Blazor for JavaScript Developers

Tim Purdum

DevUp Conf

August, 2025

esri Partner Network Gold

Esri Partner Conference 2023 Award Winner
GIS for Good

Federal Small Business Specialty

ArcGIS Hub Specialty

Nonprofit Specialty

State and Local Government Specialty

WOMEN OWNED

SBA EDWOSB CERTIFIED

You know, I'm something of a web developer myself

Goals of the Session

- Learn about Blazor
    - Getting started
    - Unique features and functionality
    - Compare & contrast to JavaScript frameworks
    - Pitfalls and drawbacks
    - How it ties into the Asp.NET Core back-end Ecosystem
    - How you can write interop code between Blazor and JS

Getting Started

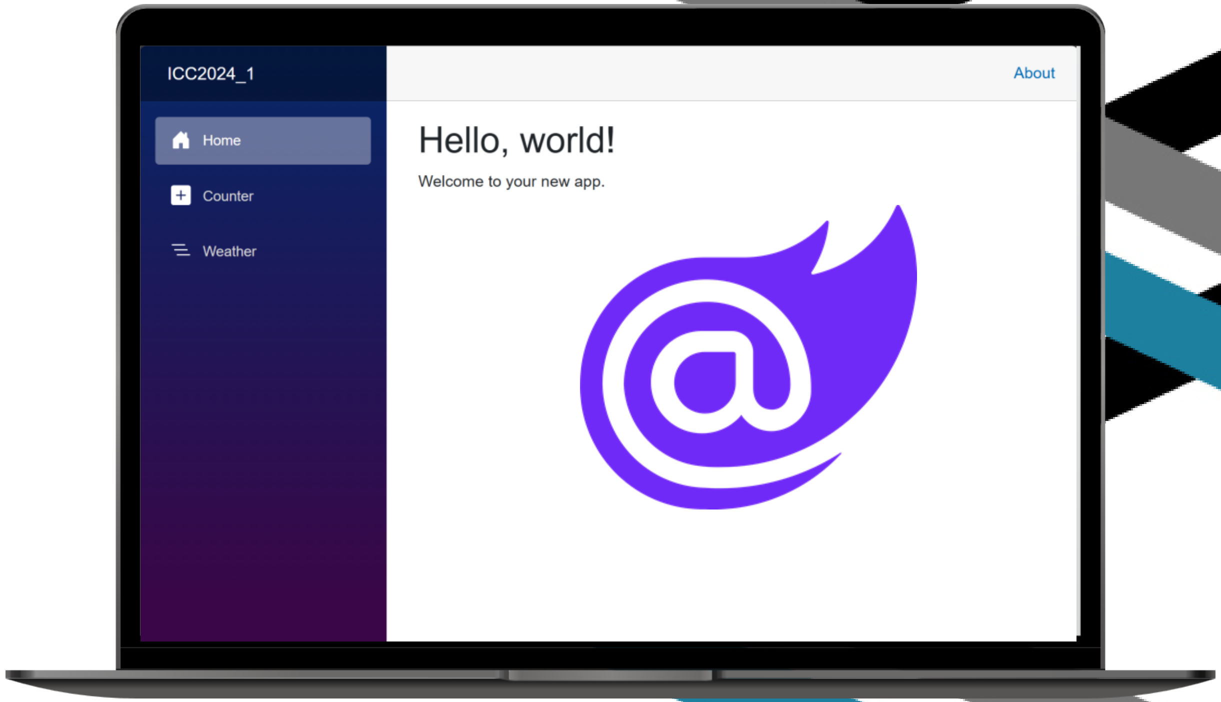
- Get .NET
  - Download from https://dotnet.microsoft.com
  - winget install Microsoft.DotNet.SDK.9
- Get Language Support
  - VS Code + VS License – C# Dev Kit Extension
  - Visual Studio - Community Edition (free), Professional, Enterprise
  - JetBrains Rider – Free Community License, Paid Prof. License

Getting Started (2)

- `dotnet new list` – shows all the available templates
- `dotnet new blazor -h` – displays help for completions
- `dotnet new blazor -int Auto -au Individual -o HelloWorld`
  - `-int Auto` – auto interactive mode
  - `-au Individual` – adds individual authentication user accounts
  - `-o HelloWorld` – sets the name and output folder of the project
- `cd HelloWorld` – navigate
- `code .` – open the folder for editing

What is Blazor?

- Modern full-stack web framework
- Built on Asp.NET Core and Modern .NET
- Robust, production-ready solution since .NET Core 3.1 in 2018
- Static and dynamic Server-Side rendering
- Client WebAssembly SPA applications or individual components
- High productivity with a single unifying language and framework
- Hot-reload == rapid development with robust dev tools



Hello, world!

Welcome to your new app.

C#

- Developed at Microsoft by Anders Hejlsberg, who also developed TypeScript
- Like TypeScript, C# provides compile-time guarantees of type safety, but C# also enforces at runtime.
- Unlike TypeScript, which is transpiled to JavaScript and interpreted at runtime, C# is
  - *compiled* to Intermediate Language (IL), and then, either
    - run with a Just-in-Time Compiler (JIT), OR
    - Ahead-of-Time Compiled to native machine code

.NET

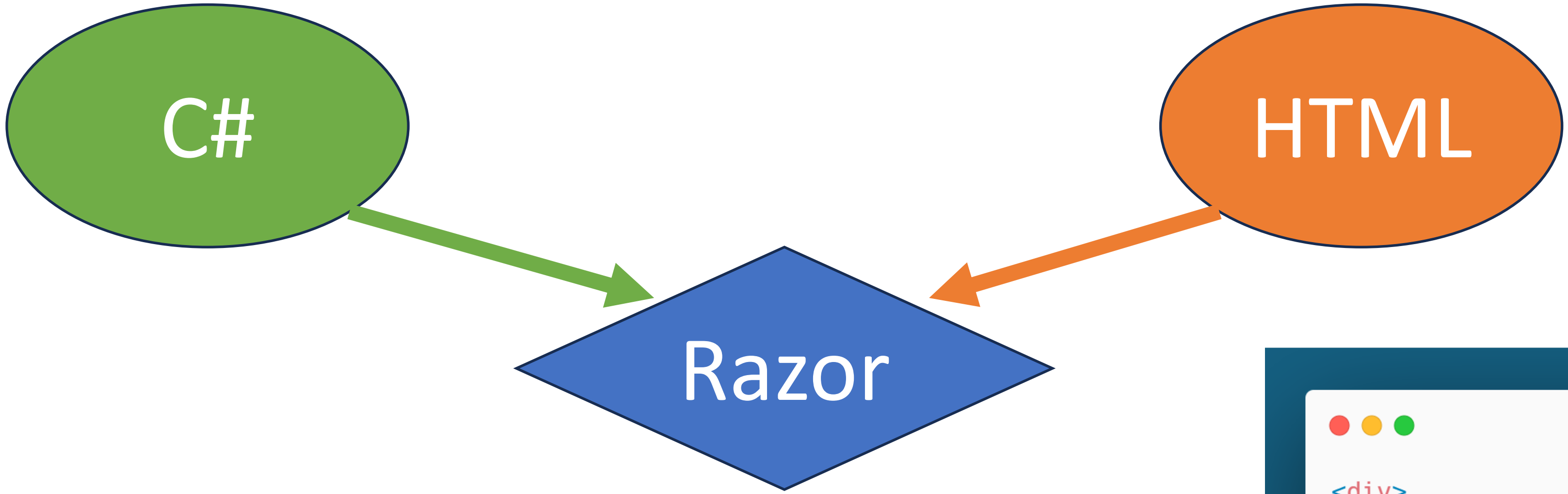
- .NET Framework (1.0 – 4.8.1)
  - Microsoft, 2000
  - Proprietary
  - Tied to Windows
- Mono
  - Ximian, 2001
  - Open-source re-implementation of .NET Framework for other platforms
  - Foundation of what later became Xamarin
- .NET Core (1.0 – 3.1)
  - Microsoft, 2016
  - Open-source
  - Runs on Windows, Linux, Mac, iOS, Android
  - Roslyn compiler written in C#
- .NET (5.0+)
  - Rebranding, merging of .NET Core and Mono/Xamarin code bases
  - Focus on performance

