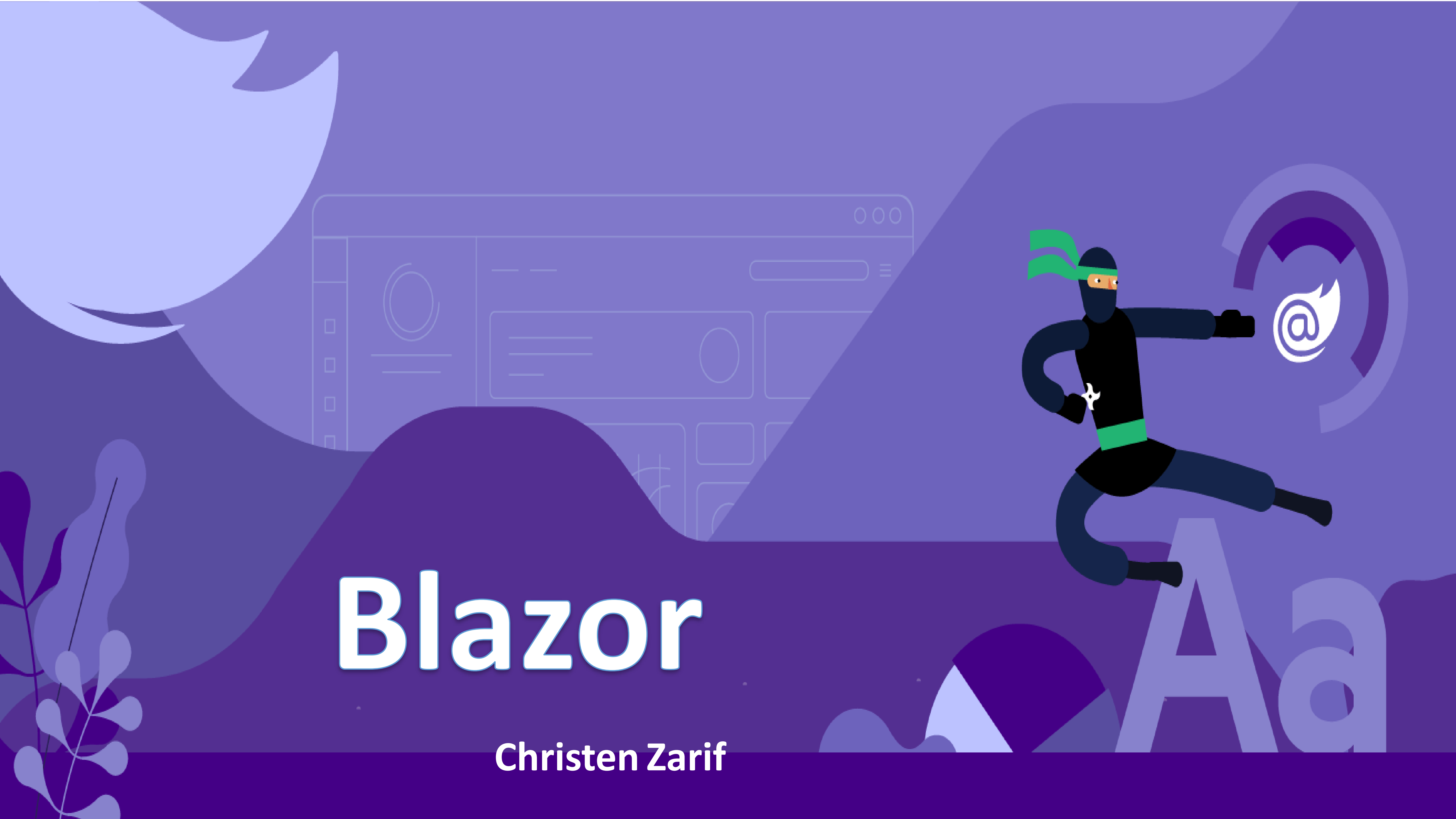


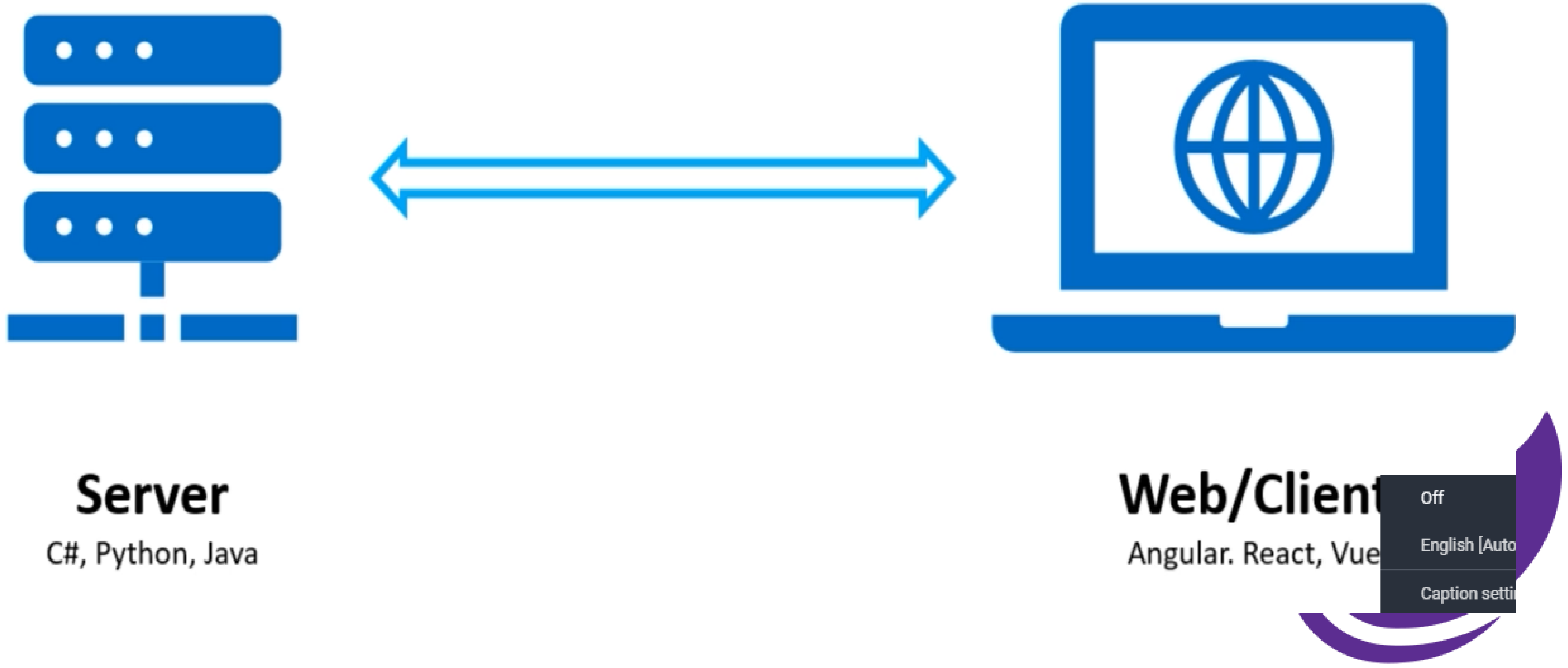
Blazor

Christen Zarif

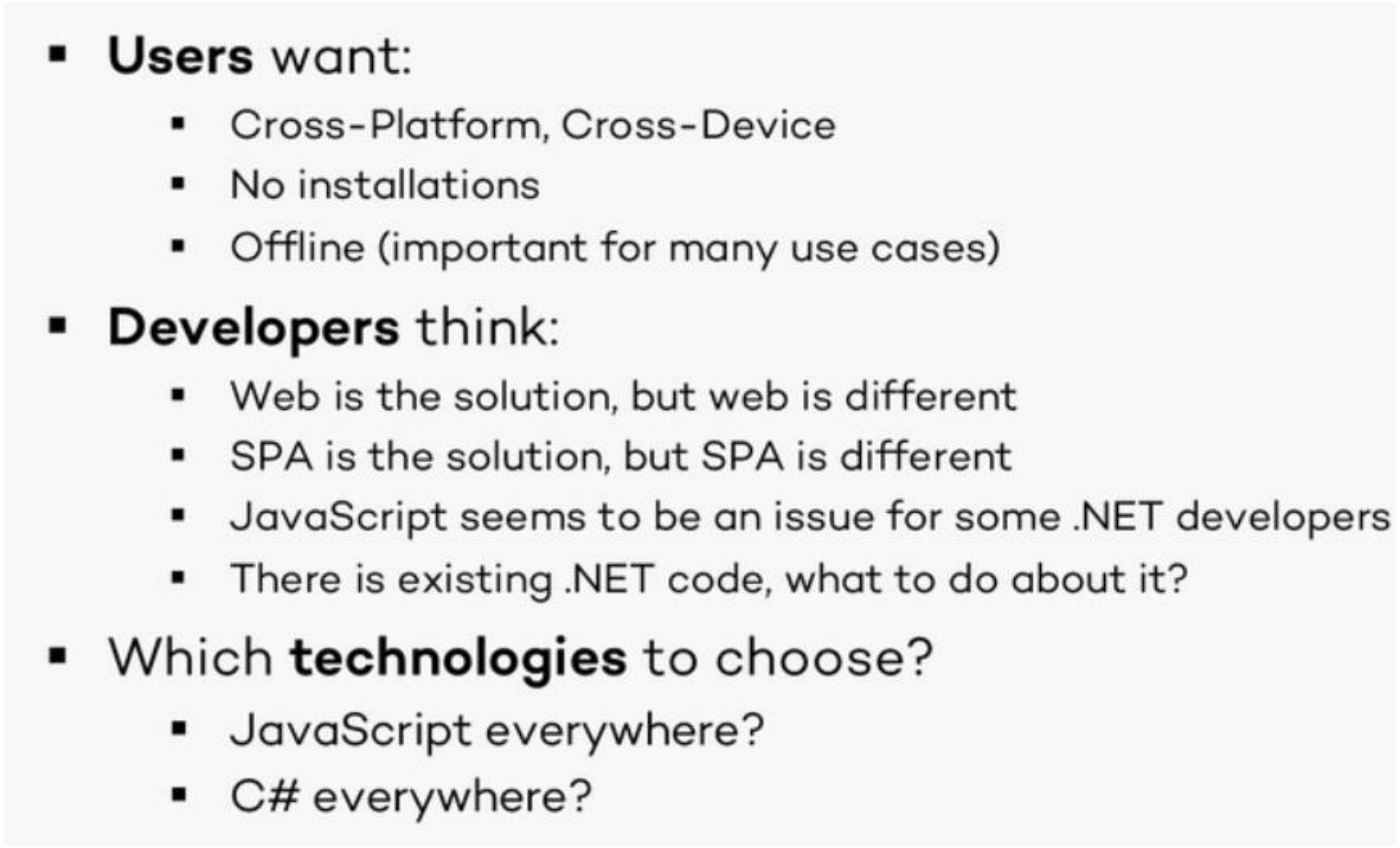
Course OutLine

- Part1
  - What is Blazor?
  - What is WebAssembly?
  - Why blazor?
  - Hosting models
    - Advantage & disadva. Of each model
  - Development environment
  - Creating a Blazor App with Visual Studio 2019
  - Project structure
  - Reviewing the generated code
  - Creating a Blazor App using the .net core cli

Traditional Web Applications



What is the Problem?



- **Users** want:
  - Cross-Platform, Cross-Device
  - No installations
  - Offline (important for many use cases)
- **Developers** think:
  - Web is the solution, but web is different
  - SPA is the solution, but SPA is different
  - JavaScript seems to be an issue for some .NET developers
  - There is existing .NET code, what to do about it?
- Which **technologies** to choose?
  - JavaScript everywhere?
  - C# everywhere?

Blazor is a framework for building interactive client-side web UI with $\boxed{\text{.NET}}$

- Microsoft

What is Blazor

- It gives all the benefits of a rich, **modern single-page application (SPA)** platform while using .NET end-to-end.

- It is based on existing web technologies like **HTML and CSS**, but you use **C# and Razor syntax** instead of JavaScript to build composable web UI.

