



Blazor: Blazing A Trail For .NET Web Developers

Shaun Walker

shaun.walker@softvision.com

Shaun Walker

- Creator of DotNetNuke
- Founder of DNN Corp
- Microsoft MVP
- ASP Insider
- Chairman .NET Foundation Advisory Council



Technical Director & Enterprise Guildmaster
Cognizant Softvision

Cognizant
softvision



DOTNETNUKE



Agenda

- What is Blazor?
- Client-Side Blazor
- Server-Side Blazor
- Blazor Fundamentals
- Oqtane Demonstration

Blazor

Blazor is a single-page app framework
for building interactive web apps
with .NET Core.

Where It All Started...

Web Apps can't really do *that*, can they?
Steve Sanderson

NDC { Oslo }

12-16 June 2017

Inspiring Software Developers since 2008

Initial Announcement



Steve Sanderson's Blog

[HOME](#) [ABOUT](#) [TWITTER](#)

Blazor: a technical introduction

Deeper technical details about Blazor

Published Feb 6, 2018



Today [the ASP.NET team announced that Blazor has moved into the ASP.NET organization](#), and we're beginning an experimental phase to see whether we can develop it into a supported shipping product. This is a big step forwards!

Releases

<https://github.com/aspnet/Blazor/releases>

0.1.0 : Mar 22, 2018

0.2.0 : Apr 17, 2018*

0.2.1 : Apr 20, 2018

0.3.0 : May 2, 2018

0.4.0 : Jun 7, 2018

0.5.0 : Aug 10, 2018*

0.5.1 : Aug 10, 2018

0.6.0 : Oct 2, 2018

0.7.0 : Nov 15, 2018

0.8.0 : Feb 13, 2019

0.9.0 : Mar 7, 2019

3.0.0-P4 : Apr 18, 2019

3.0.0-P5 : May 6, 2019

3.0.0-P6 : June 2019

3.0.0-P7 : July 2019

3.0.0-RTM : Sep 2019

Official Preview

Blazor now in official preview!



Daniel

April 18th, 2019

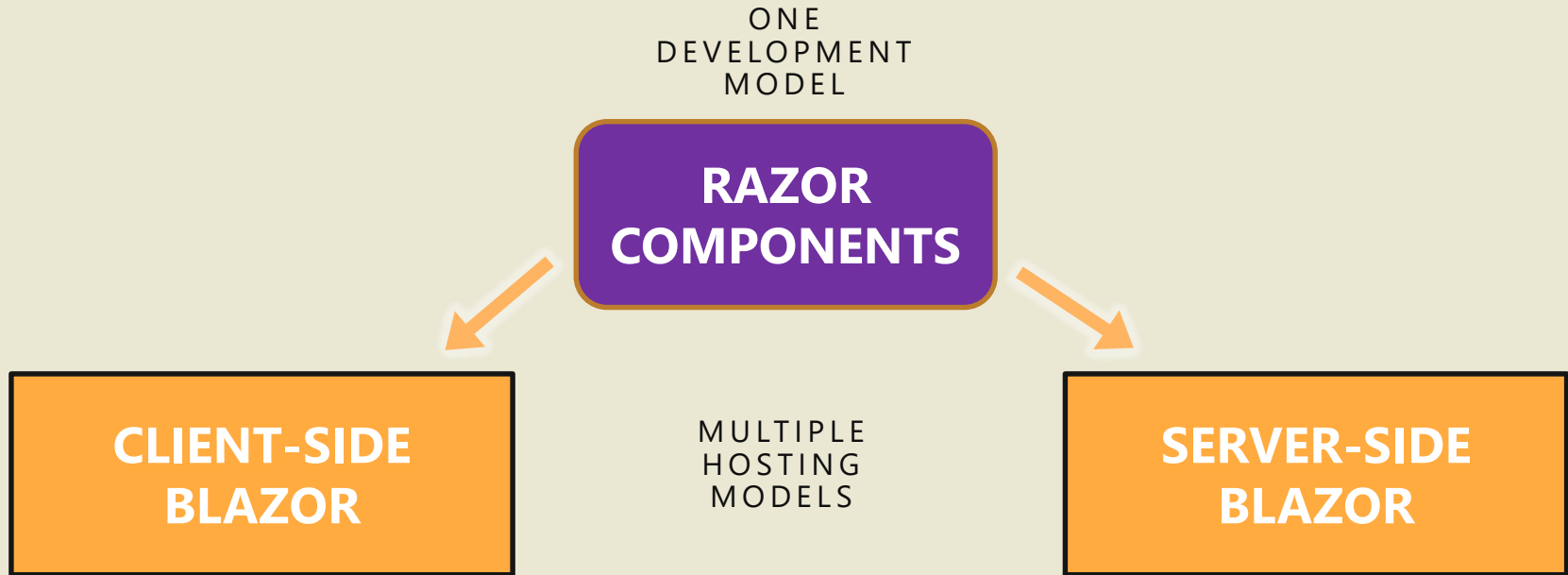
With this newest Blazor release we're pleased to announce that **Blazor is now in official preview!** Blazor is no longer experimental and we are committing to ship it as a supported web UI framework including support for running client-side in the browser on WebAssembly.

A little over a year ago we started the Blazor experimental project with the goal of building a client web UI framework based on .NET and WebAssembly. At the time Blazor was little more than a prototype and there were lots of open questions about the viability of running .NET in the browser. Since then we've shipped nine experimental Blazor releases addressing a variety of concerns including component model, data binding, event handling, routing, layouts, app size, hosting models, debugging, and tooling. We're now at the point where we think Blazor is ready to take its next step.

2 Flavors of Blazor



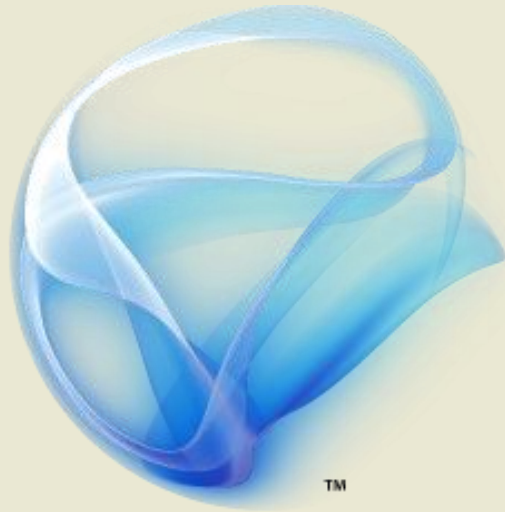
2 Flavors of Blazor





Client-Side Blazor

What it is Not!