DON'T PANIC

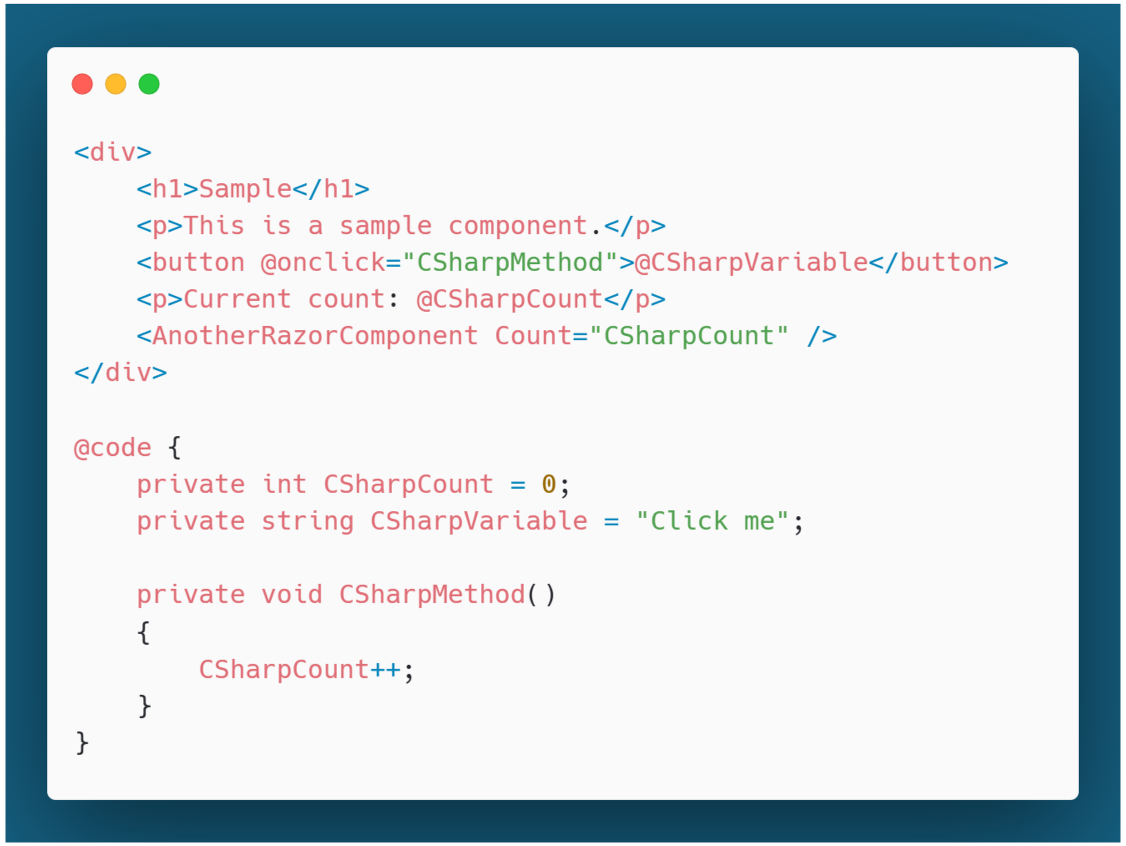
Blazor Supports Modern Web Standards

- HTML
  - Full support, only changes would be to escape `@` text characters
  - Can create nested components primarily out of HTML simply for organizational structure, even if not using other functionality
  - Declarative head tags, links, scripts, and component-level injection of extra head content, links, and scripts
- CSS
  - Inline support
  - ***/wwwroot*** files imported with link tags
  - *Scoped* CSS files per component: e.g., `Component1.razor` => `Component1.razor.css`
- W {
  - {
  - L / `IJSRuntime` L w

Razor (the Syntax and Components behind Blazor)



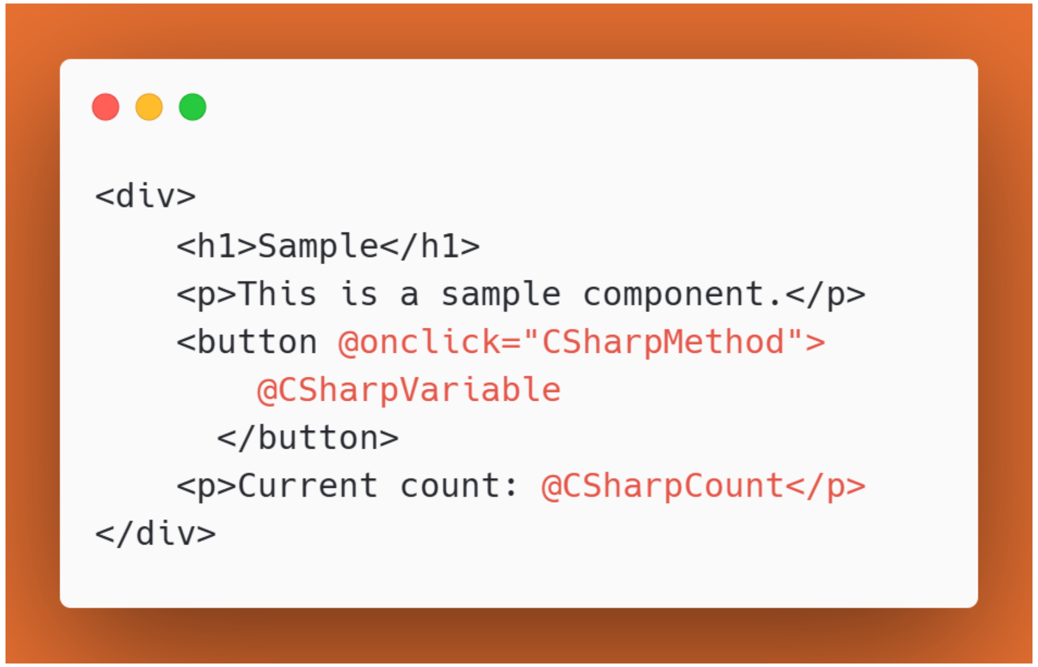
- `.razor` file extension
- Encapsulate UI and functionality
- Reusable and composable
- Each one can run client-side or server-side
- @ symbol identifies start of C# code
- Parentheses and Braces define code scopes
- Markup tags can be nested inside conditional logic on new lines



```
<div>
    <h1>Sample</h1>
    <p>This is a sample component.</p>
    <button @onclick="CSharpMethod">@CSharpVariable</button>
    <p>Current count: @CSharpCount</p>
    <AnotherRazorComponent Count="CSharpCount" />
</div>

@code {
    private int CSharpCount = 0;
    private string CSharpVariable = "Click me";

    private void CSharpMethod()
    {
        CSharpCount++;
    }
}
```