- It is a framework for client-side applications written in .NET, running under WebAssembly

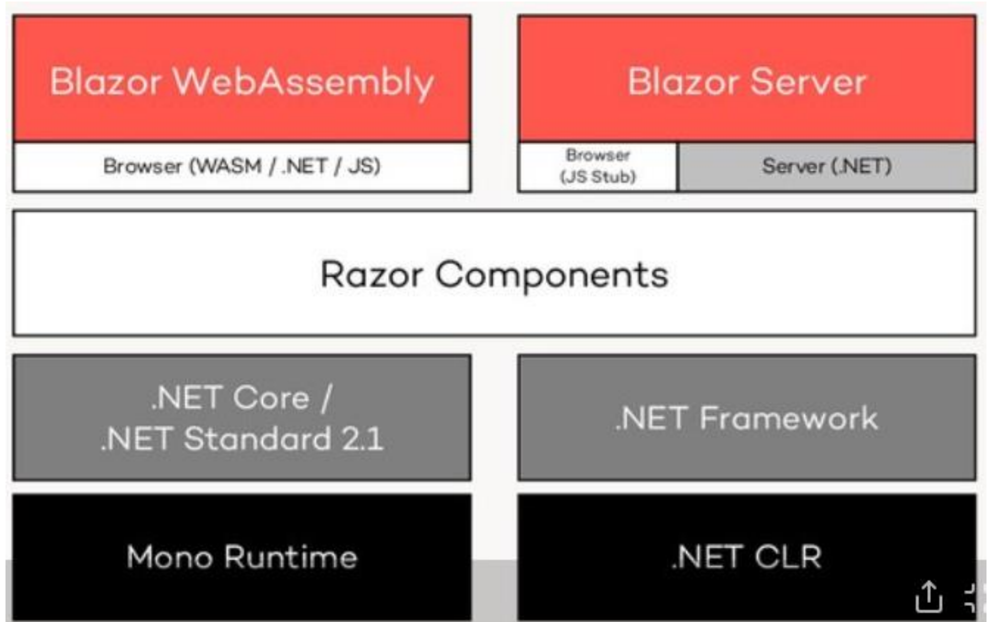
Blazor is a

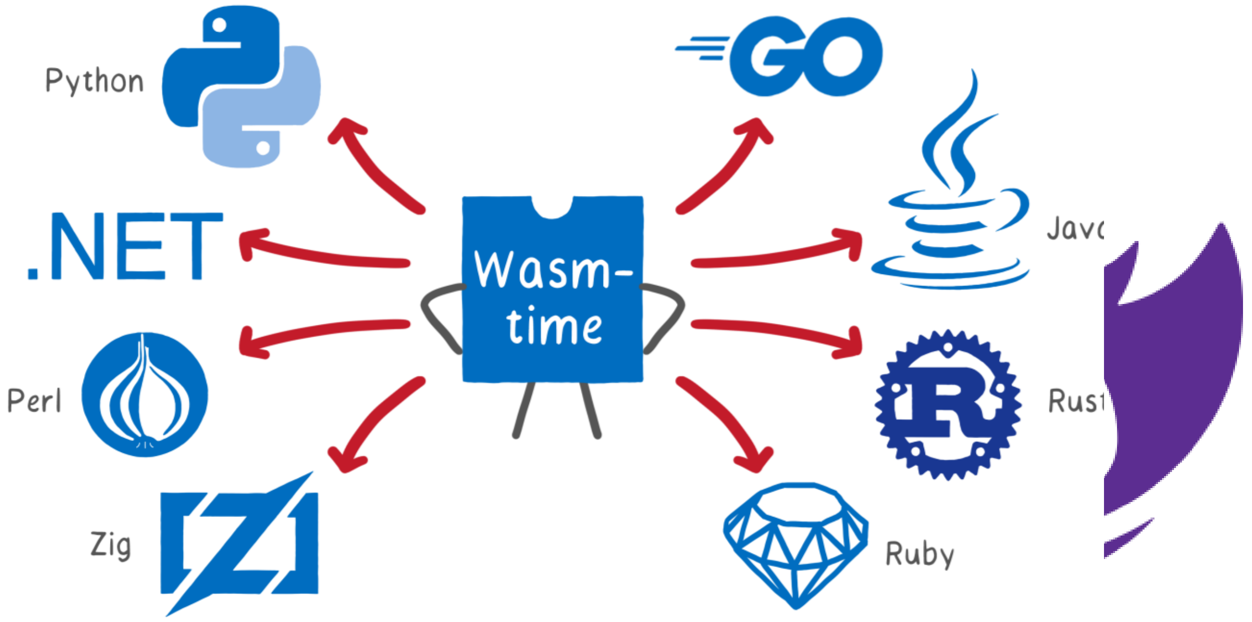
Single Page Application

development framework.

How can a browser execute C# code?

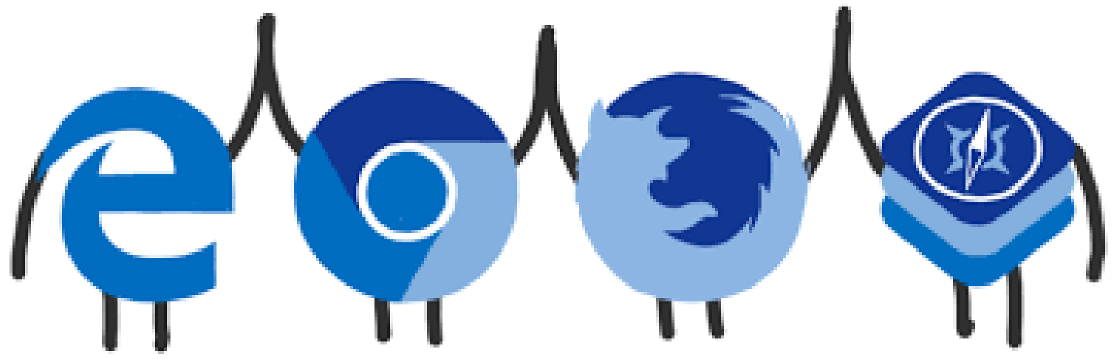
- Remember one thing, the browsers can only understand and execute JavaScript code. Then How can we execute our c# code in the client browser? The answer is by using something called as **WebAssembly**.





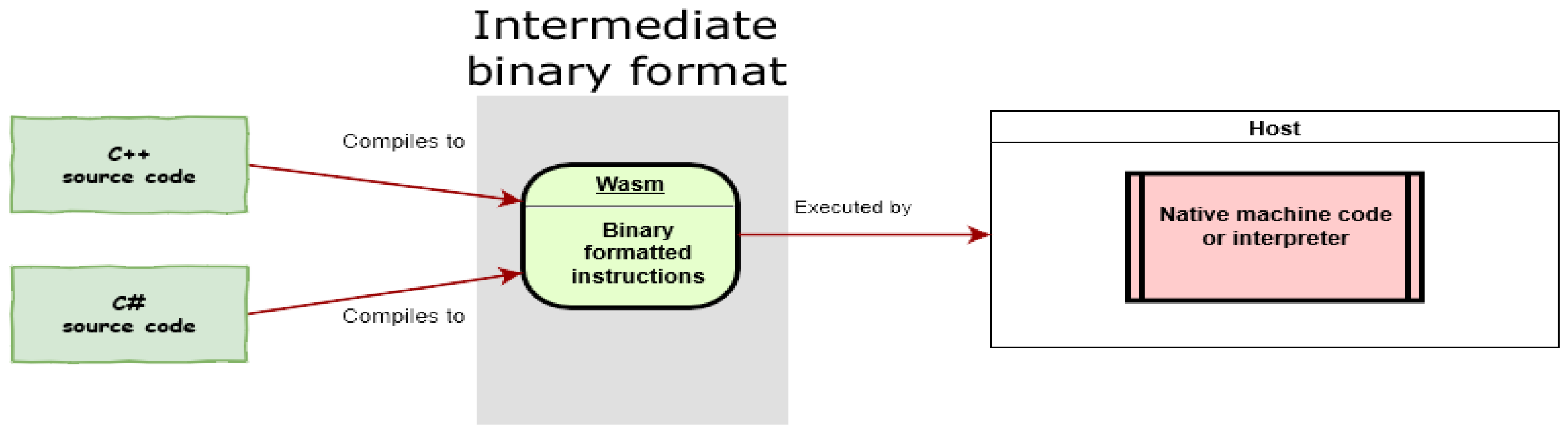
WebAssembly

(WASM)

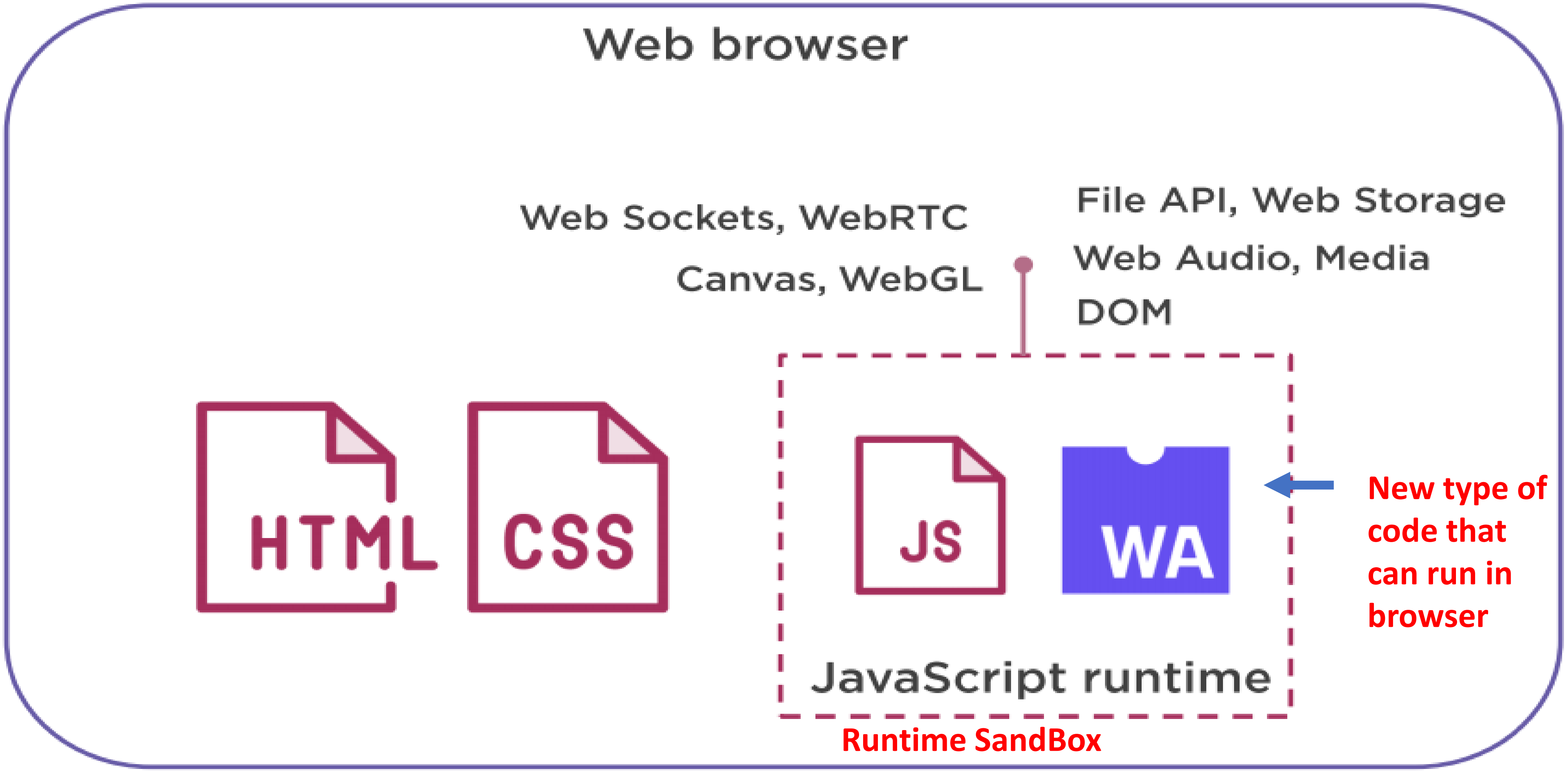


WebAssembly

- WebAssembly is a binary format for the code in the browser
- it runs much faster than traditional JavaScript.
- The main advantage of WebAssembly is that it handles memory-rich jobs and multi-threading very well as compared to javascript.