Microsoft®
Silverlight™

Some History

Alternative Browser Runtimes



1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015

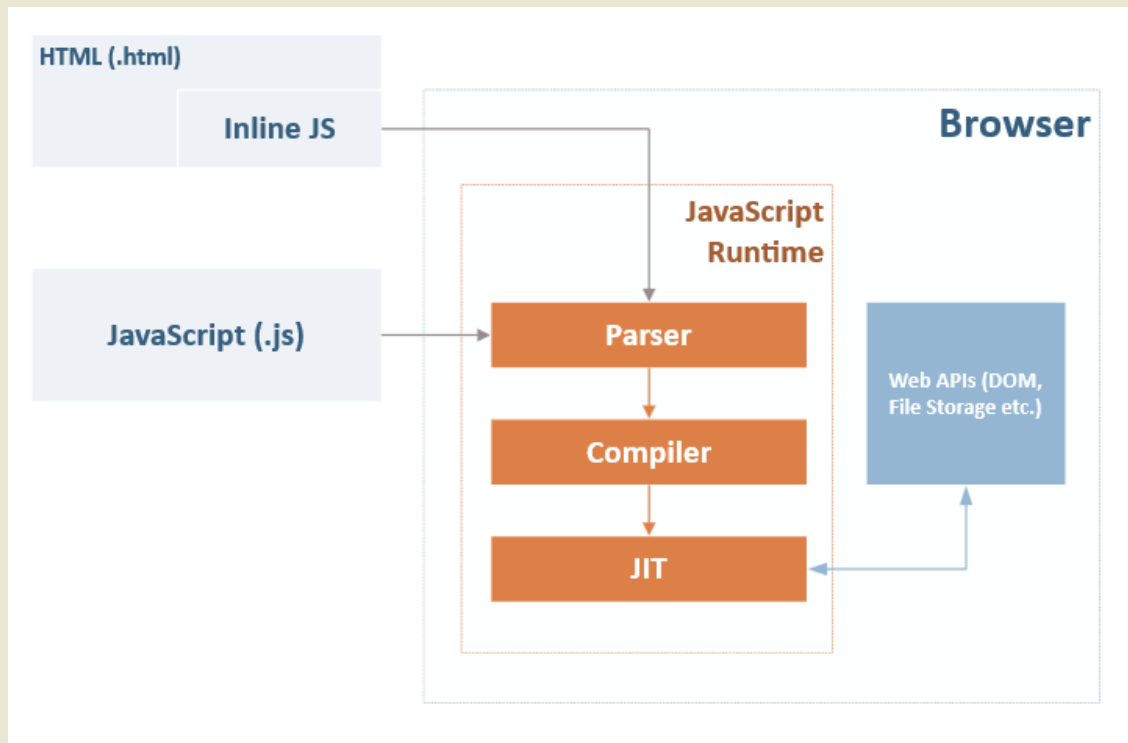


WebAssembly (Wasm)

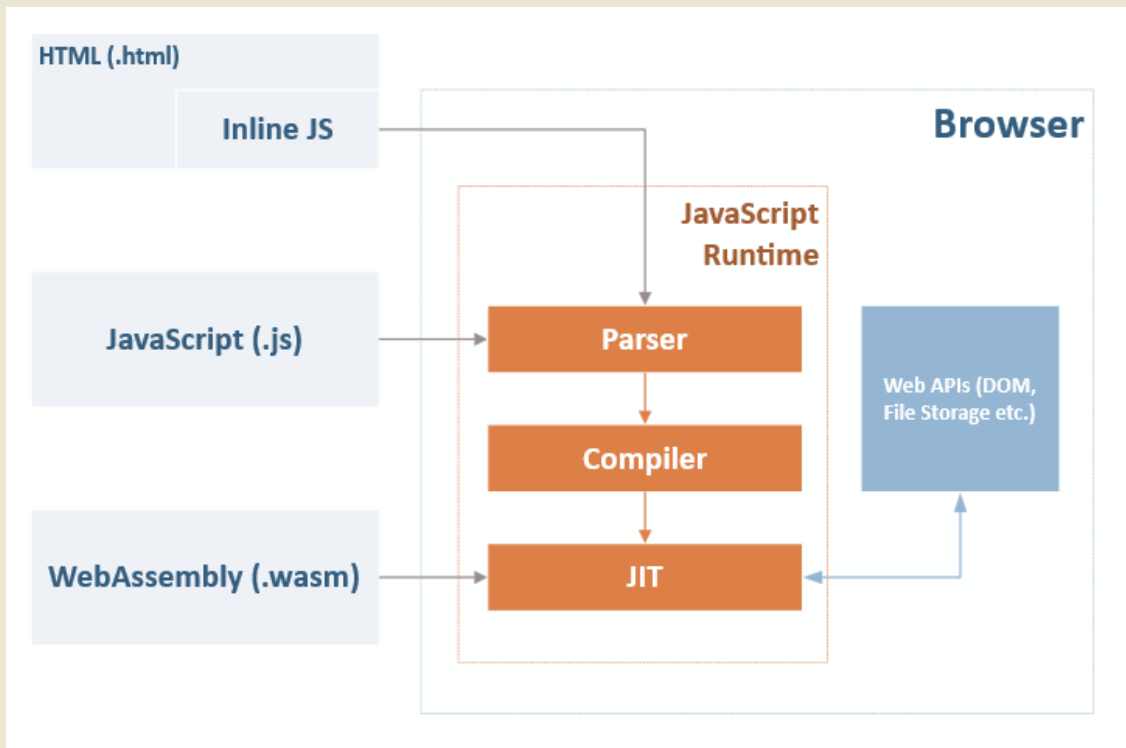
- WebAssembly (abbreviated as Wasm) is a binary instruction format for a stack-based virtual machine
- Originally introduced June 17, 2015
- The first language since JavaScript that is able to run natively in a web browser (without plug-ins)