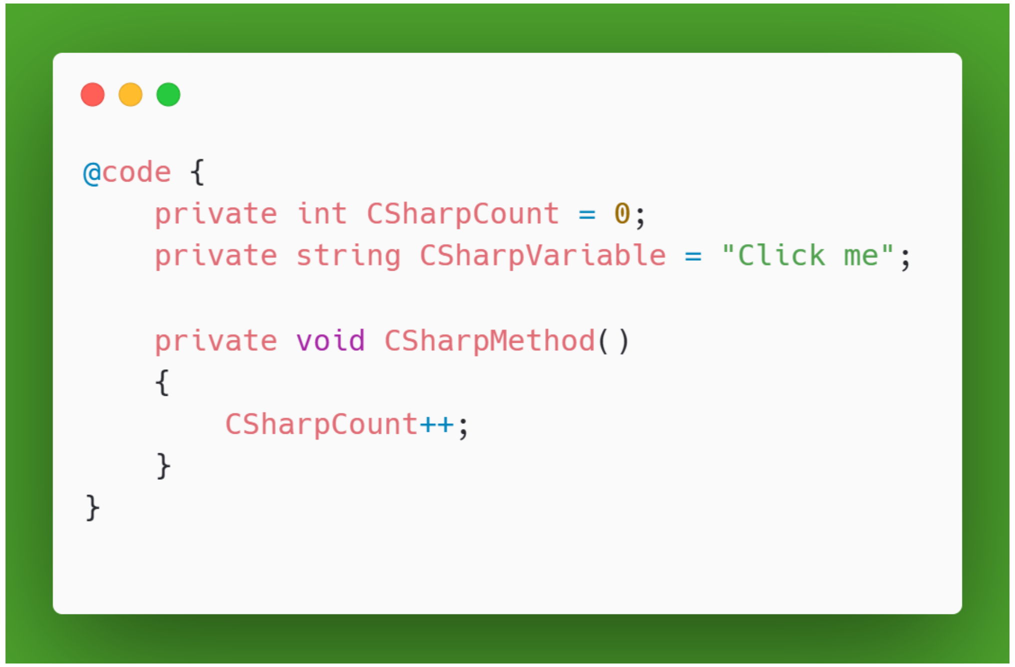
Razor Component Structure

Razor Markup



```
<div>
    <h1>Sample</h1>
    <p>This is a sample component.</p>
    <button @onclick="CSharpMethod">
        @CSharpVariable
      </button>
    <p>Current count: @CSharpCount</p>
</div>
```

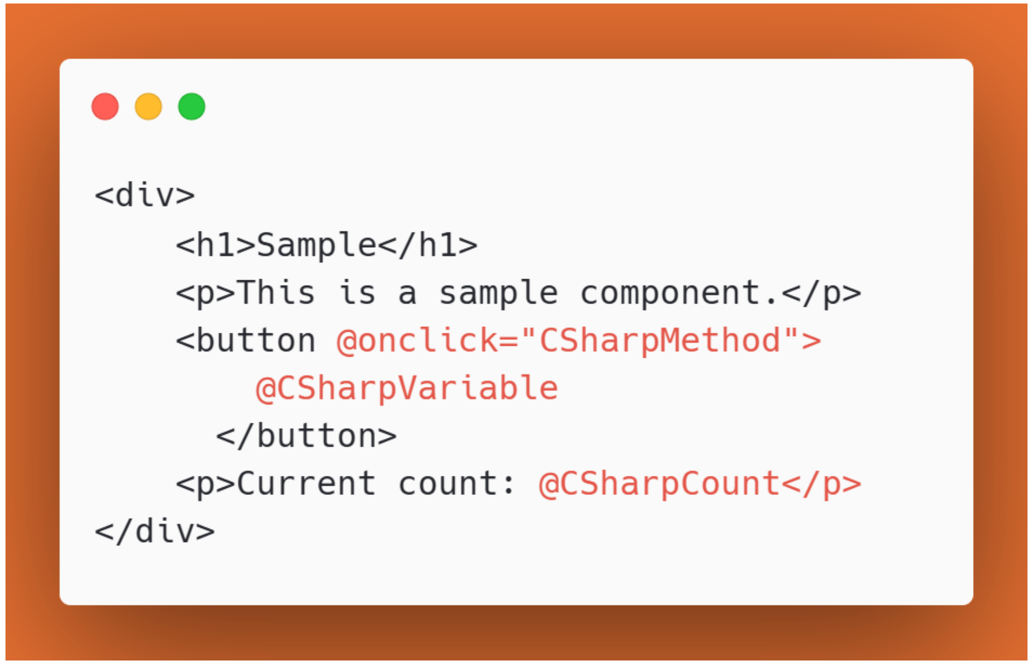
Code Block



```
@code {
    private int CSharpCount = 0;
    private string CSharpVariable = "Click me";

    private void CSharpMethod()
    {
        CSharpCount++;
    }
}
```

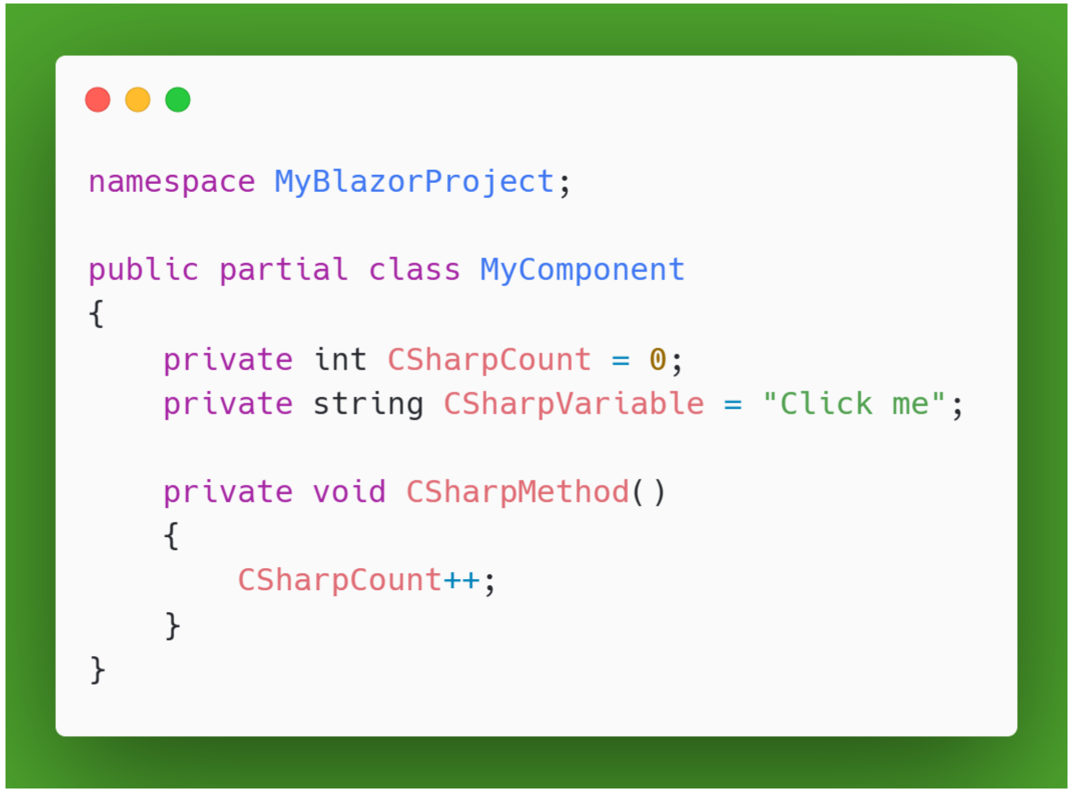
Razor Partial Class ("Code-Behind" Pattern)

MyComponent.razor



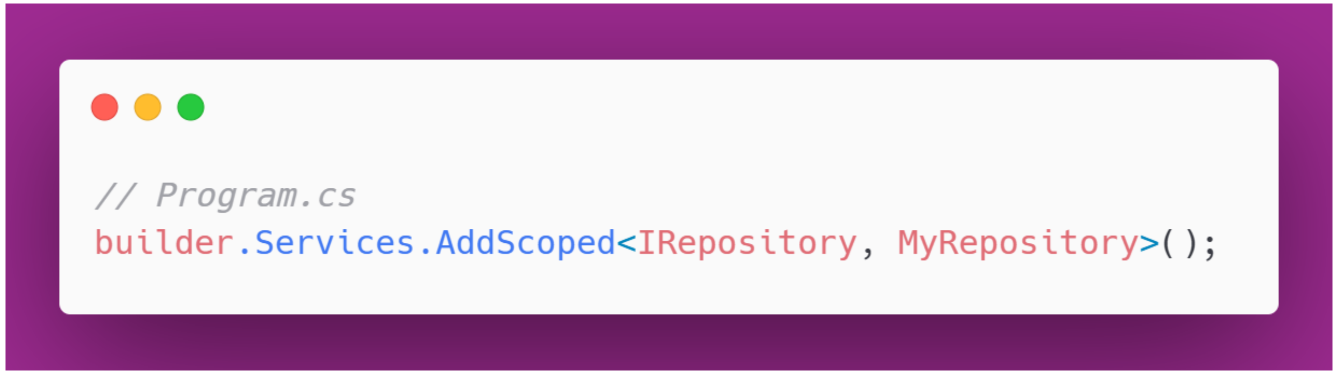
```
<div>
    <h1>Sample</h1>
    <p>This is a sample component.</p>
    <button @onclick="CSharpMethod">
        @CSharpVariable
      </button>
    <p>Current count: @CSharpCount</p>
</div>
```