Intermediate binary format
C++ source code
C# source code
Compiles to
Compiles to
Wasm
Binary formatted instructions
Executed by
Host
Native machine code or interpreter

WebAssembly



WebAssembly

Javascript High Level interpreter Programming Language

WebAssembly near to native code Runtime execute it direct without need to interpreted or parse

```
function ShowDate() {
   document.getElementById('demo’)
             .innerHTML = Date();
}
```

Main.js

```
0x00000000 0061736D0100000001 .asm...........
0x00000010 7F0302010007070103 .........add....
0x00000020 010700200020016A0B ... . .j....name
0x00000030 0106010003616464 02 .....add.......l
0x00000040 68730103726873      hs..rhs
```

Main.wasm

Why WebAssembly?

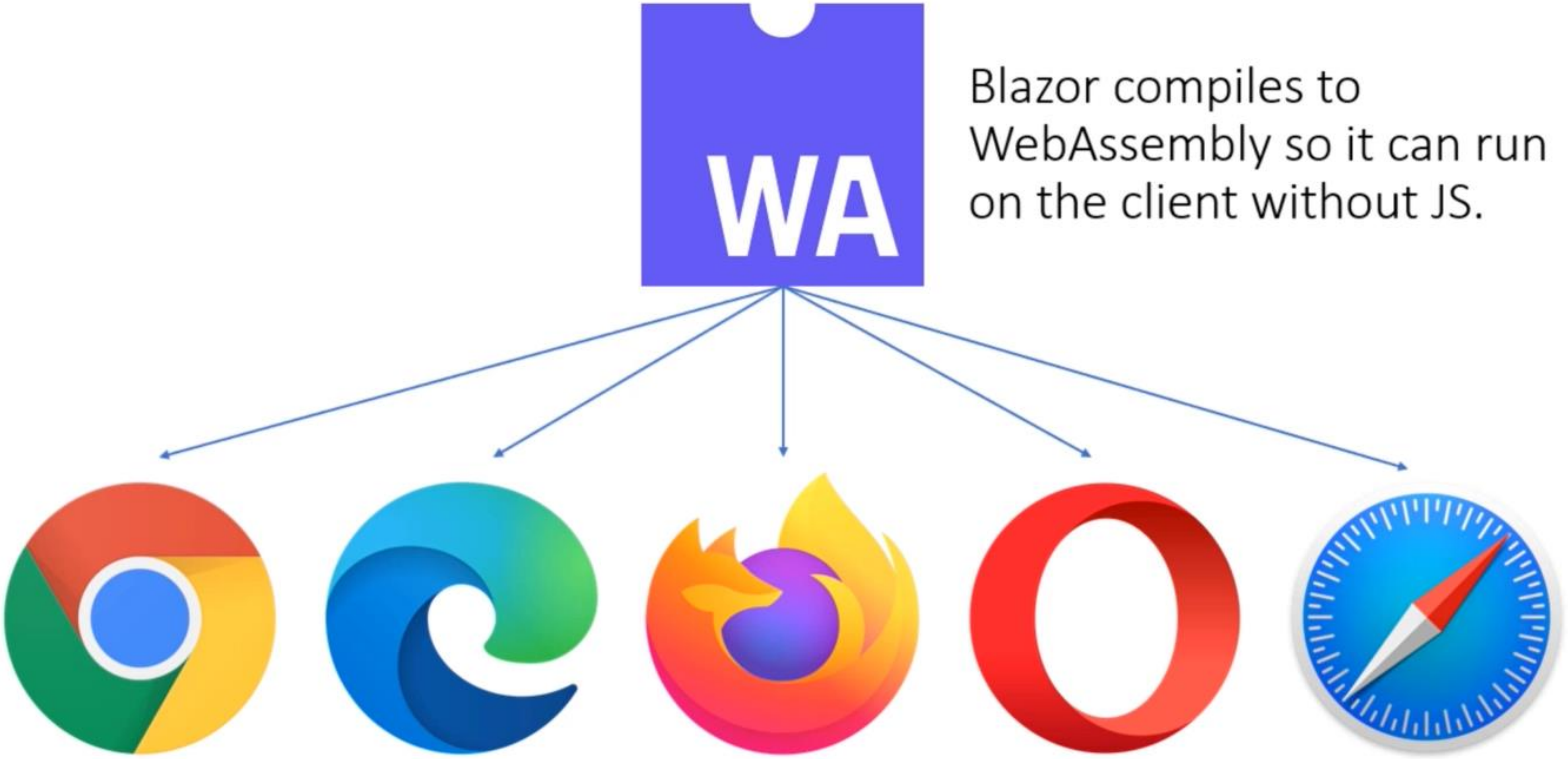
Run code at near-native speed

Other languages can be compiled to WebAssembly

Natively supported by browsers – no plugin needed

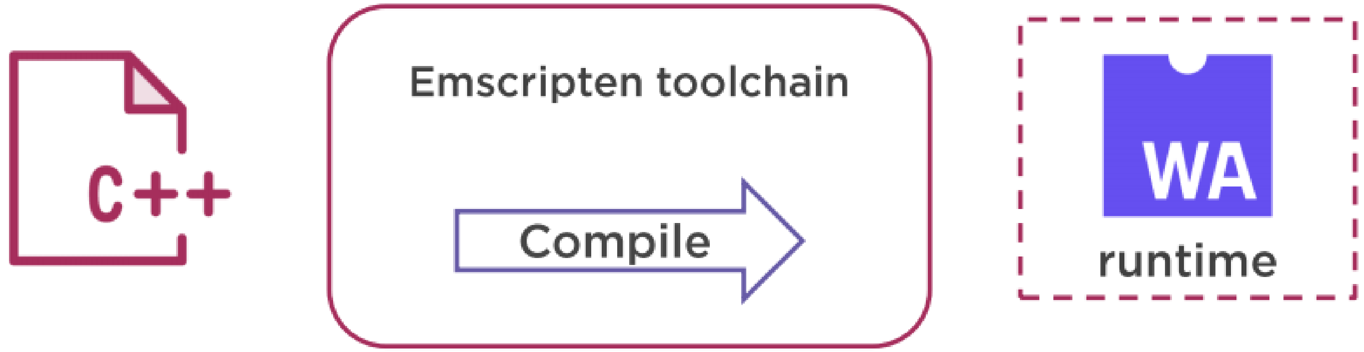
Secure by design – it runs in the JavaScript sandbox

JavaScript code can run WebAssembly modules



Blazor compiles to WebAssembly so it can run on the client without JS.

Compile Code into WebAssembly



Convert Direct From c++ to webAssembly

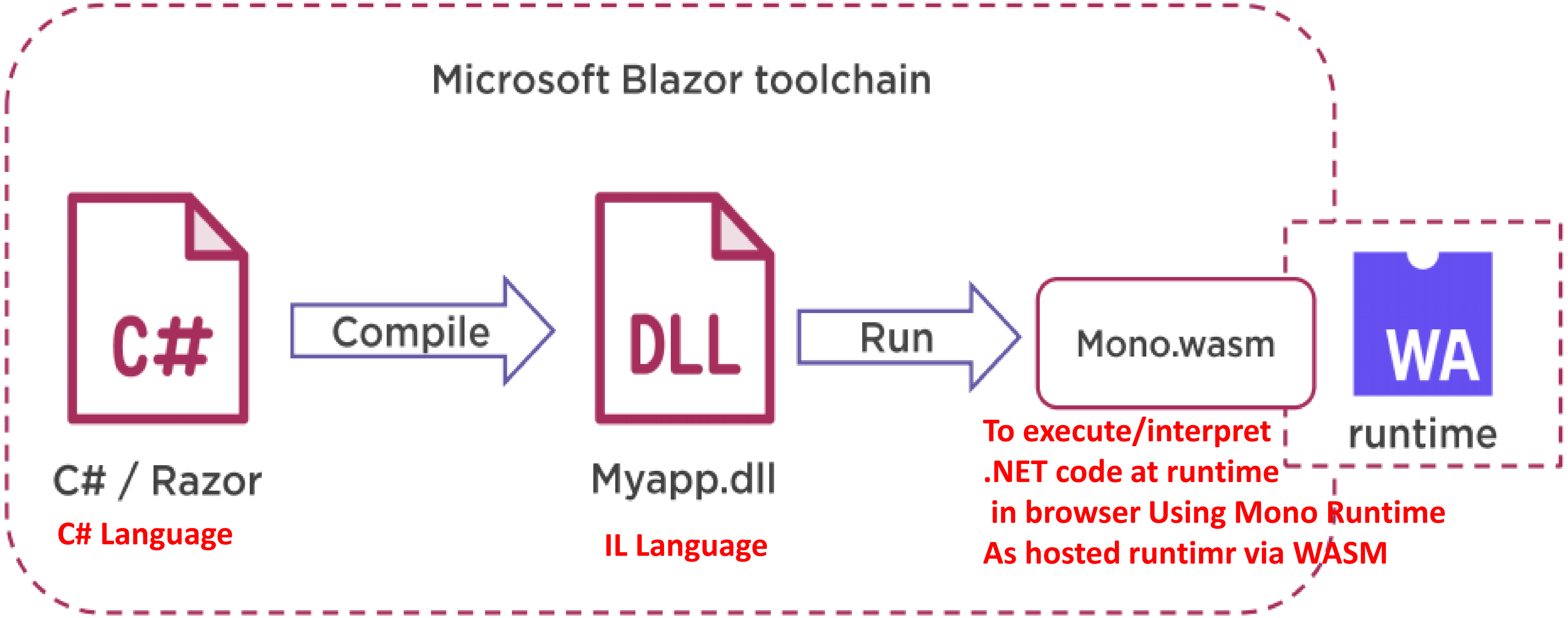
```
#include <iostream>

int main()
{
    std::cout << "Hello, World!";
    return 0;
}
```

MyCApp.cpp

```
0x00000000 0061736D0100000001 .asm..........
0x00000010 7F0302010007070103 .........add....
0x00000020 010700200020016A0B ... . .j....name
0x00000030 010601000361646402 .....add.......l
0x00000040 68730103726873     hs..rhs
```