How JavaScript Works



How WebAssembly Works

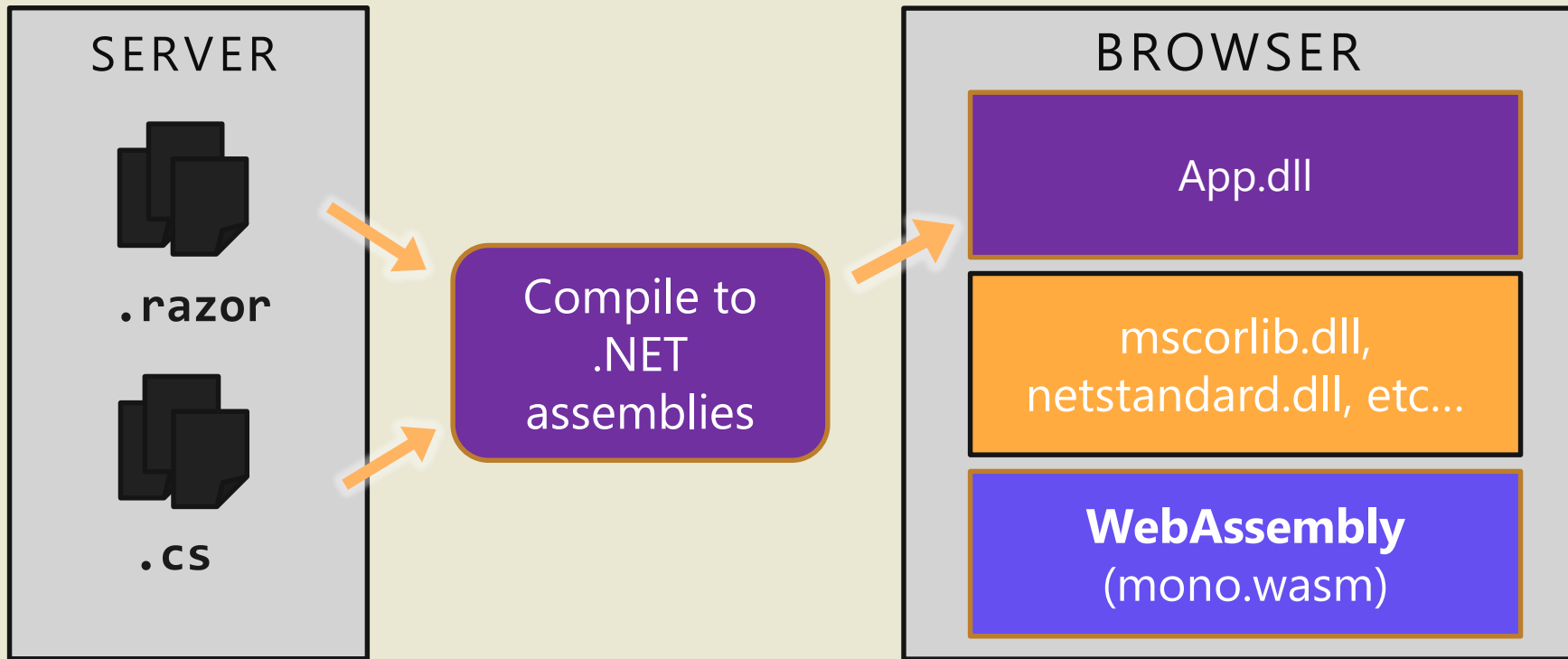


What About Mono?

- In order to run C# in the browser you need a .NET runtime that's been compiled to WebAssembly
- Microsoft's initial proof of concept used DotNetAnywhere
- Microsoft acquired Mono as part of Xamarin
- Mono.wasm is a version of Mono compiled to WebAssembly
- Mono.wasm is a full .NET runtime that can evaluate .NET assemblies



Client-Side Blazor



Why .NET in the Browser?

- JavaScript is the most popular programming language in the world
- With so many powerful front-end JavaScript frameworks (ie. Angular, React, Vue) why do we want to run .NET in the browser?

Because JavaScript SUCKS!!

Why .NET in the Browser?

- **Stable and consistent:** .NET offers standard APIs, tools, and build infrastructure across all .NET platforms that are stable, feature rich, and easy to use.
- **Modern innovative languages:** .NET languages like C# and F# make programming a joy and keep getting better with innovative new language features.

Why .NET in the Browser?

- **Industry leading tools:** The Visual Studio product family provides a great .NET development experience on Windows, Linux, and macOS.
- **Fast and scalable:** .NET has a long history of performance, reliability, and security for web development on the server.

Challenges

- Mono not yet optimized to act as a runtime in a browser
- Mono currently only supports .NET Standard 2.0
- Download sizes are currently very large
- Limited debugging support with current tooling
- Will not be supported in initial release of .NET Core 3.0