MyComponent.razor.cs

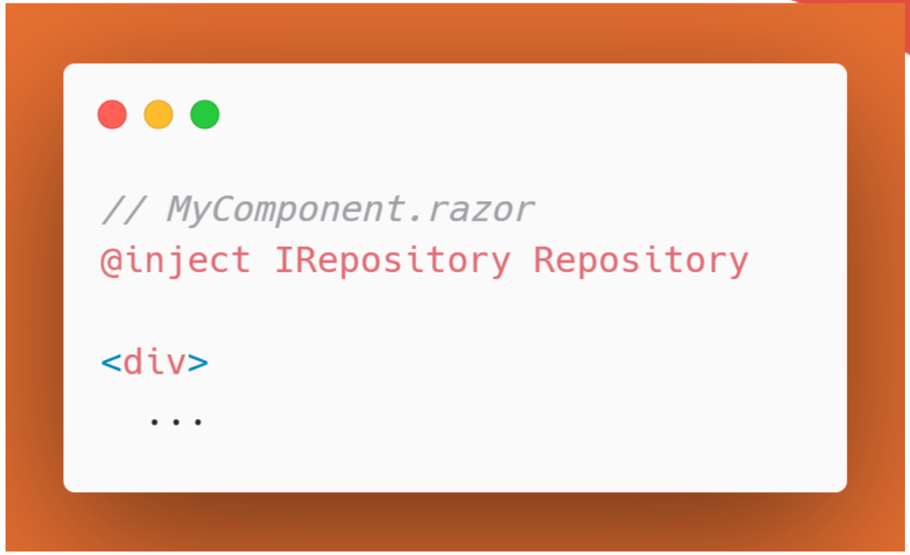


```
namespace MyBlazorProject;

public partial class MyComponent
{
    private int CSharpCount = 0;
    private string CSharpVariable = "Click me";

    private void CSharpMethod()
    {
        CSharpCount++;
    }
}
```

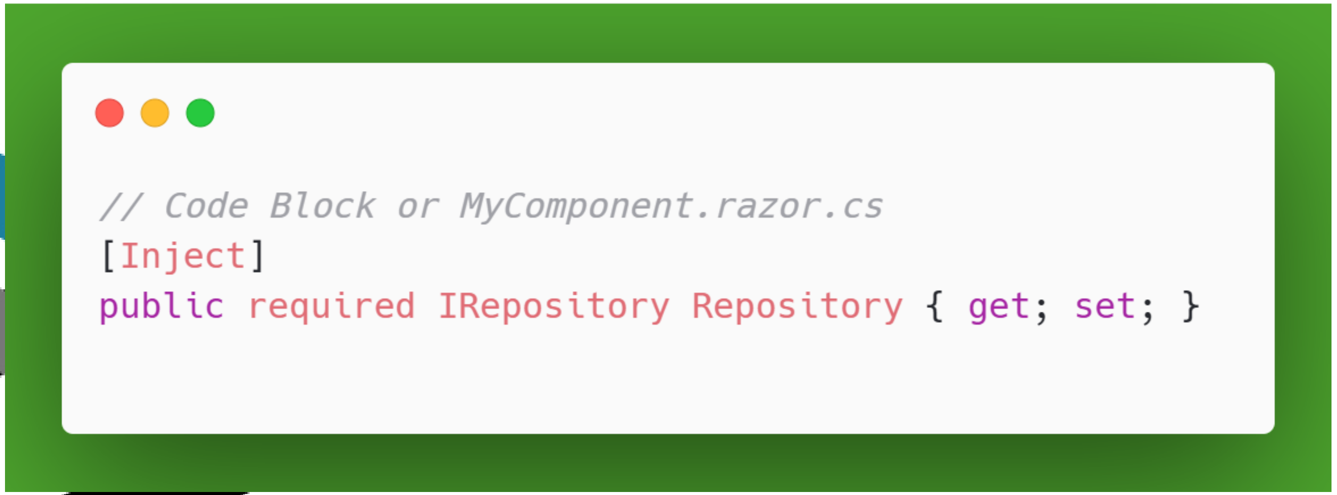
Dependency Injection



```
// Program.cs
builder.Services.AddScoped<IRepository, MyRepository>();
```



```
// MyComponent.razor
@inject IRepository Repository

<div>
  ...
```