MyCApp.wasm

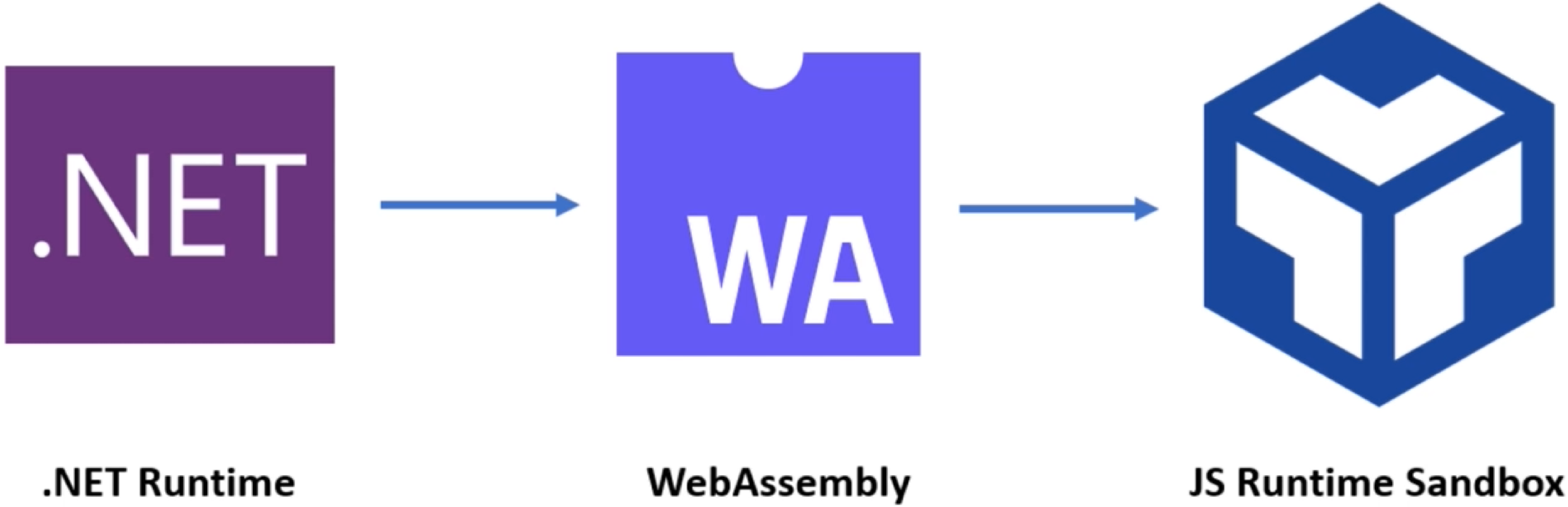
Compile Code into WebAssembly



C# Language

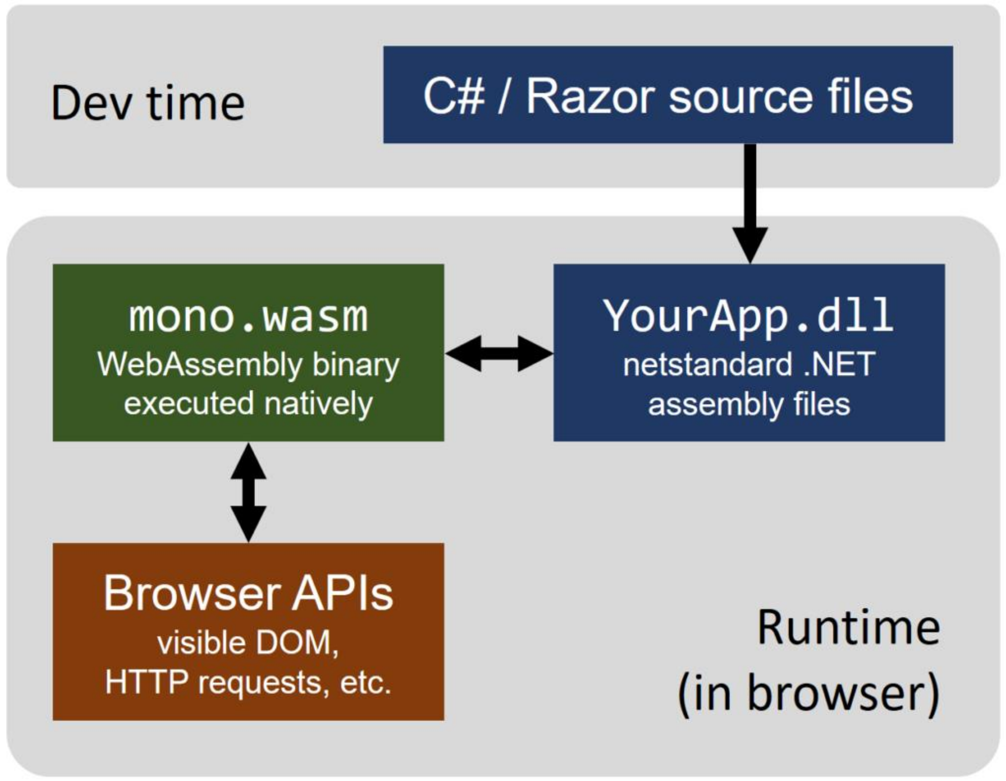
IL Language

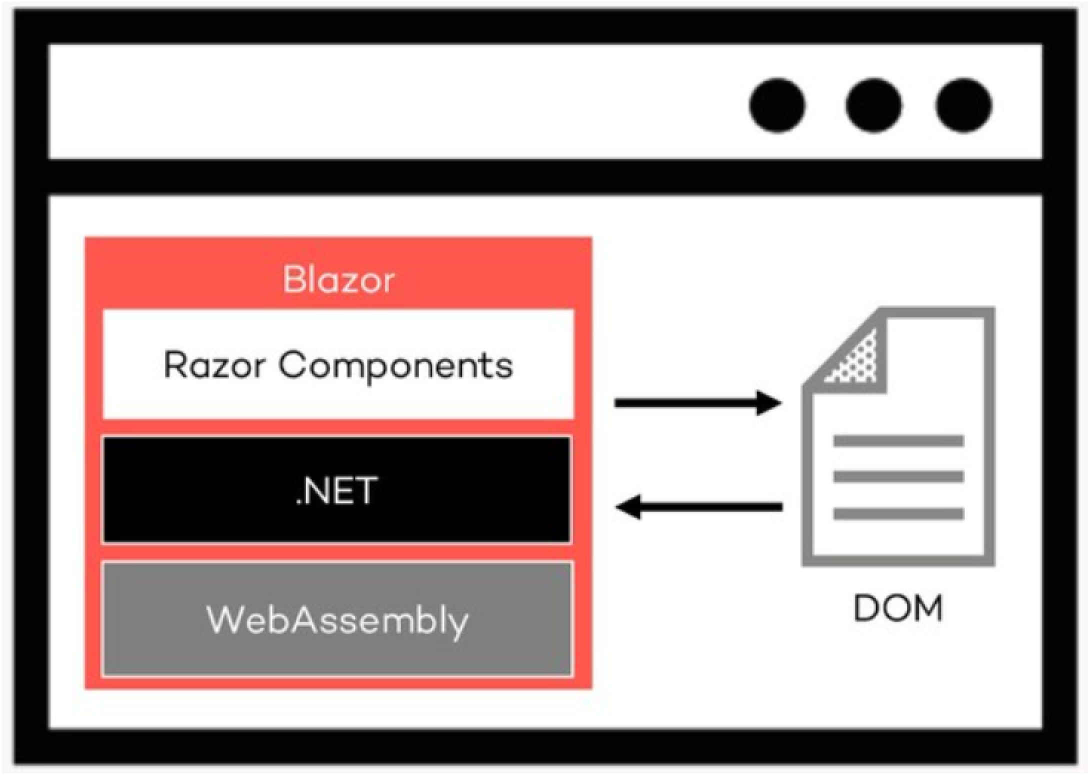
To execute/interpret .NET code at runtime in browser Using Mono Runtime As hosted runtimr via WASM