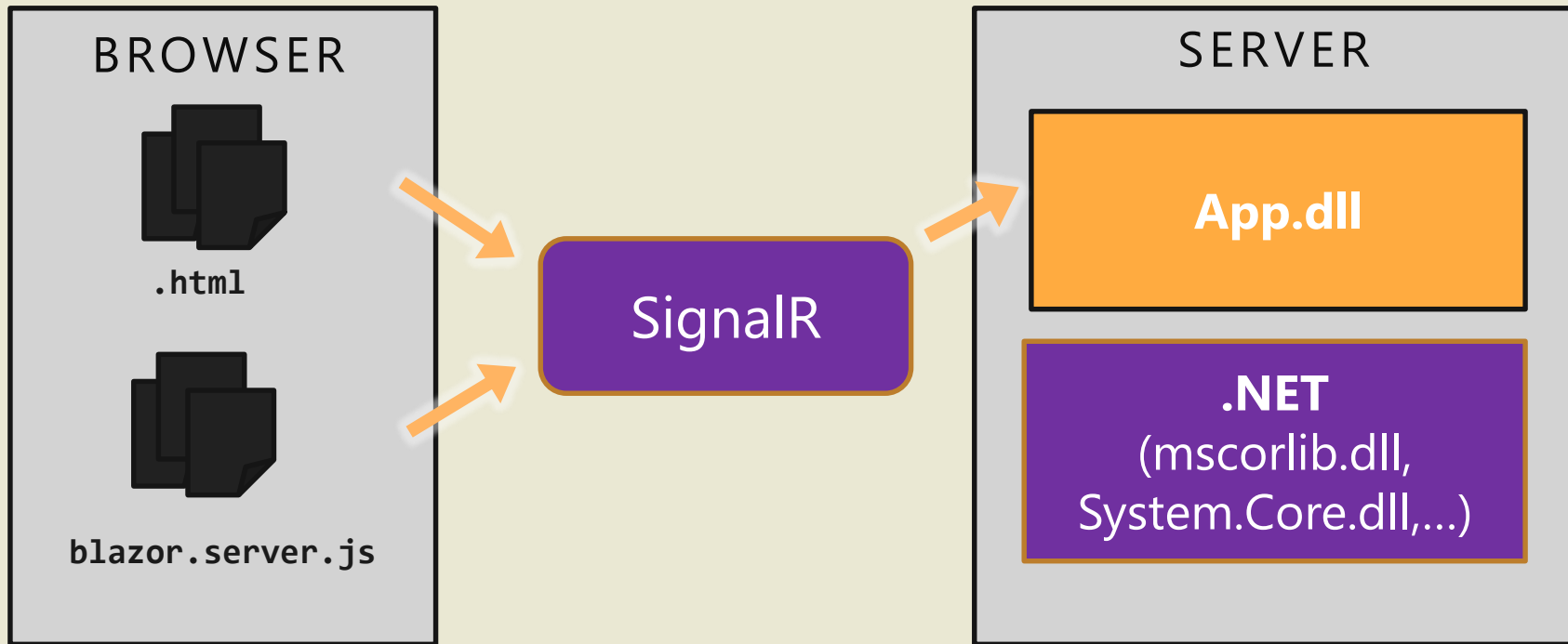


Server-Side Blazor

Server-Side Blazor

- Leverages the same Razor component development model
- Utilizes a server-side hosting model - does not rely on WebAssembly
- Uses SignalR for communication between browser and server

Server-Side Blazor



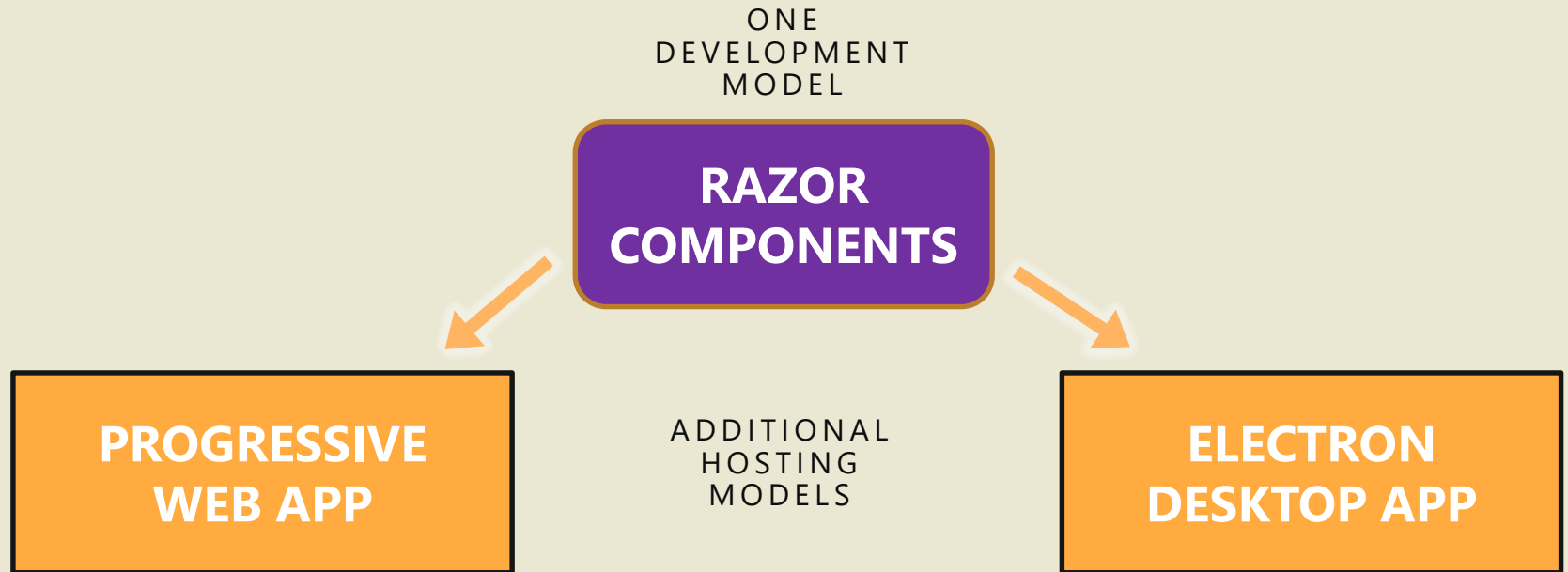
Server-Side Benefits

- Avoids unnecessary full page refreshes
- Smaller app download size and faster app startup
- Full access to all .NET Core APIs
- Debugging support using standard .NET tooling
- Supports legacy browsers with no WebAssembly support
- Will be officially supported in initial release of .NET Core 3.0

Challenges

- **Latency:** every user interaction now involves a network hop.
- **No offline support:** if the client connection goes down, the app stops working.
- **Scalability:** the server must manage multiple client connections and handle client state

Future Opportunities



(these examples were already demonstrated by Steve Sanderson at MVP Summit 2019)

Blazor Fundamentals

- Getting Started
- Component Model
- Routing
- Layouts
- Data & Event Binding
- Dependency Injection
- Javascript Interop

Getting Started

Installation Instructions

- .NET Core 3.0 Preview 4 SDK (3.0.100-preview4-011223))
- Visual Studio 2019 **Preview** with “ASP.NET and Web Development Workload”
- The latest **Blazor Extension** from the Visual Studio Marketplace
- Enable Visual Studio to use preview SDKs: Open **Tools > Options** in the menu bar. Open the **Environment** node. Open the **Preview Features** tab. Check the box for **Use previews of the .NET Core SDK**. Select OK.