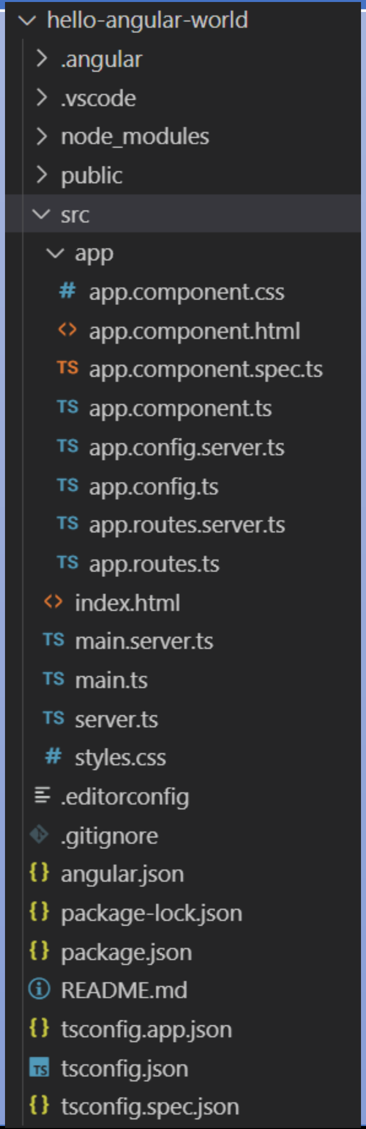
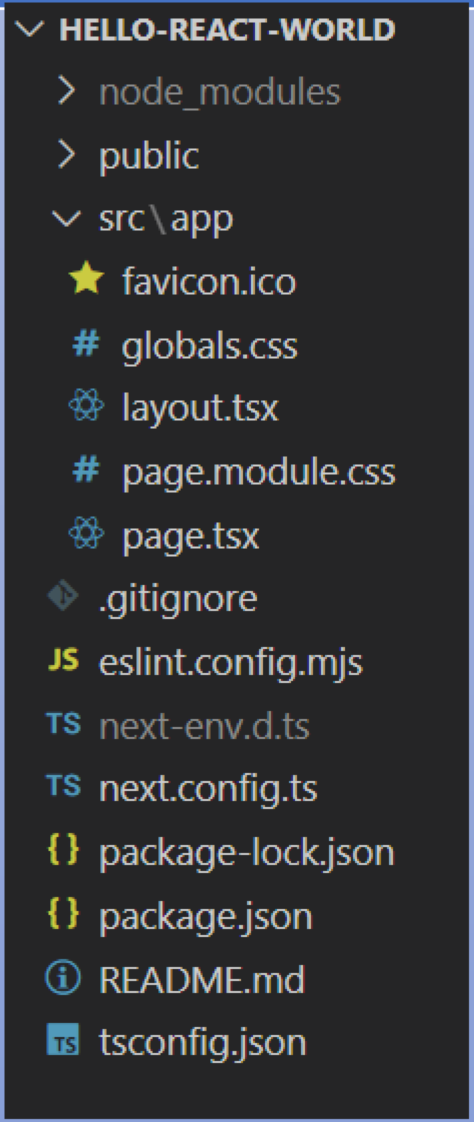
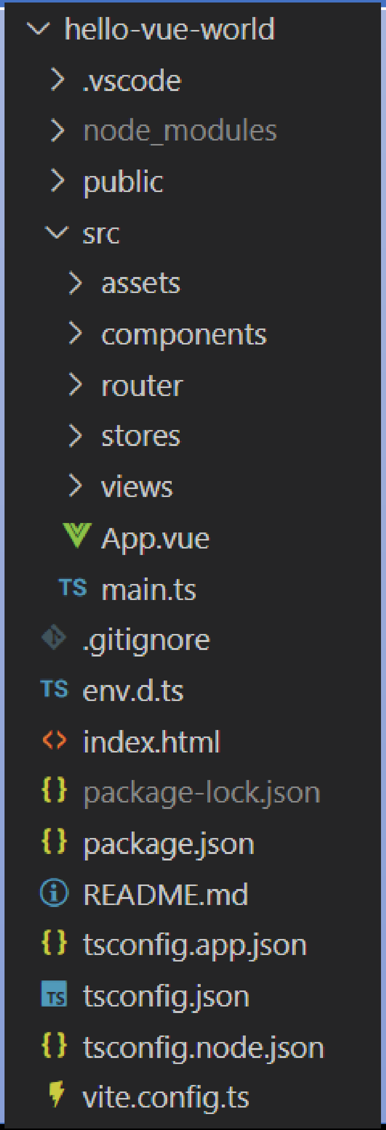
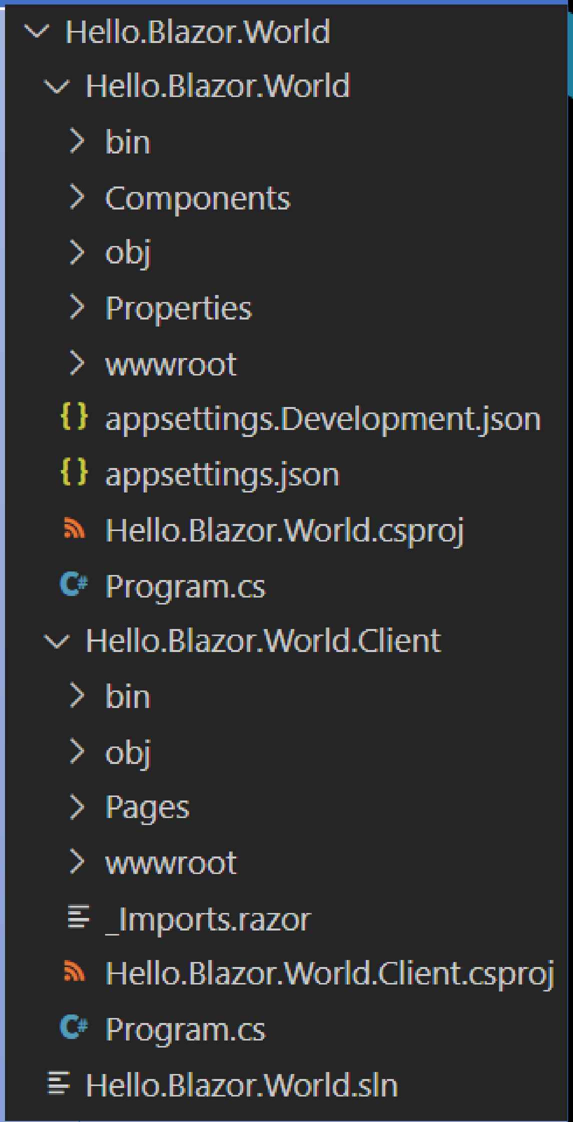


```
// Code Block or MyComponent.razor.cs
[Inject]
public required IRepository Repository { get; set; }
```

Comparing Blazor to JavaScript Frameworks

Feature / Aspect	Angular	React	Vue	Blazor
Created By	Google	Meta	Evan You (ex-Google)	Microsoft
First Release	2010	2013	2014	2018
Language	TypeScript	JavaScript / Typescript	JavaScript / TypeScript	C#/WebAssembly
Architectural Patterns	MVC, MVVM, Modules, Templates, Components, Dependency Injection Hexagonal, Onion, Vertical Slice	SPA, Flux, Redux, Components, HOC, SSR, Code Splitting, Reactive	MVVM, Components, Templates, Flat Structure, Modules, Micro Front Ends, Reactive	Reactive, Components, Dependency Injection, SSR, InteractiveServer

Comparing Blazor to JavaScript Frameworks

Feature / Aspect	Angular	React	Vue	Blazor
Project Structure				

Angular:

```
hello-angular-world
  .angular
  .vscode
  node_modules
  public
  src
    app
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.config.server.ts
      app.config.ts
      app.routes.server.ts
      app.routes.ts
    index.html
    main.server.ts
    main.ts
    server.ts
    styles.css
  .editorconfig
  .gitignore
  angular.json
  package-lock.json
  package.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.spec.json
```

React:

```
HELLO-REACT-WORLD
  node_modules
  public
  src\app
    favicon.ico
    globals.css
    layout.tsx
    page.module.css
    page.tsx
  .gitignore
  eslint.config.mjs
  next-env.d.ts
  next.config.ts
  package-lock.json
  package.json
  README.md
  tsconfig.json
```

Vue:

```
hello-vue-world
  .vscode
  node_modules
  public
  src
    assets
    components
    router
    stores
    views
    App.vue
    main.ts
  .gitignore
  env.d.ts
  index.html
  package-lock.json
  package.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.node.json
  vite.config.ts
```

Blazor:

```
Hello.Blazor.World
  Hello.Blazor.World
    bin
    Components
    obj
    Properties
    wwwroot
    appsettings.Development.json
    appsettings.json
    Hello.Blazor.World.csproj
    Program.cs
  Hello.Blazor.World.Client
    bin
    obj
    Pages
    wwwroot
    _Imports.razor
    Hello.Blazor.World.Client.csproj
    Program.cs
  Hello.Blazor.World.sln
```

Comparing Blazor to JavaScript Frameworks

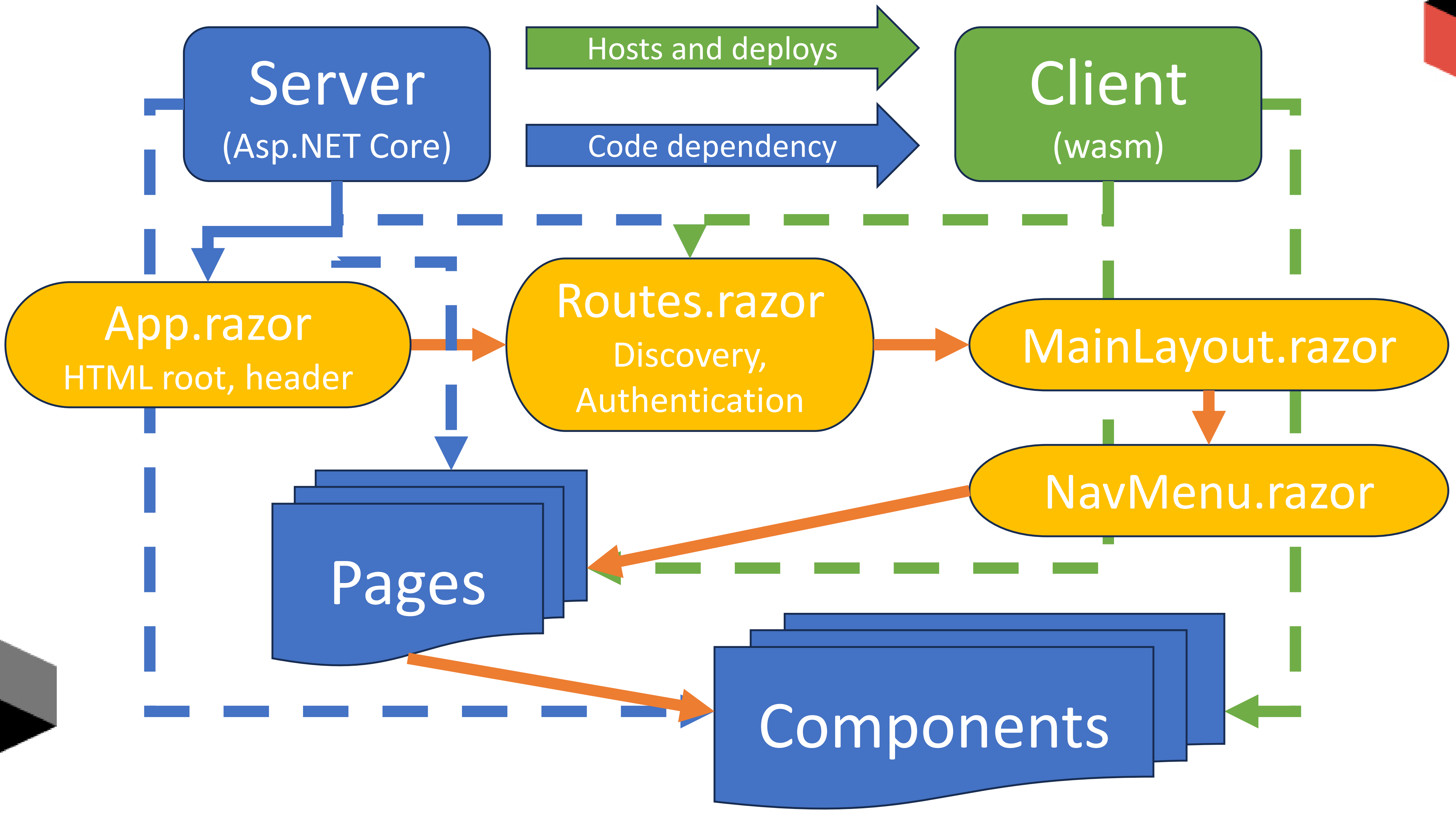
Feature / Aspect	Angular	React	Vue	Blazor
Markup Syntax	HTML Templates	JSX	SFC	Razor
File Structures	app.component.html app.component.css app.component.ts	page.tsx page.module.css	App.vue	Home.razor Home.razor.cs Home.razor.css Home.razor.js