.NET
WA
.NET Runtime
WebAssembly
JS Runtime Sandbox

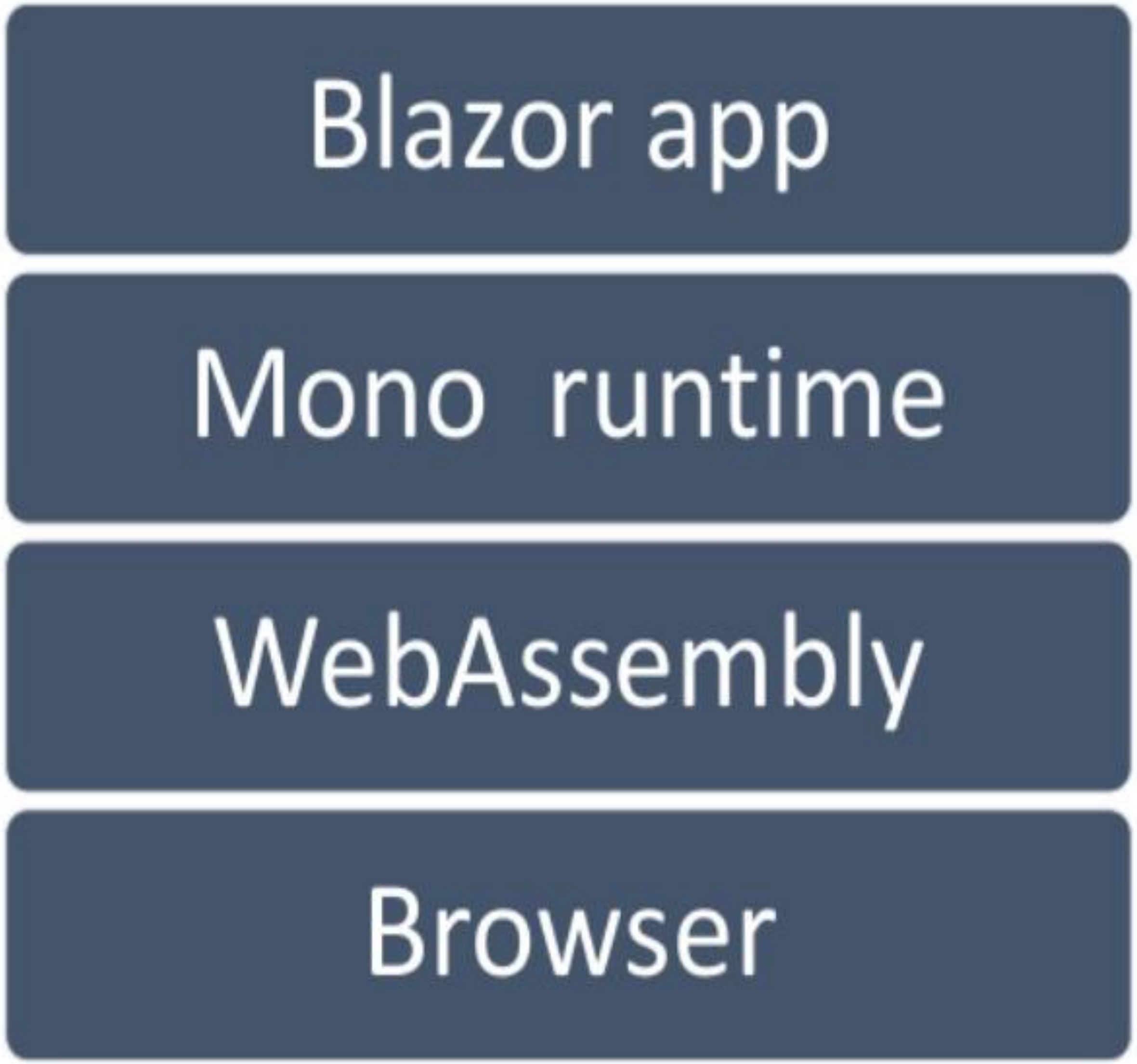
Blazor WebAssembly in browser

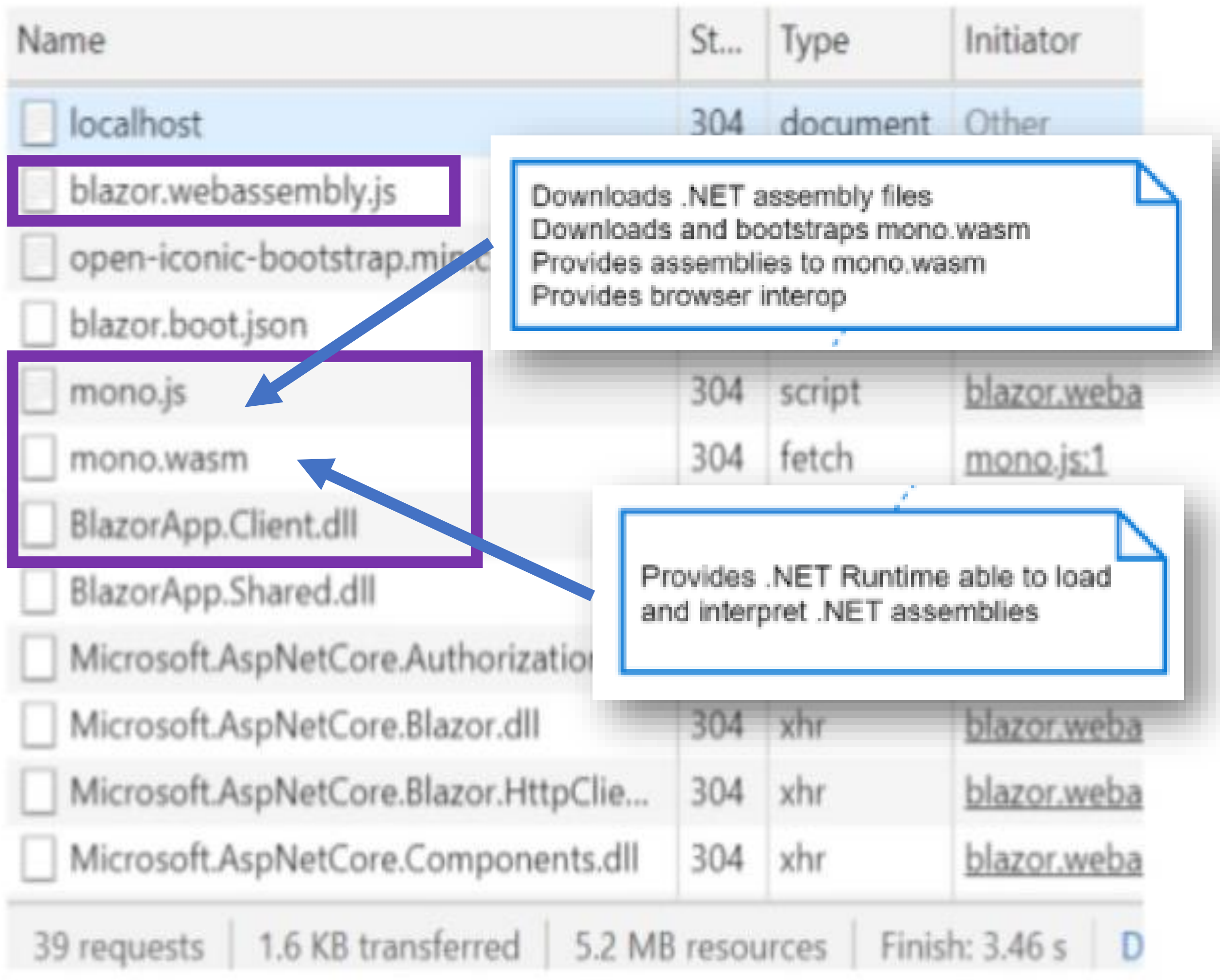
- Currently Blazor uses a approach of using Mono as a hosted runtime via WASM to execute/interpret .NET code at runtime.


Dev time
C# / Razor source files
mono.wasm
WebAssembly binary executed natively
YourApp.dll
netstandard .NET assembly files
Browser APIs
visible DOM, HTTP requests, etc.
Runtime (in browser)



Blazor WebAssembly





Name	St...	Type	Initiator
localhost	304	document	Other
blazor.webassembly.js			
open-iconic-bootstrap.mi			
blazor.boot.json			
mono.js	304	script	blazor.weba
mono.wasm	304	fetch	mono.js:1
BlazorApp.Client.dll			
BlazorApp.Shared.dll			
Microsoft.AspNetCore.Authorizatio			
Microsoft.AspNetCore.Blazor.dll	304	xhr	blazor.weba
Microsoft.AspNetCore.Blazor.HttpClie...	304	xhr	blazor.weba
Microsoft.AspNetCore.Components.dll	304	xhr	blazor.weba

39 requests | 1.6 KB transferred | 5.2 MB resources | Finish: 3.46 s | D

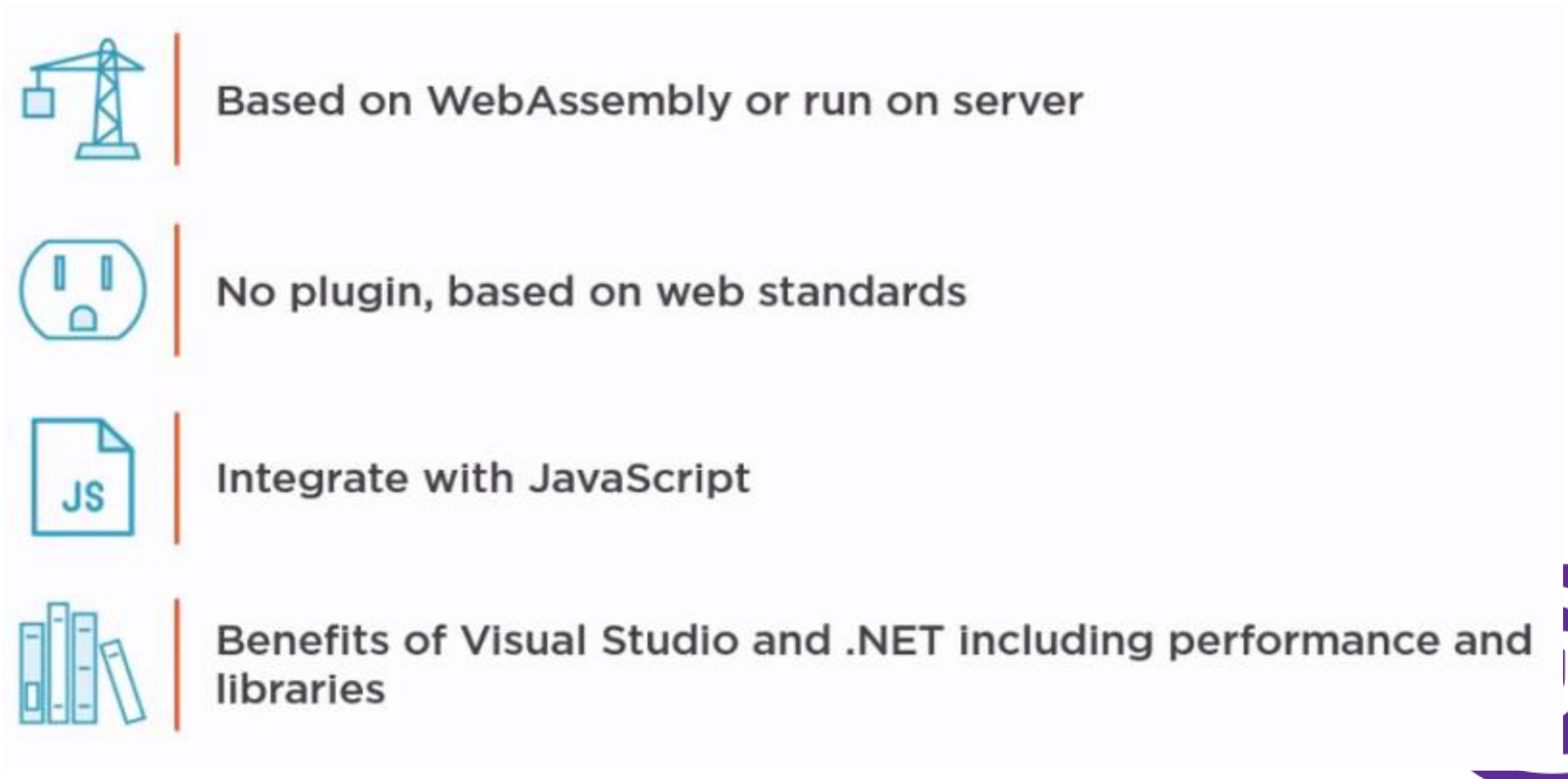
Downloads .NET assembly files
Downloads and bootstraps mono.wasm
Provides assemblies to mono.wasm
Provides browser interop

Provides .NET Runtime able to load and interpret .NET assemblies

Why Use Blazor?

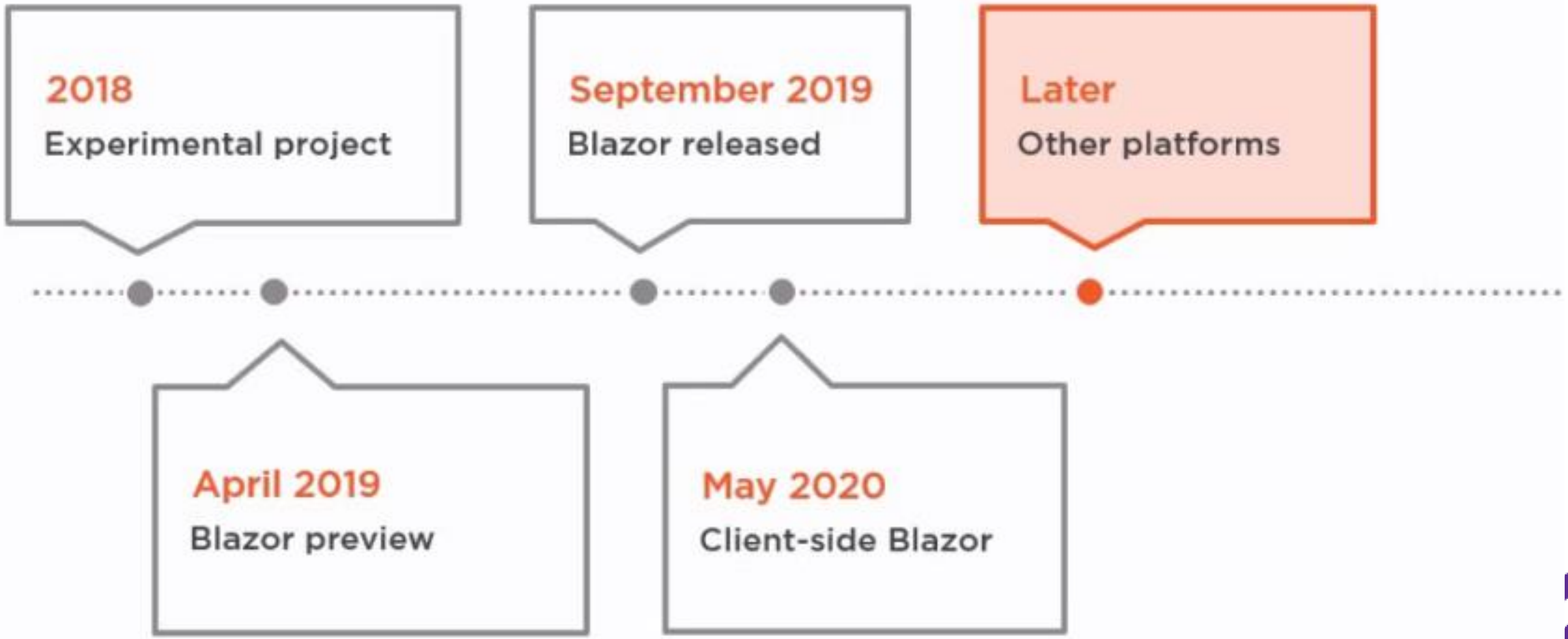
- ✓ Supported by all major browsers, including mobile devices
- Write code in C# instead of JavaScript.
- Leverage the existing .NET ecosystem of .NET libraries.
- Near-native performance
- Rich tooling and debugging (Visual Studio, Visual Code)

Conclusion



- Based on WebAssembly or run on server
- No plugin, based on web standards
- Integrate with JavaScript
- Benefits of Visual Studio and .NET including performance and libraries