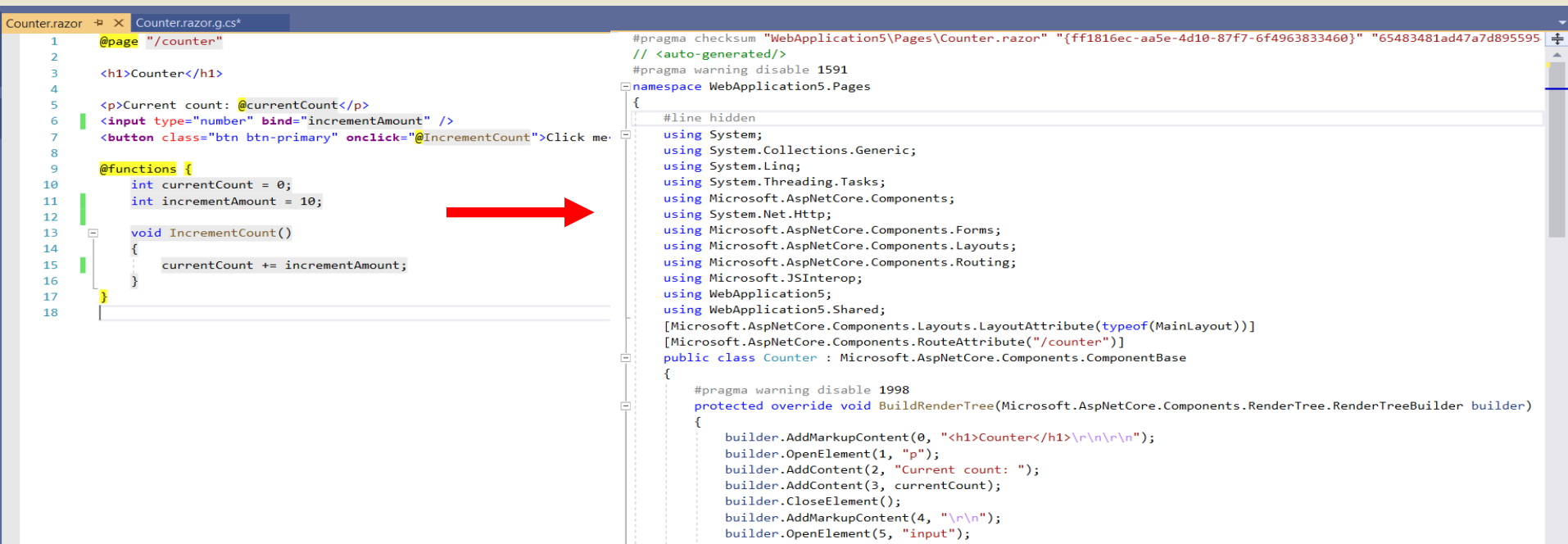
Component Model

- Razor Components are self-contained chunks of user interface (UI), such as a page, dialog, or form.
- Components includes HTML markup and the processing logic required to inject data or respond to UI events.
- Components are flexible and lightweight. They can be nested, reused, and shared among projects.

Component Model

Razor is converted into a component class during compilation (*.razor >> *.razor.g.cs)

\\WebApplication.Client\obj\Debug\netstandard2.0\Razor*

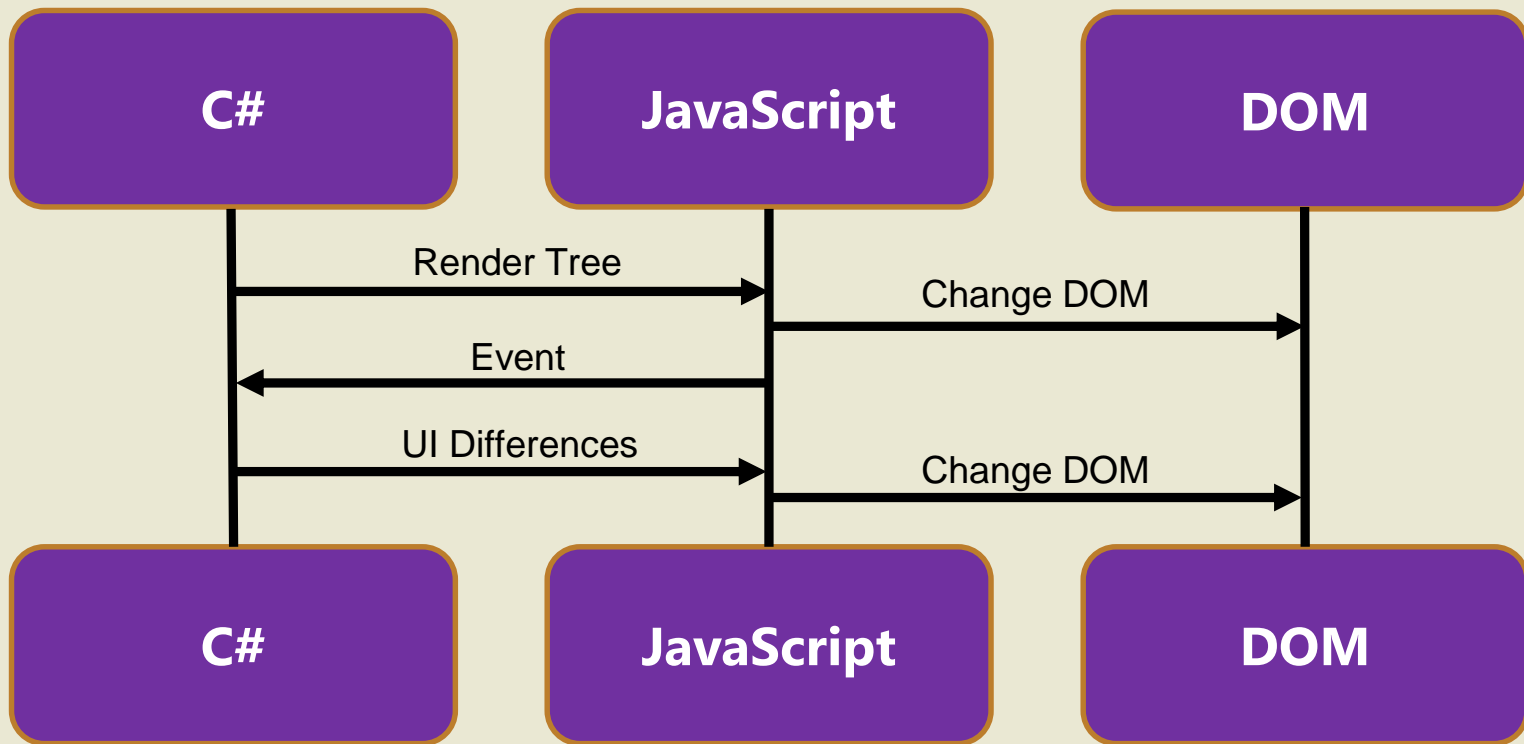


```
Counter.razor  Counter.razor.g.cs*
1  @page "/counter"
2
3  <h1>Counter</h1>
4
5  <p>Current count: @currentCount</p>
6  <input type="number" bind="incrementAmount" />
7  <button class="btn btn-primary" onclick="@IncrementCount">Click me</button>
8
9  @functions {
10     int currentCount = 0;
11     int incrementAmount = 10;
12
13     void IncrementCount()
14     {
15         currentCount += incrementAmount;
16     }
17 }
18
```

