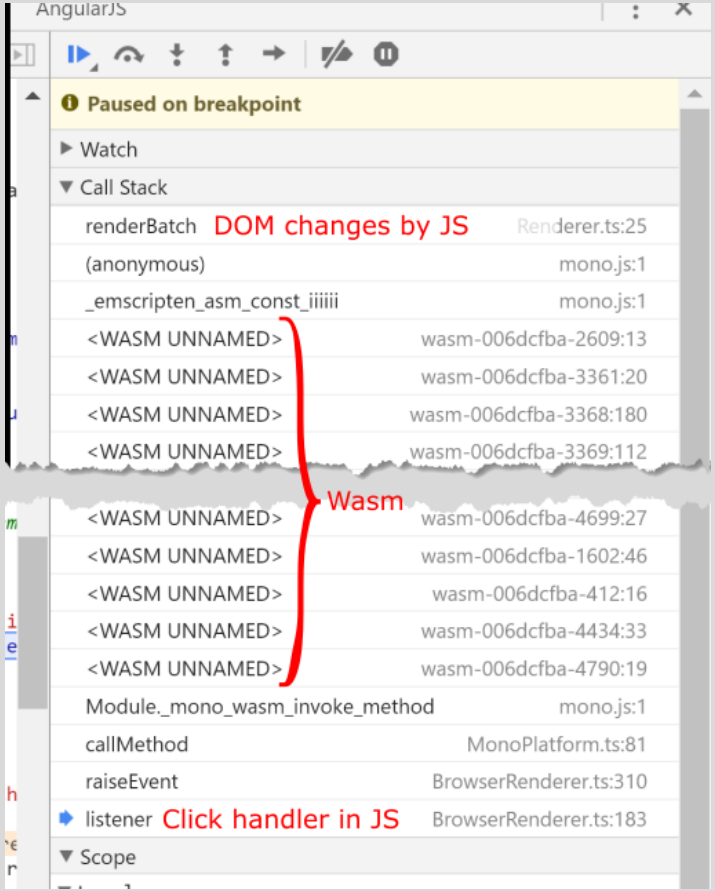
Anatomy

of a Blazor App

Rendering



JavaScript Intertop



Dependency Injection

Basics

- Open *Startup.cs* in *DependencyInjection* sample
- Open *CustomerList.cshtml* in *DependencyInjection* sample – *@inject*
- Speak about DI basics
- Open Repository.cs in *DependencyInjection* sample – constructor injection

HttpClient

- Open *FetchData.cshtml* in *RestApi* sample
- Speak about *HttpClient* standard service
  - *HttpClient(HttpMessageHandler)* Constructor
  - *HttpClient* creation in Blazor (Browser)
  - *BrowserHttpMessageHandler* in Blazor (Browser)
  - JS implementation using *fetch* API
- Show call stack for Web API calls in *RestApi* service

Router

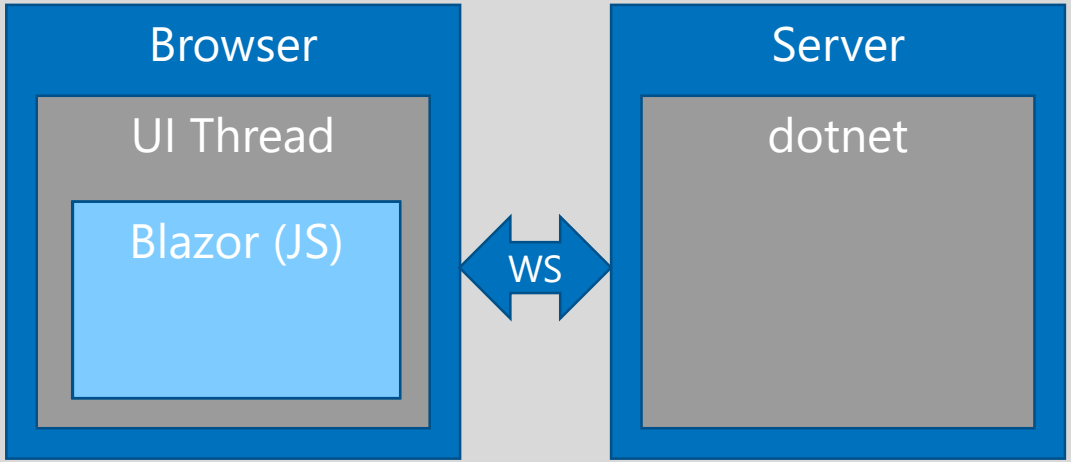
Basics (in *RouterDemo* sample)