Blazor Roadmap



Hosting Models

Blazor Hosting Models

WebAssembly

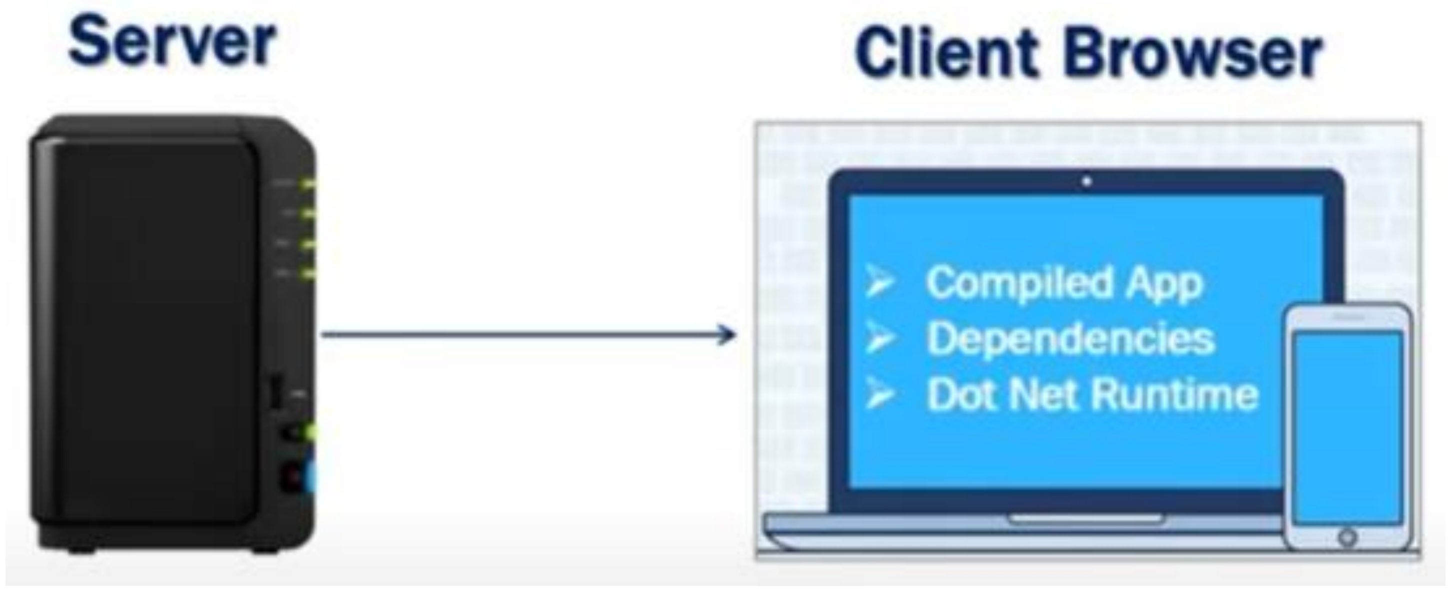
$\xrightarrow[\text{Side}]{\text{Client}}$

- Code runs in the **browser**
- Dependencies are downloaded

Server

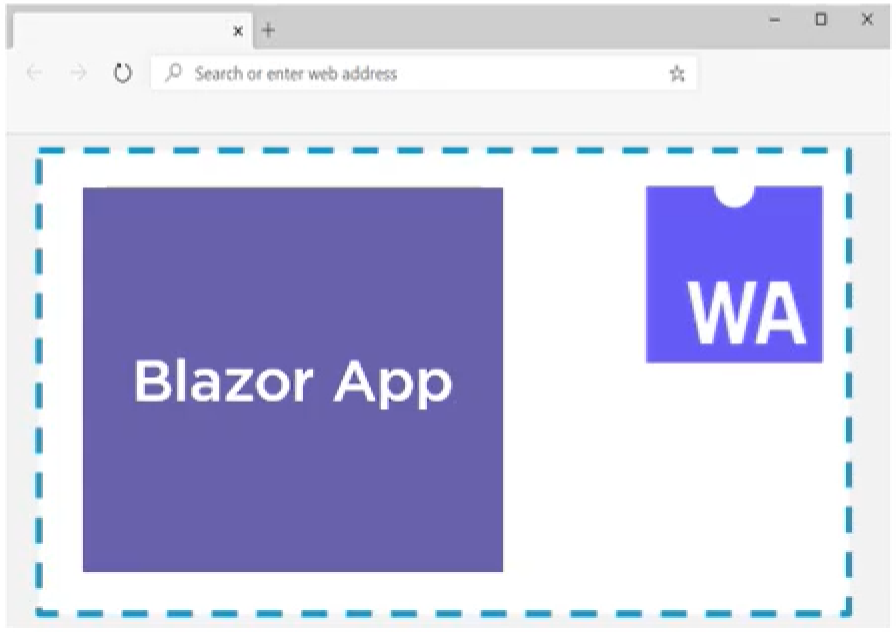
$\xrightarrow[\text{Side}]{\text{Server}}$

- Code runs in the **Server**
- UI updates over **SignalR**



Blazor WebAssembly

Blazor WebAssembly



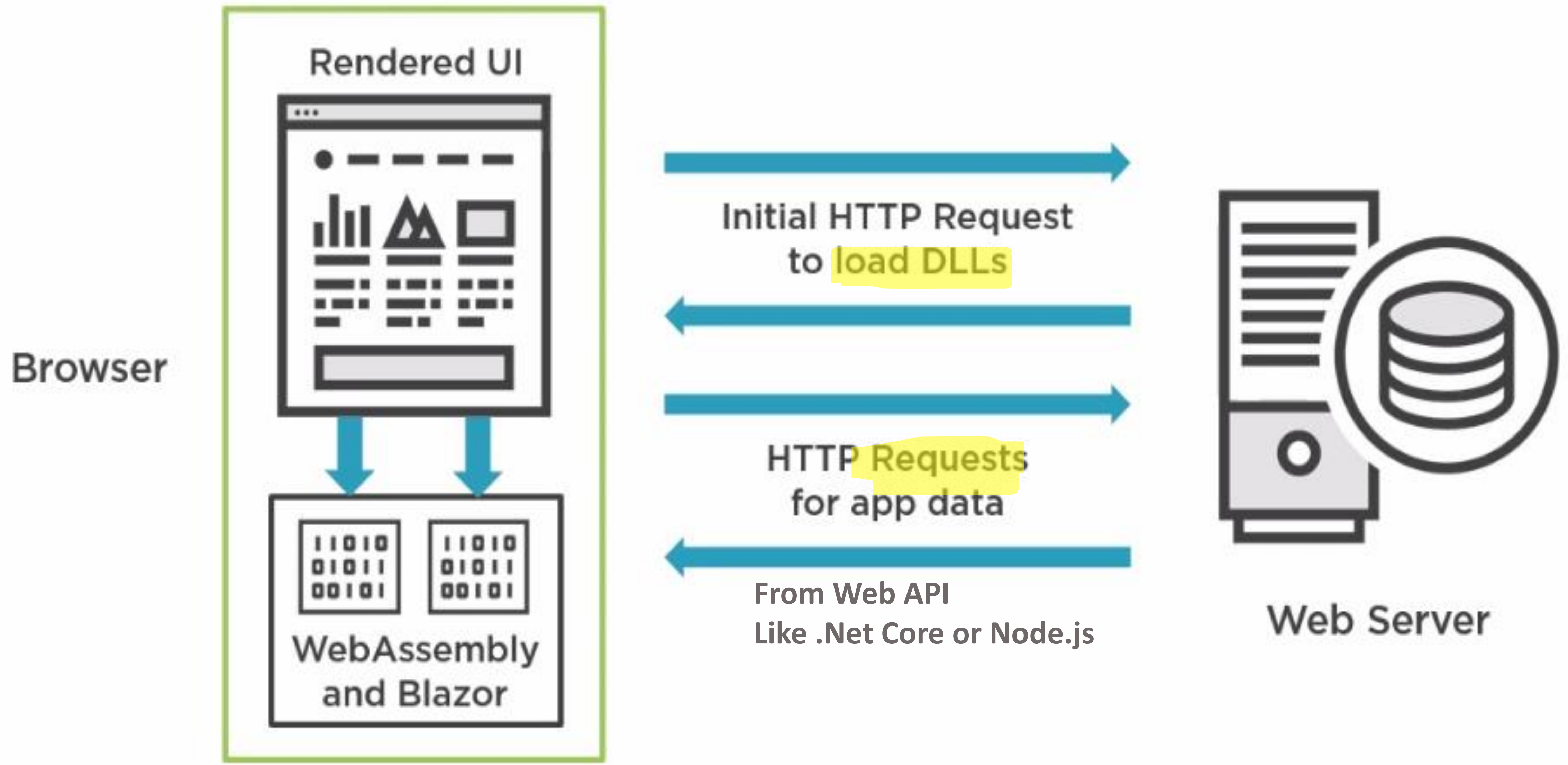
Downloads everything to the browser

- HTML, CSS, JavaScript
- Application (.NET Standard DLLs)
- .NET Runtime

Runs on WebAssembly

No server connection needed

Understanding Blazor WebAssembly



From Web API
Like .Net Core or Node.js

Blazor WebAssembly Pros & Cons

Pros

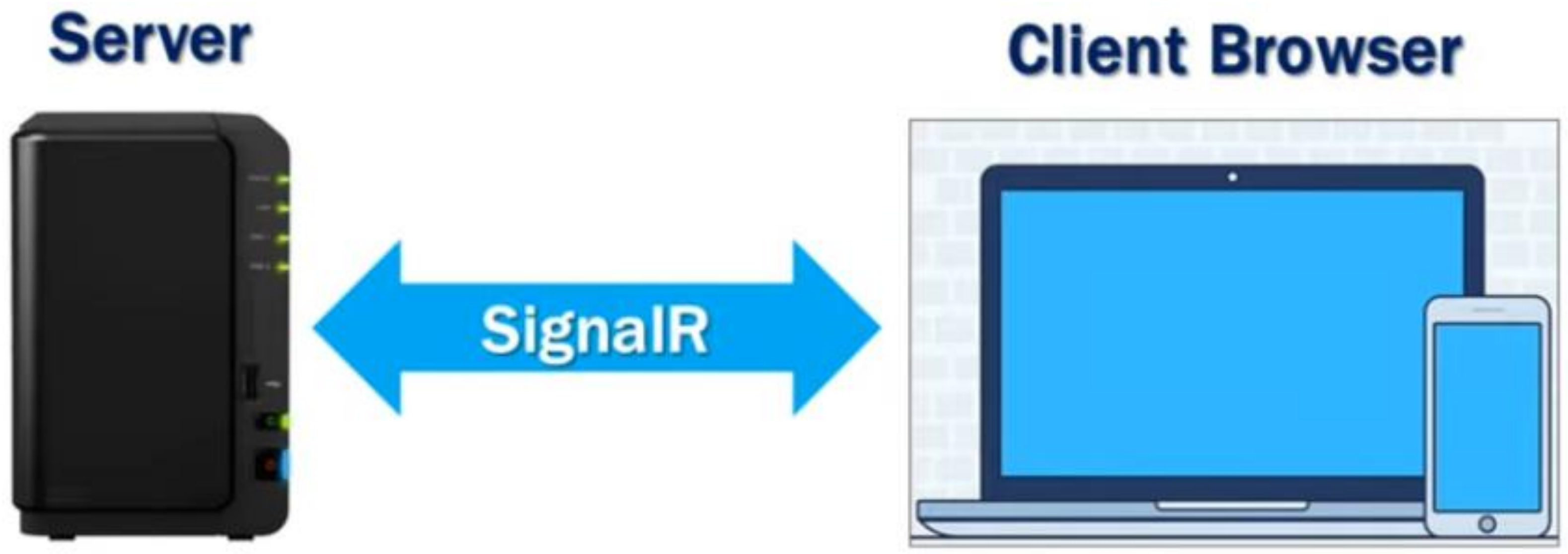
- Near native performance
- Offline support
- No server needed Ex. (.Net)
- Uses client side

- SPA user Experience
- Active server connection not required
- Runs on all modern browsers (no plugin need)