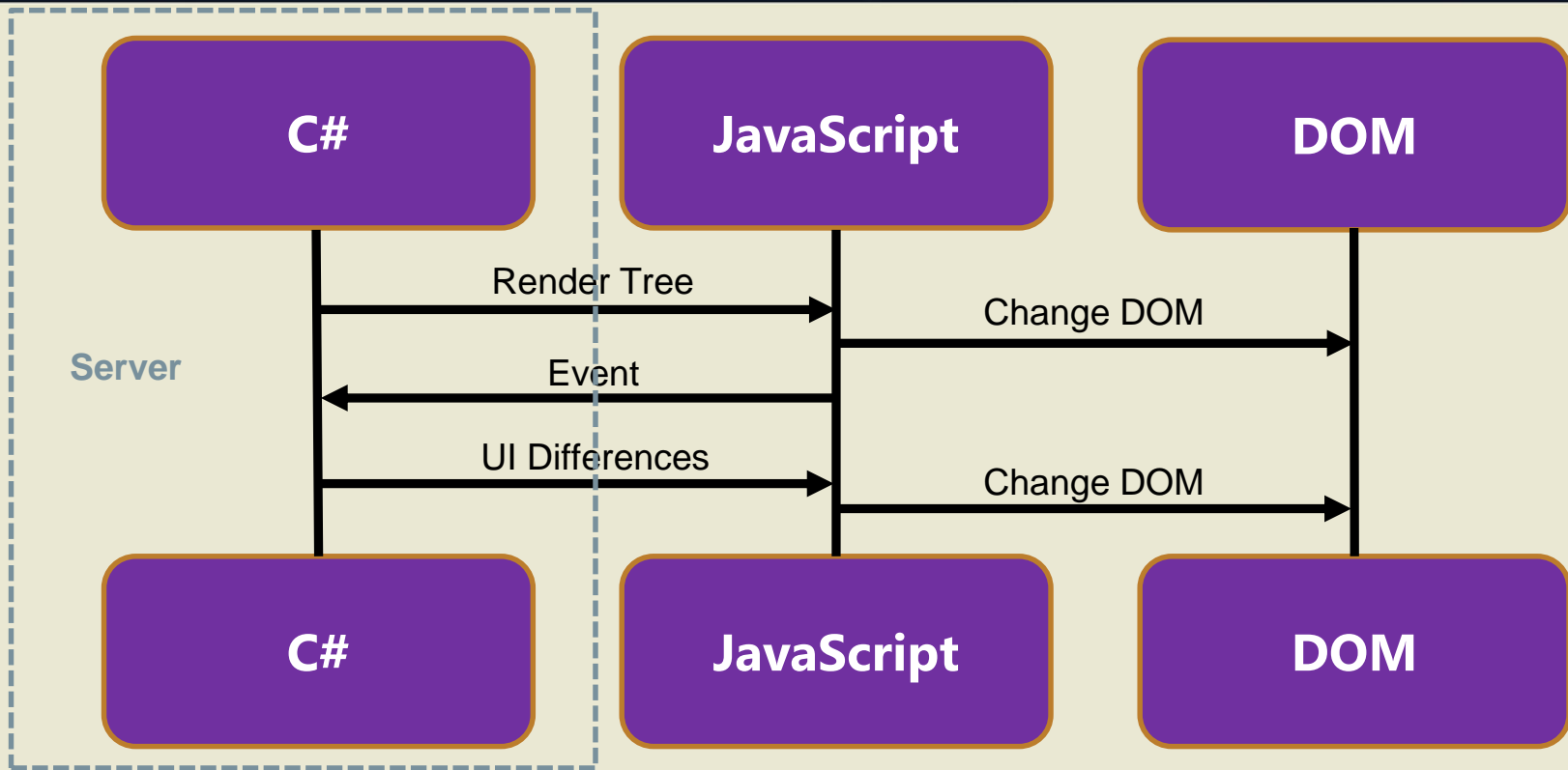
```
#pragma checksum "WebApplication5\Pages\Counter.razor" "{ff1816ec-aa5e-4d10-87f7-6f4963833460}" "65483481ad47a7d895595"
// <auto-generated/>
#pragma warning disable 1591
namespace WebApplication5.Pages
{
    #line hidden
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Threading.Tasks;
    using Microsoft.AspNetCore.Components;
    using System.Net.Http;
    using Microsoft.AspNetCore.Components.Forms;
    using Microsoft.AspNetCore.Components.Layouts;
    using Microsoft.AspNetCore.Components.Routing;
    using Microsoft.JSInterop;
    using WebApplication5;
    using WebApplication5.Shared;
    [Microsoft.AspNetCore.Components.Layouts.LayoutAttribute(typeof(MainLayout))]
    [Microsoft.AspNetCore.Components.RouteAttribute("/counter")]
    public class Counter : Microsoft.AspNetCore.Components.ComponentBase
    {
        #pragma warning disable 1998
        protected override void BuildRenderTree(Microsoft.AspNetCore.Components.RenderTree.RenderTreeBuilder builder)
        {
            builder.AddMarkupContent(0, "<h1>Counter</h1>\r\n\r\n");
            builder.OpenElement(1, "p");
            builder.AddContent(2, "Current count: ");
            builder.AddContent(3, currentCount);
            builder.CloseElement();
            builder.AddMarkupContent(4, "\r\n");
            builder.OpenElement(5, "input");

```


Component Rendering



Server-Side Rendering



Using Components

Components use familiar HTML syntax and can reference other components:

```
<TitleComponent Title="Wow!" />
```

Components can accept Parameters using non-public properties:

```
<h1>@Title</h1>  
@functions {  
    [Parameter]  
    private string Title { get; set; }  
}
```

Client-side Routing

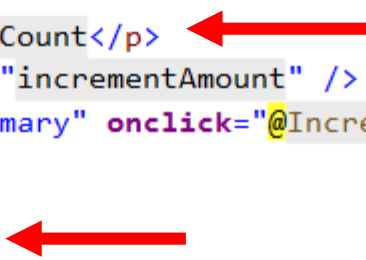
```
1  @page "/counter" ←
2
3  <h1>Counter</h1>
4
5  <p>Current count: @currentCount</p>
6  <input type="number" bind="incrementAmount" />
7  <button class="btn btn-primary" onclick="@IncrementCount">Click me</button>
8
9  @functions {
10     int currentCount = 0;
11     int incrementAmount = 10;
12
13     void IncrementCount()
14     {
15         currentCount += incrementAmount;
16     }
17 }
```