Field/Property Injection	`<span>`
	{{ propVal }}
	`</span>`
		`<span>`
		{ propVal }
		`</span>`
			`<span>`
			{{ propVal }}
			`</span>`
				`<span>`
				@propVal
				`</span>`

Property Binding	[src]="variableUrl"	src={variableUrl}	:src="variableUrl"	src="@variableUrl"
Click Handler	(click)="jsFunction()"	onClick={jsFunction}	@click="jsFunction"	@onclick="CsharpMethod"

Two-way Form Binding	@angular/forms
	[(ngModel)]="model.item"
		value={item}
		onChange={setItem}
			v-model="item"	@bind="item" OR
				@bind:get="item" &
				@bind:set="method"

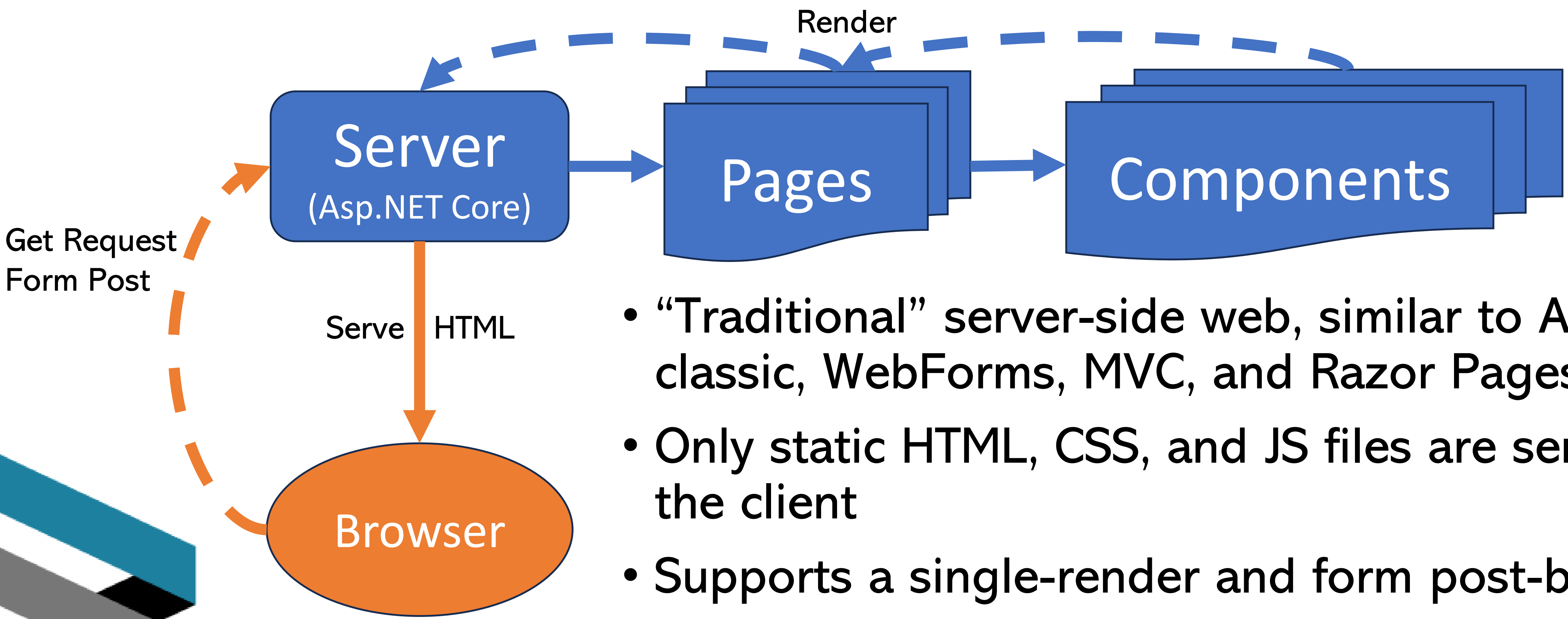
Project Structure



Blazor Component Render Modes

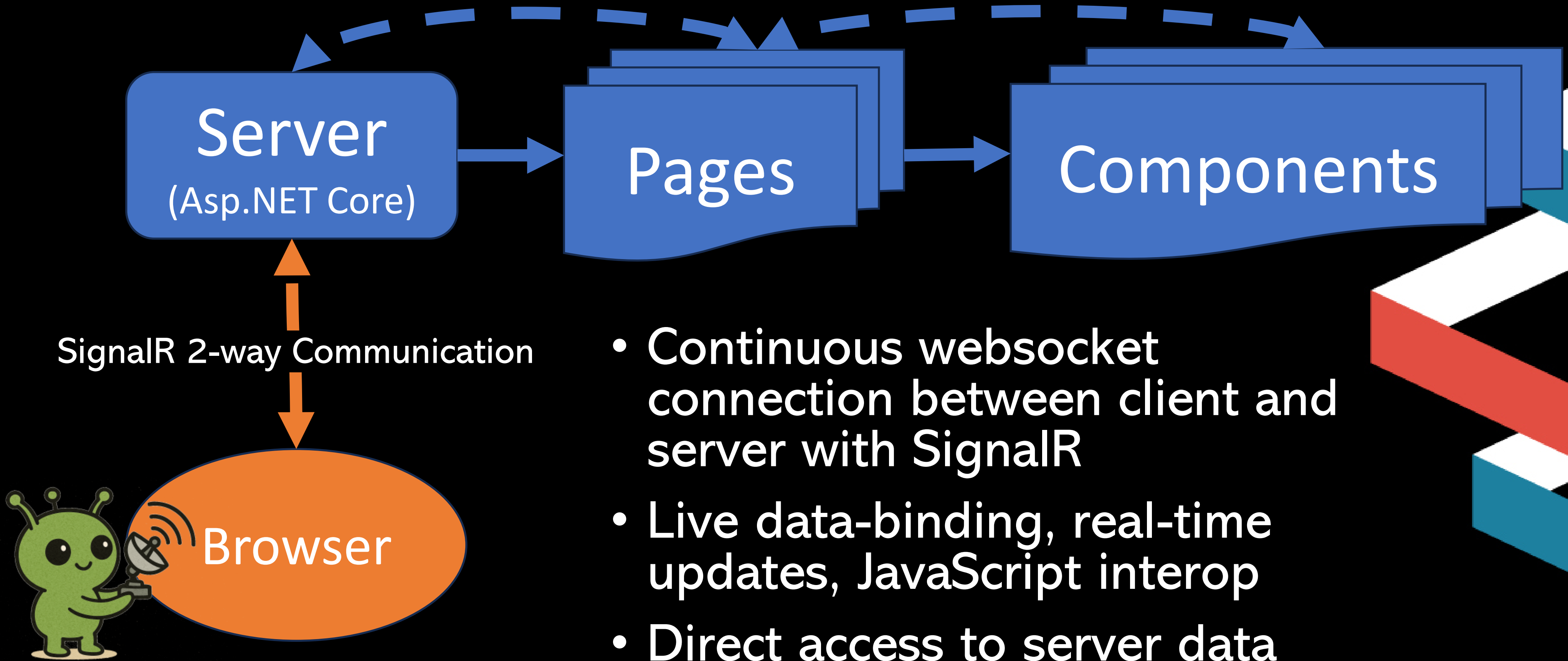
- Static Server Mode
- Interactive Server Mode
- Interactive WebAssembly Mode
- Interactive Auto Mode
- Blazor Hybrid (MAUI)