Cons

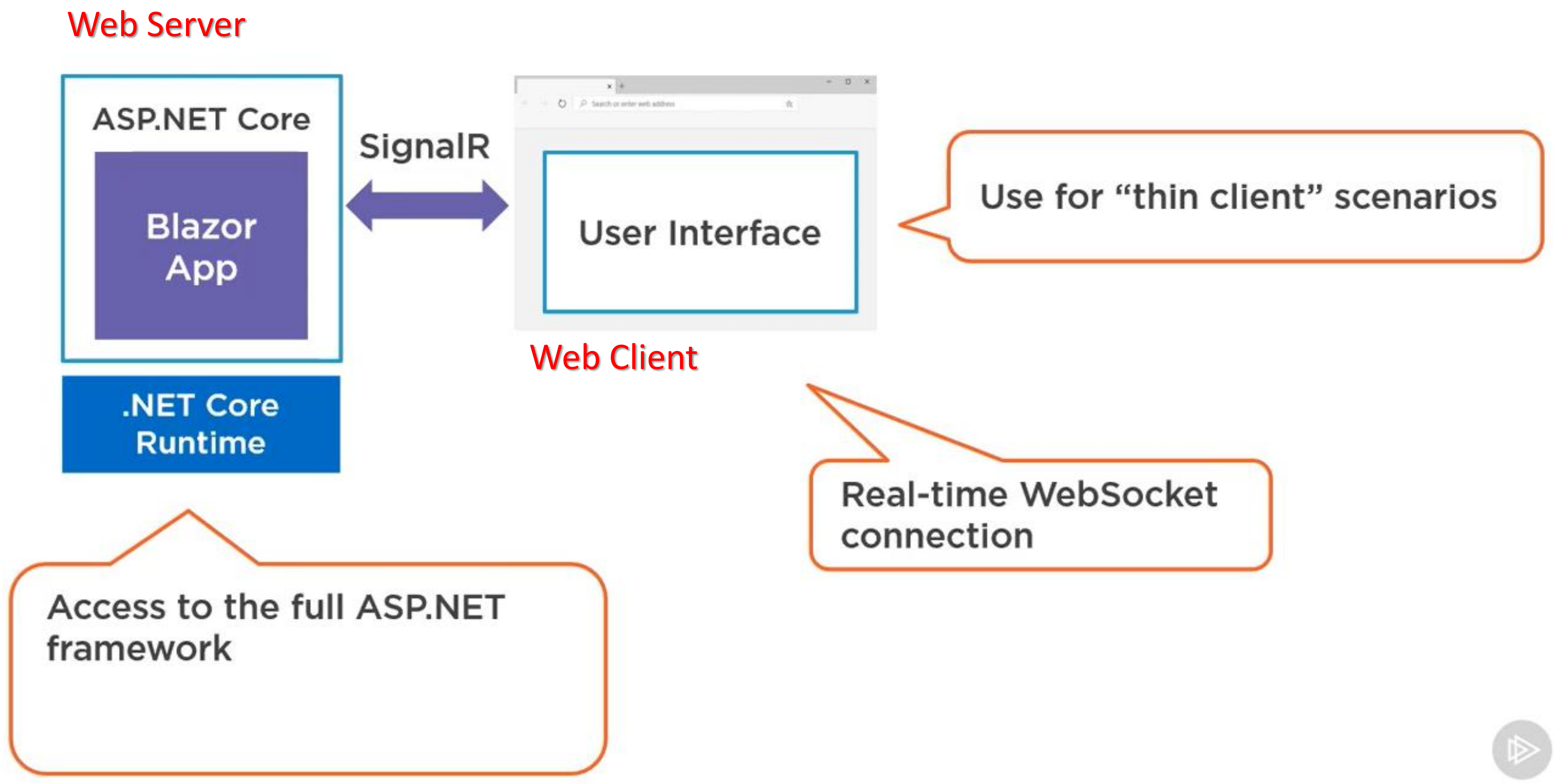
- Restricted to browser capabilities
- Longer loading time
- Client-sided secrets
- Older Browser might not be supported
- Debugging Support

- Webassembly is required



Blazor Server

Blazor Server


Web Server
ASP.NET Core
Blazor App
.NET Core Runtime
SignalR
User Interface
Web Client
Use for “thin client” scenarios
Real-time WebSocket connection
Access to the full ASP.NET framework

Blazor Server Pros & Cons

Pros

- Small downloads
- Full Asp.Net capabilities
- WebAssembly not required
- Server-side secrets

- **Full Debugging support**
- **Works with all server-side APIs**
- **All The Client needs to use the app is a browser**
- **Application Loads mush Faster**

Cons

- No offline support
- Need a server
- Less performant Network Delay

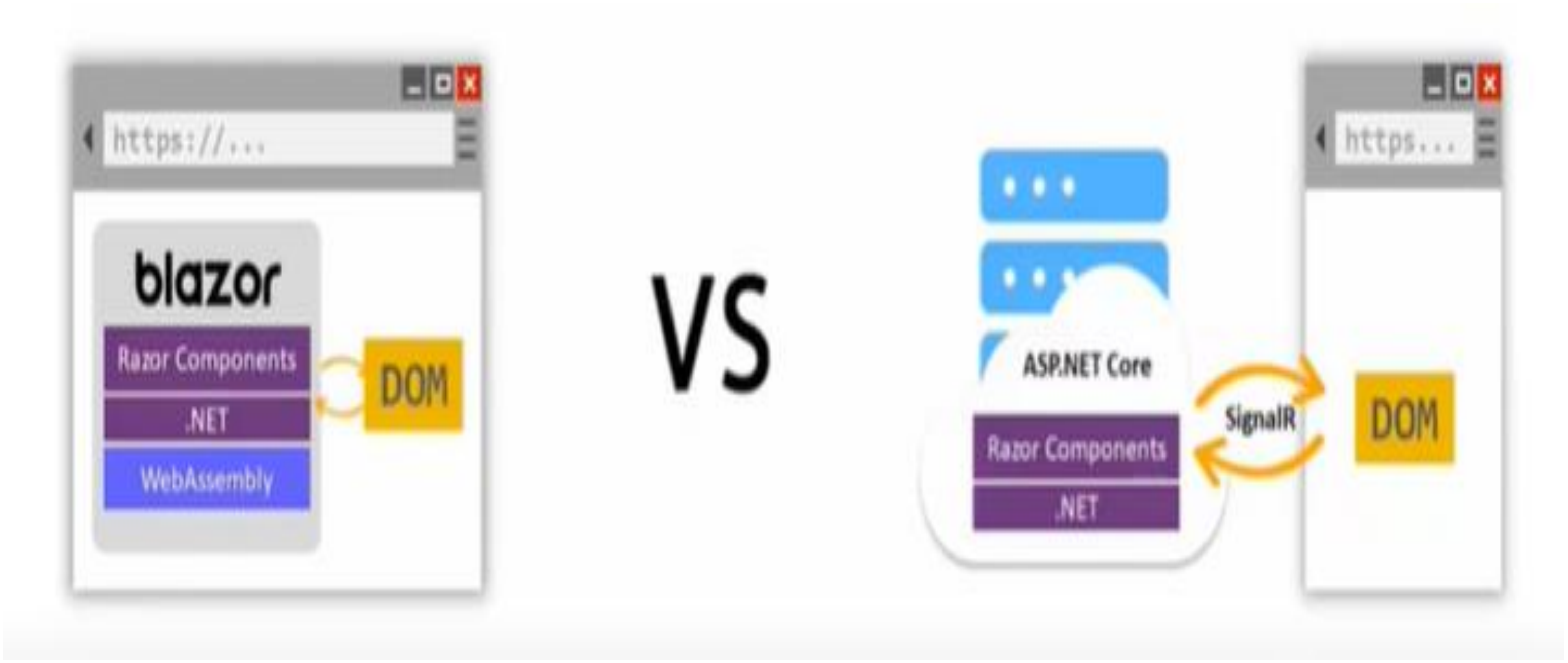
- **Asp.net Core server is required**

What to Use When?

	Blazor WebAssembly	Blazor Server
When you need near-native performance	●	
When you need to connect to server-side resources		●
When you can’t rely on WebAssembly		●
When you need to work offline	●	
When you don’t want to run an ASP.NET Core server	●	
When you want to create fast, interactive web apps with C#	●	●

Blazor webAssembly

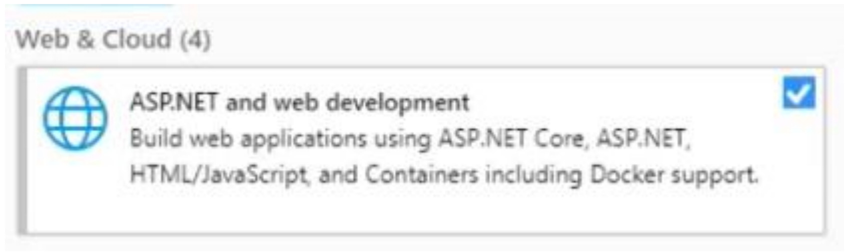
Blazor Server



Development Tools

Development Envirnoment

- The development tools available for Blazor are the following:
  1. Visual Studio—IDE
  2. Visual Studio Code—IDE
  3. C#—programming language
  4. .NET Core—development platform



Web & Cloud (4)

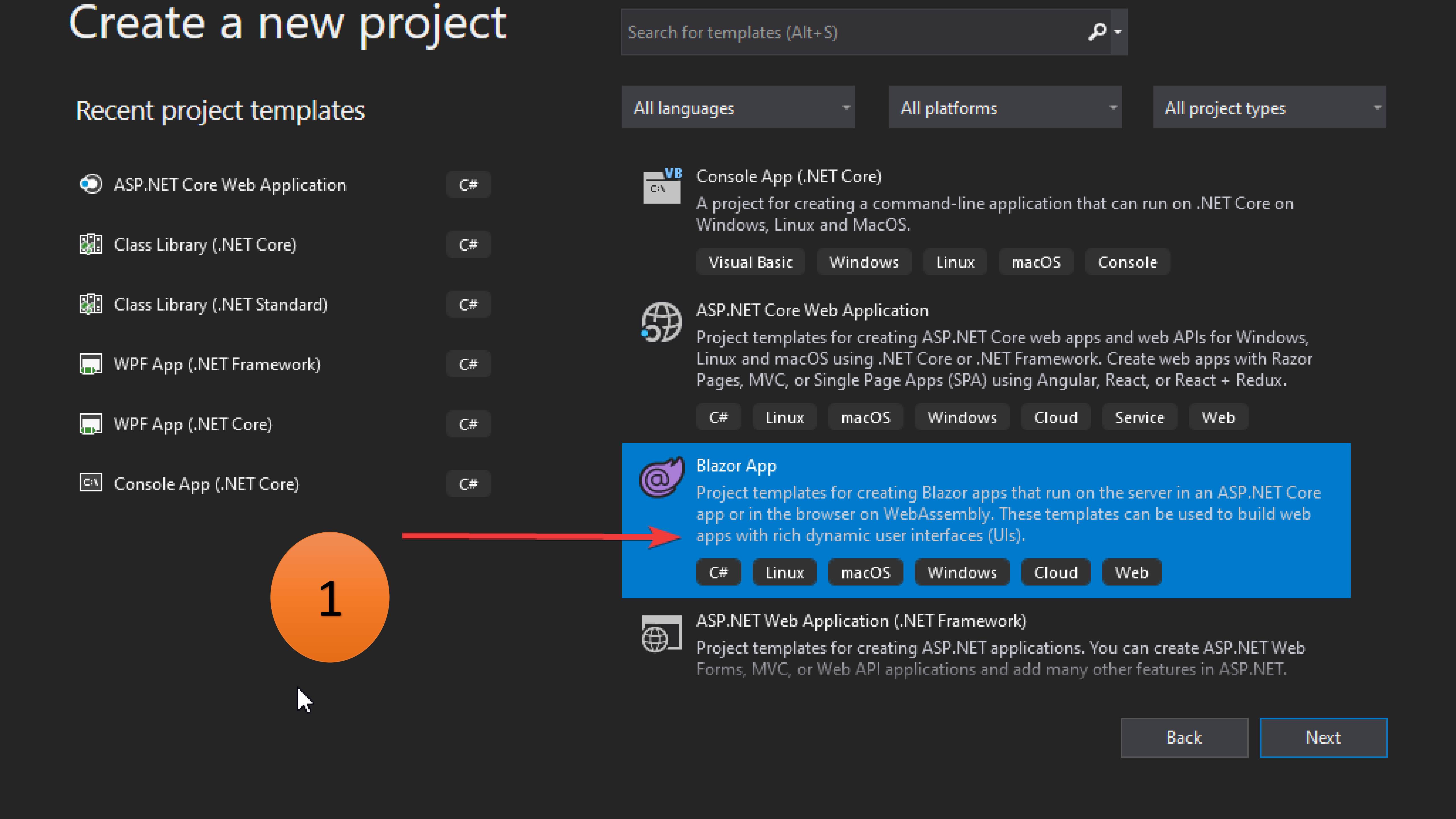
ASP.NET and web development
Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker support.