HelloUniverse.cshtml
HelloPlanet.cshtml
HelloWorld.cshtml

Links

MainMenu.cshtml
Talk about *base* tag in *index.html*

Server-Side Hosting



Client-side

- All benefits of a SPA
- Restrictions because of WASM
- Maturity of tooling and runtime
- Larger initial download

Server-side

- Same Blazor programming model
- Full .NET environment
- Smaller initial download
- More server resources
- No offline support

Server-Side Hosting

Create new Blazor app with Server-Side Hosting

Code Walkthrough

- Show *blazor.server.js* reference in *index.html*
- Show *UseServerSideBlazor<T>* in *Startup.cs*

Debug

- Run app
- Show WebSockets traffic in Chrome Dev Tools
- Set breakpoint in counter increment, show it hitting

What else is in the box?

Debugging

Early prototype

Layouts

Master pages

Many details about component model

E.g. Child content

So what?

Is Blazor the Angular/React/Vue Killer?

Should I use it?

JS-based Frameworks

TypeScript
Huge ecosystem of tools and components
Mature
Feature-richness
Proven for small and large projects

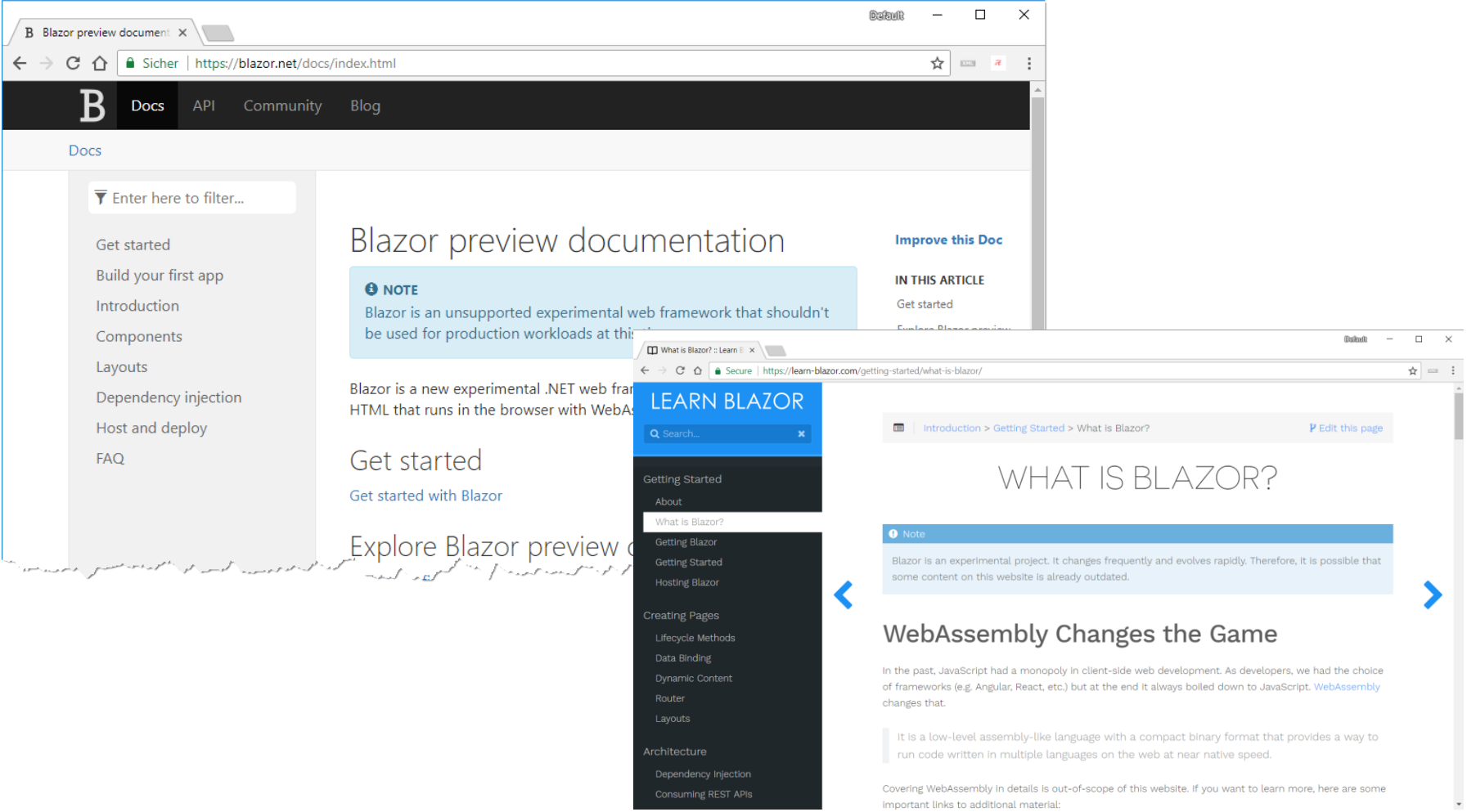
Web development skills necessary
No reuse of C# code possible