Layouts

```
1  @inherits LayoutComponentBase ←
2
3  <div class="sidebar">
4      <NavMenu />
5  </div>
6
7  <div class="main">
8      <div class="top-row px-4">
9          <a href="http://blazor.net" target="_blank" class="ml-md-auto">About</a>
10     </div>
11
12     <div class="content px-4">
13         @Body ←
14     </div>
15 </div>
16
```

One Way Data Binding

```
1  @page "/counter"
2
3  <h1>Counter</h1>
4
5  <p>Current count: @currentCount</p>
6  <input type="number" bind="incrementAmount" />
7  <button class="btn btn-primary" onclick="@IncrementCount">Click me</button>
8
9  @functions {
10     int currentCount = 0;
11     int incrementAmount = 10;
12
13     void IncrementCount()
14     {
15         currentCount += incrementAmount;
16     }
17 }
```

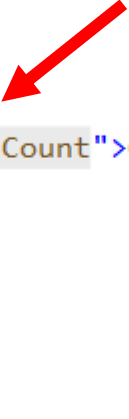


Two Way Data Binding

```
1  @page  "/counter"
2
3  <h1>Counter</h1>
4
5  <p>Current count: @currentCount</p>
6  <input type="number" bind="incrementAmount" /> ←
7  <button class="btn btn-primary" onclick="@IncrementCount">Click me</button>
8
9  @functions {
10     int currentCount = 0;
11     int incrementAmount = 10; ←
12
13     void IncrementCount()
14     {
15         currentCount += incrementAmount;
16     }
17 }
```

Event Binding

```
1  @page  "/counter"
2
3  <h1>Counter</h1>
4
5  <p>Current count: @currentCount</p>
6  <input type="number" bind="incrementAmount" />
7  <button class="btn btn-primary" onclick="@IncrementCount">Click me</button>
8
9  @functions {
10     int currentCount = 0;
11     int incrementAmount = 10;
12
13     void IncrementCount()
14     {
15         currentCount += incrementAmount;
16     }
17 }
```



Dependency Injection

```
1  @page "/fetchdata"
2  @inject HttpClient Http ←
3
4  <h1>Weather forecast</h1>
5
6  <p>This component demonstrates fetching data from the server.</p>
7
8  @if (forecasts == null)
9  {
10     <p><em>Loading...</em></p>
11 }
```

JavaScript Interop

- Call JavaScript from C# or call C# from JavaScript

```
@inject IJSRuntime JsRuntime;
```

```
<p>Cookie Value: @value</p>
```

```
<button class="btn btn-primary" onclick="@Getvalue">Click me</button>
```

```
@functions {
```

```
    string value = "";
```

```
    private async void Getvalue()
```

```
{
```

```
        var value = await JsRuntime.InvokeAsync<string>("GetCookie", "MyCookie");
```

```
}
```

```
}
```

Write Once, Run Anywhere?

- It is possible to target multiple hosting models; however, you need to architect your application appropriately
- Client vs. Server Workloads
- .NET Standard 2.0 vs .NET Core 3.0
- HttpClient/Json Parser
- Async services



Demonstration

Introducing....

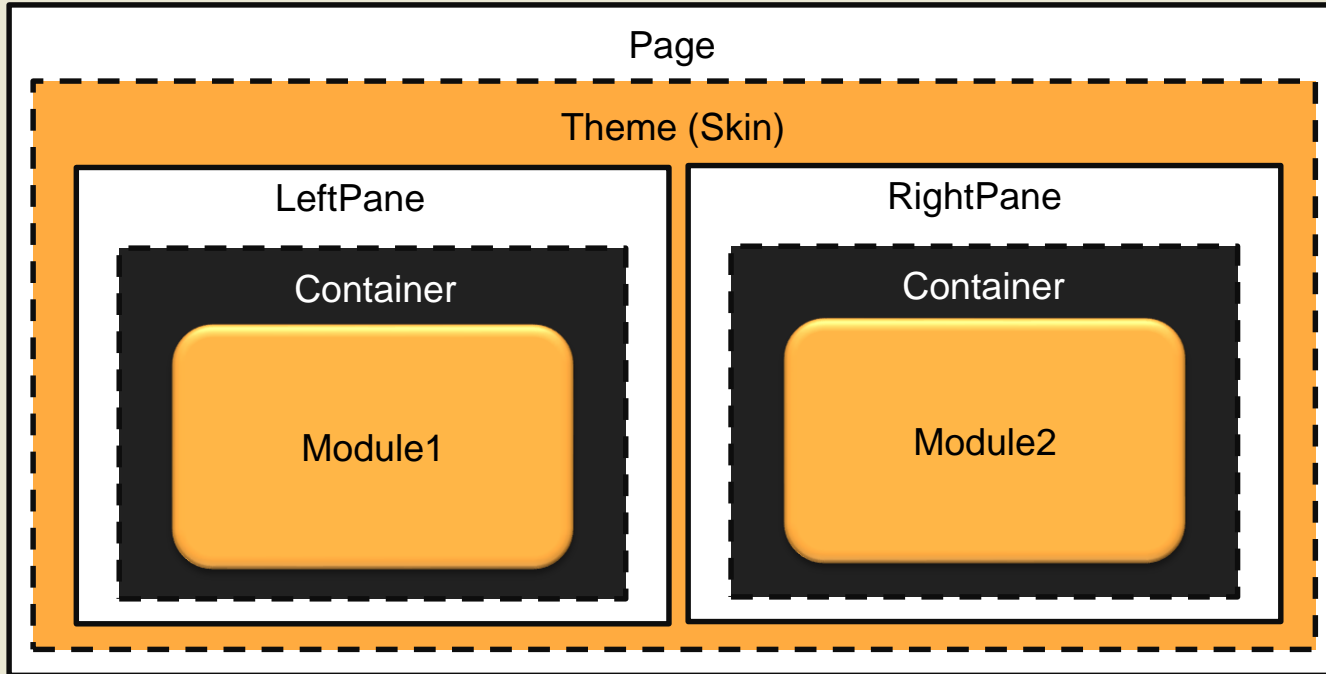


A Modular Application Framework for Blazor

Oqtane

- Inspired by DotNetNuke (but NOT a CMS)
- Written natively using Blazor
- Familiar page compositing/dashboard model
- Multi-tenant
- Designer friendly themes (skins)
- Distributable modules
- Open source