Blazor Render Modes: Static Server



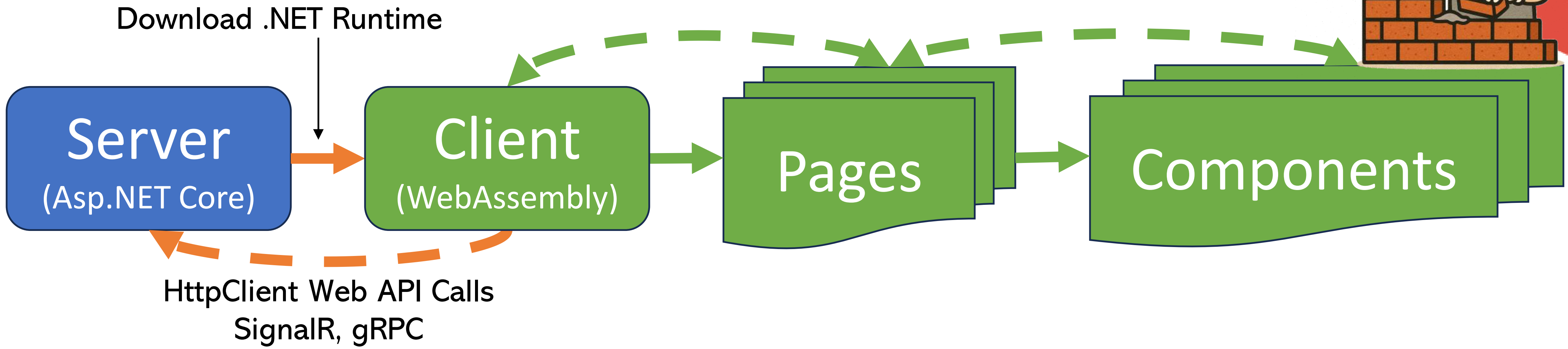
- "Traditional" server-side web, similar to ASP classic, WebForms, MVC, and Razor Pages
- Only static HTML, CSS, and JS files are sent to the client
- Supports a single-render and form post-backs
- No interactive updates via C# (can still use JS)
- Great for blog posts or other readonly content and simple forms

Blazor Render Modes: Interactive Server



- Continuous websocket connection between client and server with SignalR
- Live data-binding, real-time updates, JavaScript interop
- Direct access to server data store
- Fast on first load
- Leaving browser tabs open can cause disconnection issues

Blazor Render Modes: Interactive WebAssembly



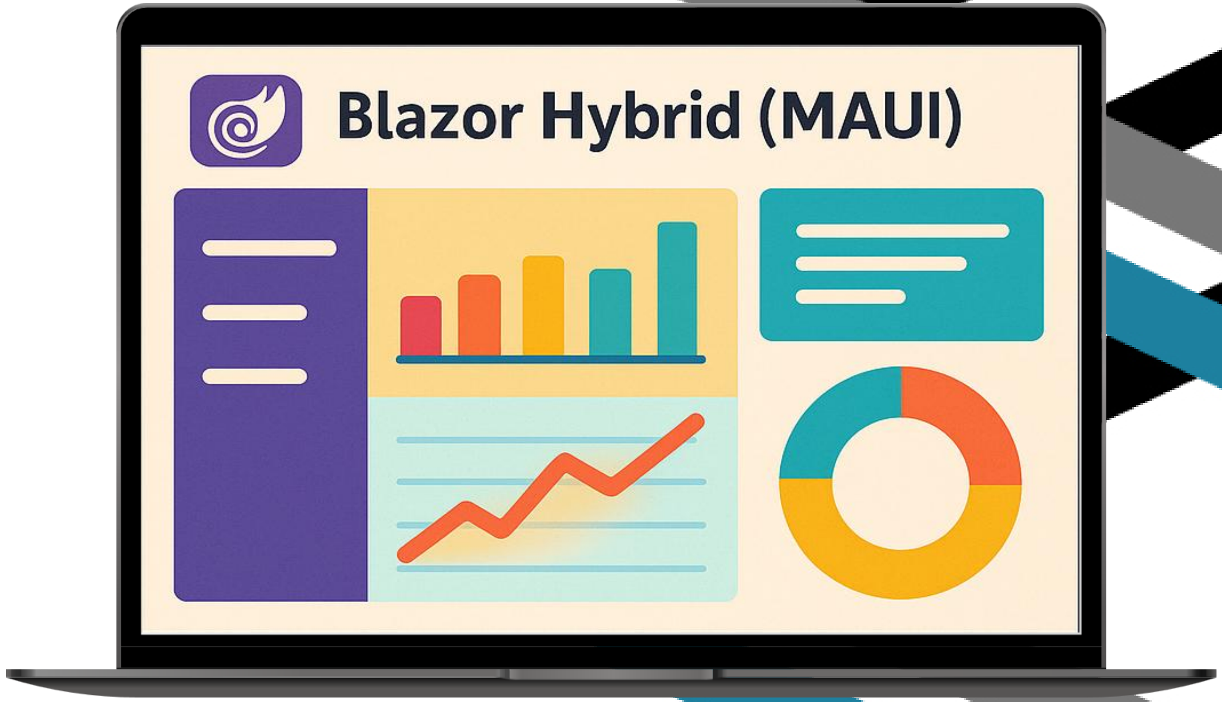
- Runs in the client browser
- Live data-binding, real-time updates, JavaScript interop
- HttpClient calls to communicate with server web API
- Single-threaded
- Large/slow first load
- Fast interactions after load
- Closest in approach to most JS SPA frameworks

Blazor Render Modes: Interactive Auto

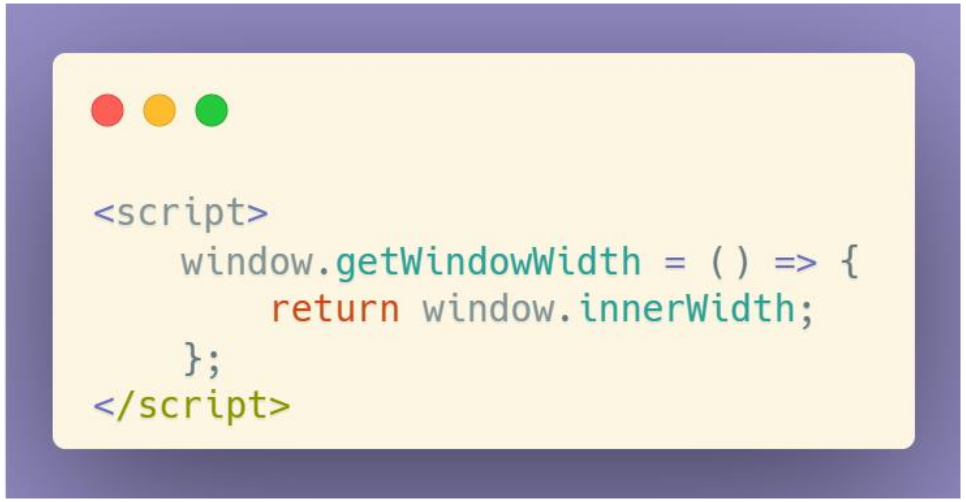
- On first load, runs from server, creating SignalR connection
- In the background, downloads .NET runtime and client code
- On next load, switches to running from WebAssembly
- “Best of both worlds”
  - Fast start on first load (server)
  - More responsive and robust interactions (client)
- Requires flexible data handling/abstraction to handle both client and server modes