Blazor

C# and JavaScript/TypeScript
Limited community
Immature tools
Limited functionality
(B)Leading Edge

Less web development skills necessary
Reuse of C# code possible
Maturity of C#/.NET

Learning More…


Blazor preview documen
Sicher | https://blazor.net/docs/index.html
Docs
API
Community
Blog
Docs
Enter here to filter...
Get started
Build your first app
Introduction
Components
Layouts
Dependency injection
Host and deploy
FAQ
Blazor preview documentation
NOTE
Blazor is an unsupported experimental web framework that shouldn't be used for production workloads at
Blazor is a new experimental .NET web
HTML that runs in the browser with
Get started
Get started with Blazor
Explore Blazor preview
Improve this Doc
IN THIS ARTICLE
Get started
What is Blazor? :: Learn
Secure | https://learn-blazor.com/getting-started/what-is-blazor/
LEARN BLAZOR
Search...
Getting Started
About
What is Blazor?
Getting Blazor
Getting Started
Hosting Blazor
Creating Pages
Lifecycle Methods
Data Binding
Dynamic Content
Router
Layouts
Architecture
Dependency Injection
Consuming REST APIs
Introduction > Getting Started > What is Blazor?
Edit this page
WHAT IS BLAZOR?
Note
Blazor is an experimental project. It changes frequently and evolves rapidly. Therefore, it is possible that some content on this website is already outdated.
WebAssembly Changes the Game
In the past, JavaScript had a monopoly in client-side web development. As developers, we had the choice of frameworks (e.g. Angular, React, etc.) but at the end it always boiled down to JavaScript. WebAssembly changes that.
It is a low-level assembly-like language with a compact binary format that provides a way to run code written in multiple languages on the web at near native speed.
Covering WebAssembly in details is out-of-scope of this website. If you want to learn more, here are some important links to additional material:

International
JavaScript Conference

THANK YOU FOR COMING!

Q&A

Thank your for coming!

Rainer Stropek

software architects gmbh

Mail rainer@timecockpit.com
Web http://www.timecockpit.com
Twitter @rstropek

time cockpit
Saves the day.