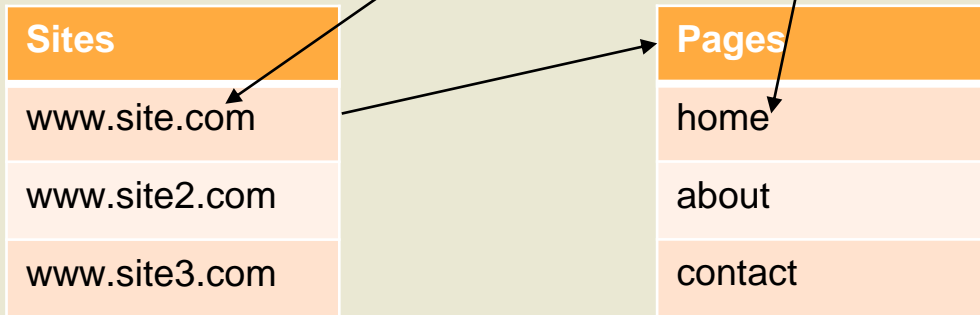
Page Compositing



Multi-Tenant

- Multi-site based on hostname/subfolder
- Virtual pages based on Url Path

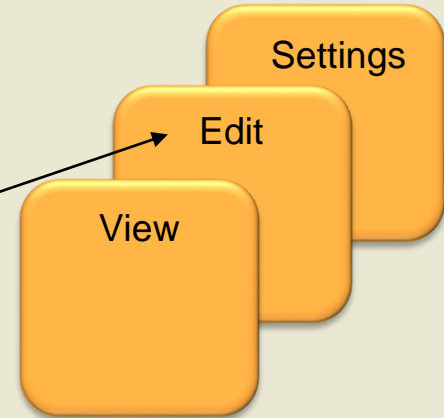
http://www.site.com/home



Routing

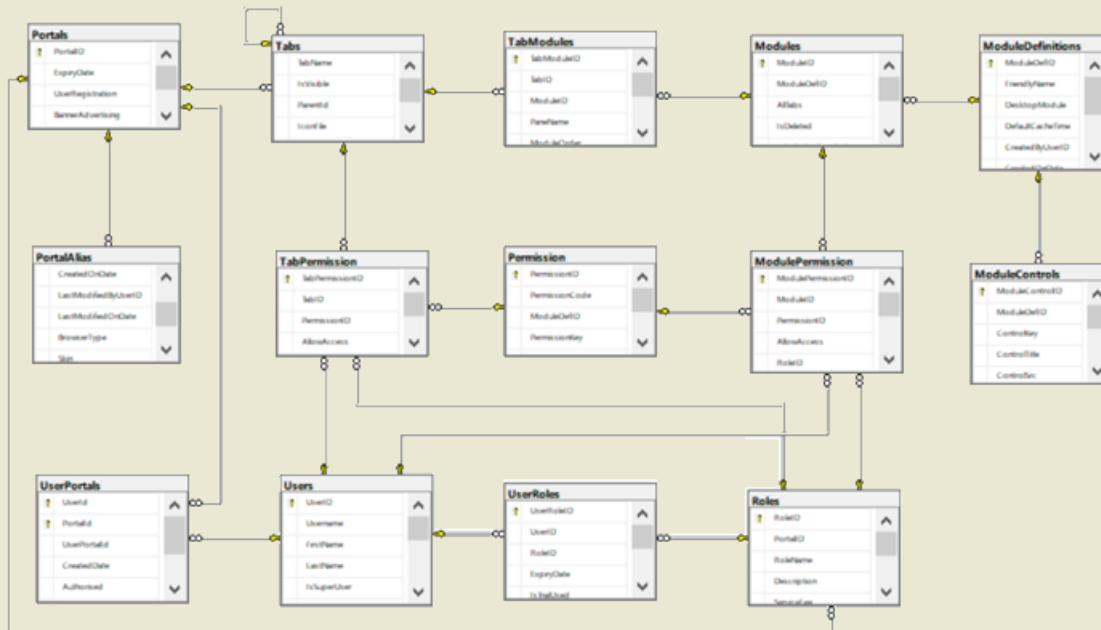
- Dynamic Page Routing
- Module Routing via Querystring (?mid=1&ctl=edit)

ModuleID	Key	Source
1	(default)	/FolderPath/View.ascx
1	Edit	/FolderPath/Edit.ascx
1	Settings	/FolderPath/Settings.ascx



Database

- Leverage “Core” DNN Data Model (~20 tables)



Technical Details

- Supports both client and server Blazor hosting models
- Custom Router
- Dynamic Components
- Cascading Parameters
- “Headless” API
- Repository Pattern

Framework Enhancements

- API Simplification
- Modern Entity Naming Convention (ie. Tab = Page)
- Isolated Tenants (ie. Separate Databases)
- Auto Module Registration
- Theme Pane Layouts
- Eliminate Requirement for Admin Skin

Support from DNN Community

- Michael Washington (Install Wizard)
- David Poindexter (Bootstrap)
- Mitch Sellers (Rename Skin to Theme)
- Charles Nurse (SQLite research)

Support From Microsoft

From: Steve Sanderson ·
Sent: Monday, May 6, 2019 4:17 AM
To: Daniel Roth; Shaun Walker ·
Cc: Ryan Nowak
Subject: RE: Clientside Blazor

Nice! Glad to see this is making its way out into the world 😊

Steve

From: Daniel Roth
Sent: 06 May 2019 04:23
To: Shaun Walker; Steve Sanderson
Cc: Ryan Nowak
Subject: RE: Clientside Blazor

Sweet! Congrats on releasing to GitHub!

Oqtane Community & Code

<https://www.oqtane.org/>

<https://github.com/oqtane/oqtane.framework>

oqtane / oqtane.framework

Watch 26 Unstar 102 Fork 16

Code Issues 6 Pull requests 0 Projects 0 Wiki Security Insights Settings

Modular Application Framework for Blazor <http://www.oqtane.org> Edit

Manage topics

49 commits 1 branch 0 releases 3 contributors MIT

Branch: master New pull request Create new file Upload files Find File Clone or download

sbwalker	Updated Enable Visual Studio to use preview SDKs ...	Latest commit bfa3877 7 days ago
Oqtane.Client	Performance improvements, refactoring of multi-tenant support, split ...	11 days ago
Oqtane.Server	small fix to register new services in DI container for client-side BL...	11 days ago
Oqtane.Shared	Performance improvements, refactoring of multi-tenant support, split ...	11 days ago
.gitignore	Add initial .gitignore file	29 days ago
LICENSE	Update LICENSE	a month ago



Thank You!