Blazor Hybrid (MAUI)

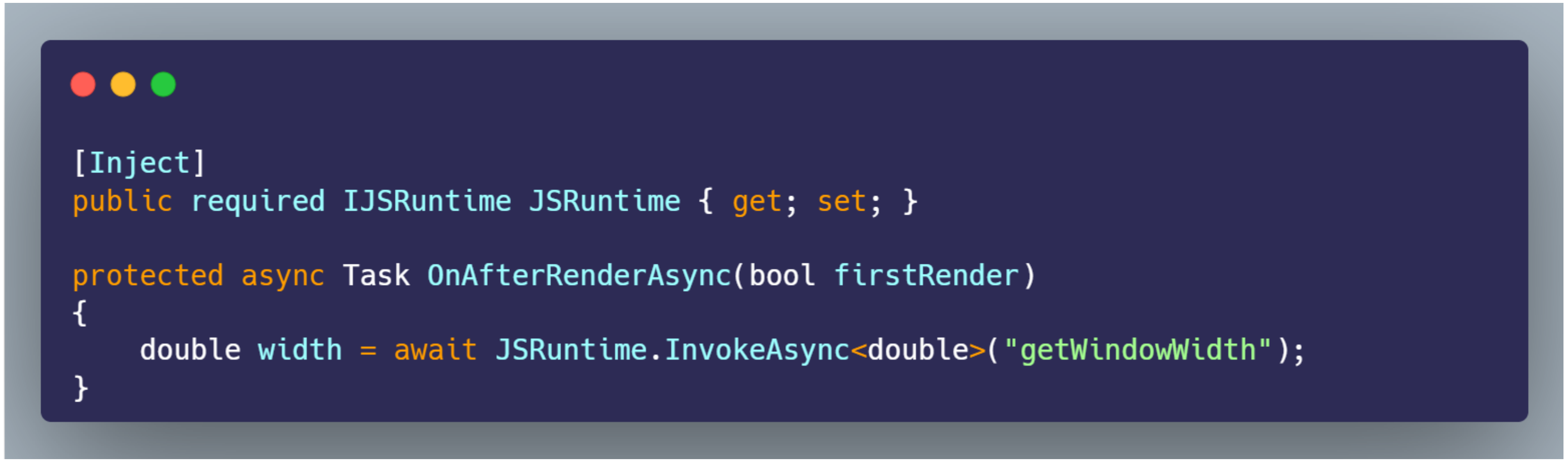
- Runs in a WebView in .NET MAUI (iOS, Android, Mac, Windows)
- Native .NET multi-threaded code execution (not WebAssembly)
- Access to device APIs (GPS, Bluetooth, photos, etc.)
- Can reuse components or entire UI applications between web, desktop, and mobile



Escaping Back into JavaScript…



```
<script>
    window.getWindowWidth = () => {
        return window.innerWidth;
    };
</script>
```



```
[Inject]
public required IJSRuntime JSRuntime { get; set; }

protected async Task OnAfterRenderAsync(bool firstRender)
{
    double width = await JSRuntime.InvokeAsync<double>("getWindowWidth");
}
```

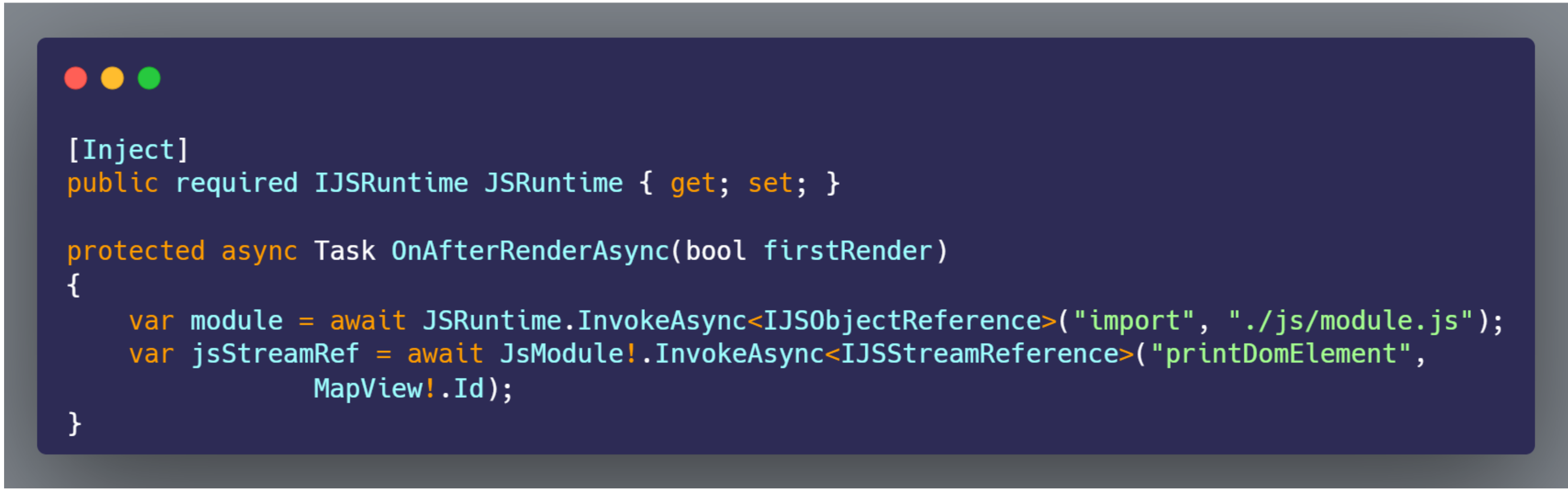
Escaping Back into JavaScript…

module.js

```
export async function printDomElement(elementId) {
    let canvas = await html2canvas(document.getElementById(elementId));
    return base64ToArrayBuffer(canvas.toDataURL("image/png").split(",")[1]);
}

function base64ToArrayBuffer(base64): Uint8Array {
    const binaryString = atob(base64);
    const bytes = new Uint8Array(binaryString.length);
    for (let i = 0; i < binaryString.length; i++) {
        bytes[i] = binaryString.charCodeAt(i);
    }
    return bytes;
}
```

Escaping Back into JavaScript…

Importing a JavaScript module



```
[Inject]
public required IJSRuntime JSRuntime { get; set; }

protected async Task OnAfterRenderAsync(bool firstRender)
{
    var module = await JSRuntime.InvokeAsync<IJSObjectReference>("import", "./js/module.js");
    var jsStreamRef = await JsModule!.InvokeAsync<IJSStreamReference>("printDomElement",
                MapView!.Id);
}
```

Escaping Back into JavaScript…

Calling .NET from JavaScript

```
window.initialize = (dotNetRef) => {
  window.addEventListener('resize', async () => {
      const width = window.innerWidth;
      const height = window.innerHeight;
      await dotNetRef?.invokeMethodAsync('OnViewSizeChanged', width, height);
  });
}
```

```
[JSInvokable]
public async Task OnViewSizeChanged(double width, double height)
{
    // update C# code
}
```

https://nation-finder.geoblazor.com

Find the country based on its outline

Thank You!

dymaptic

Notes & Links @
https://timpurdum.dev

GeoBlazor

dev up