.NET Core CLI

```
dotnet new blazorwasm -h
dotnet new blazorserver -h
```

For More

Blazor WebAssembly Project Structure



Create a new project

Search for templates (Alt+S)

Recent project templates

ASP.NET Core Web Application	C#
Class Library (.NET Core)	C#
Class Library (.NET Standard)	C#
WPF App (.NET Framework)	C#
WPF App (.NET Core)	C#
Console App (.NET Core)	C#

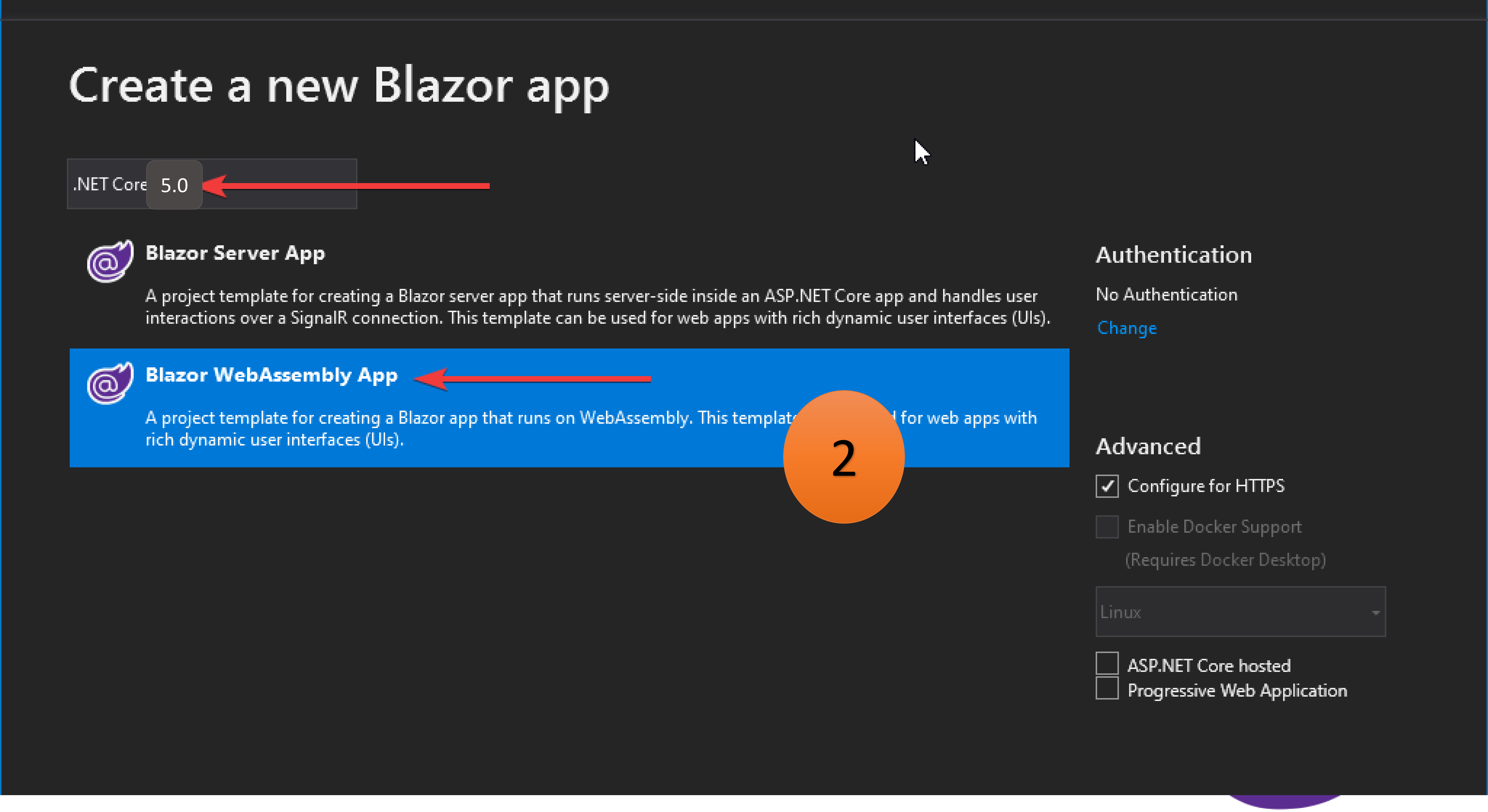
All languages | All platforms | All project types

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic | Windows | Linux | macOS | Console

ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# | Linux | macOS | Windows | Cloud | Service | Web

Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# | Linux | macOS | Windows | Cloud | Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.



Create a new Blazor app

.NET Core 5.0

Blazor Server App

A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).

Blazor WebAssembly App

A project template for creating a Blazor app that runs on WebAssembly. This templat[illegible] for web apps with rich dynamic user interfaces (UIs).

Authentication

No Authentication

Change

Advanced

[✓] Configure for HTTPS

[ ] Enable Docker Support

(Requires Docker Desktop)

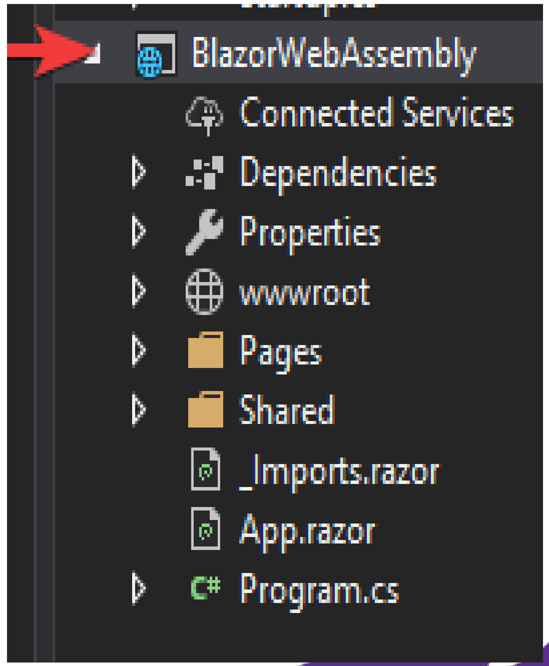
Linux

[ ] ASP.NET Core hosted

[ ] Progressive Web Application

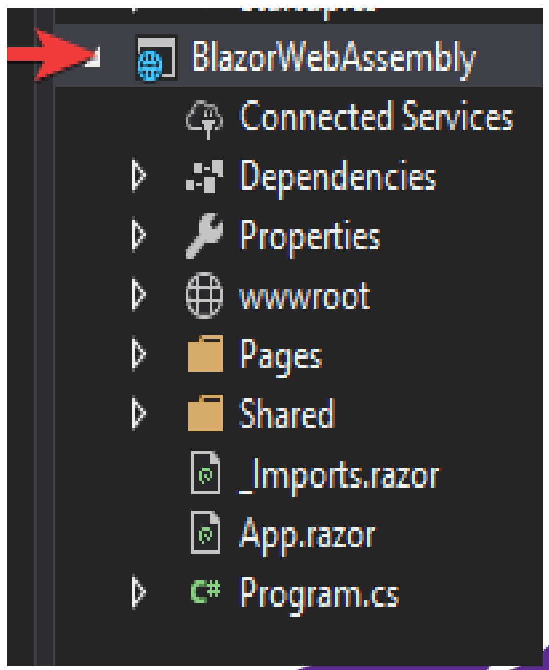
Blazor WebAssembly Project Structure

- **Program.cs:**
  - The app's entry point that sets up the WebAssembly host:
  - Specify the Root Component to Start "App"
    - Specified as the div DOM element with an id of app (<div id="app">Loading...</div> in wwwroot/index.html)
  - Services are added and configured
    - builder.Services.AddSingleton <IMyDependency, MyDependency>()



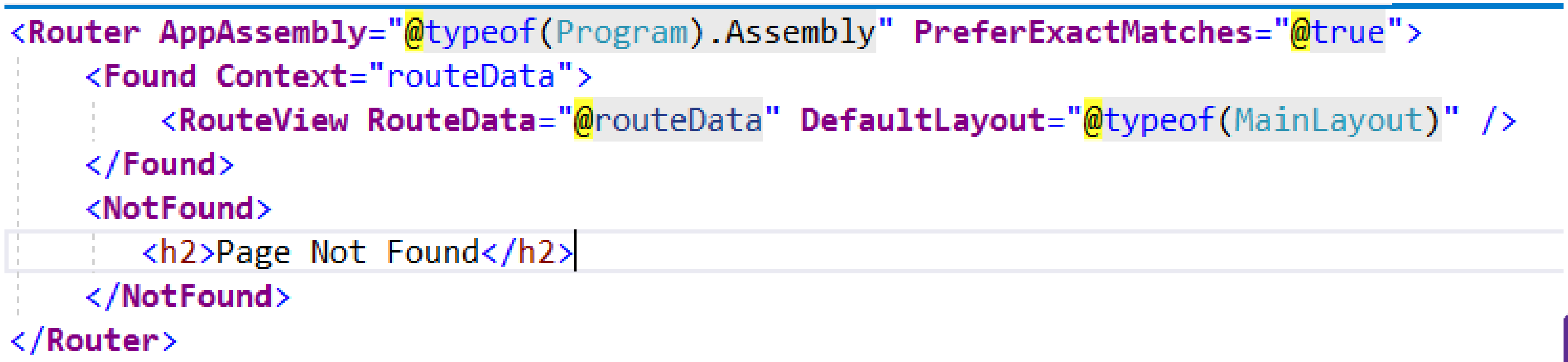
Blazor WebAssembly Project Structure

- **The App component**
  - The root component of the app.
  - That sets up **client-side routing** using the Router component.
  - The Router component intercepts browser navigation and renders the page that matches the requested address.



RouteComponent

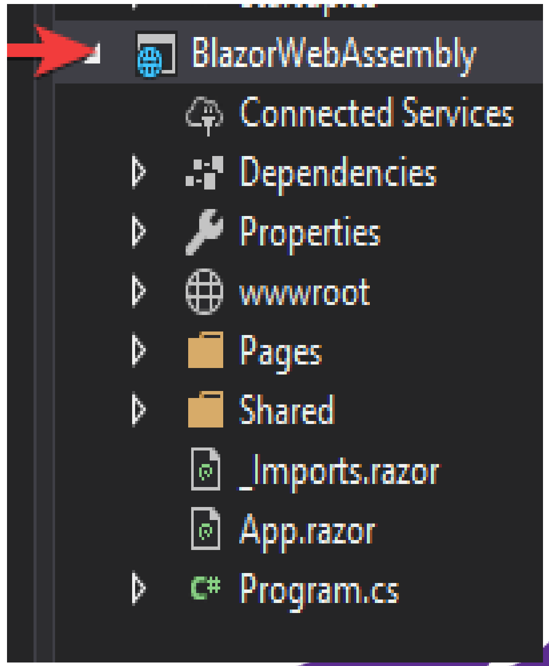
- Compoent like other component but it blazor framework component



```
<Router AppAssembly="@typeof(Program).Assembly" PreferExactMatches="@true">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
    </Found>
    <NotFound>
        <h2>Page Not Found</h2>
    </NotFound>
</Router>
```

Blazor WebAssembly Project Structure

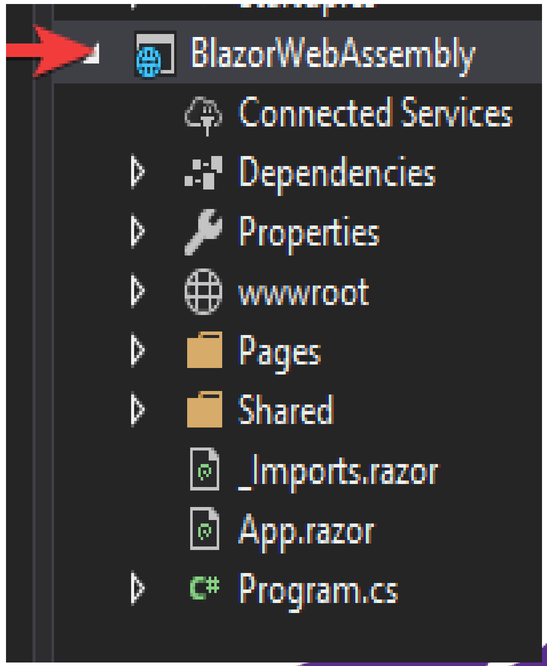
- **Pages folder:**
  - Contains the routable components/pages (.razor) that make up the Blazor app.
  - The route for each page is specified using the @page directive



Blazor WebAssembly Project Structure

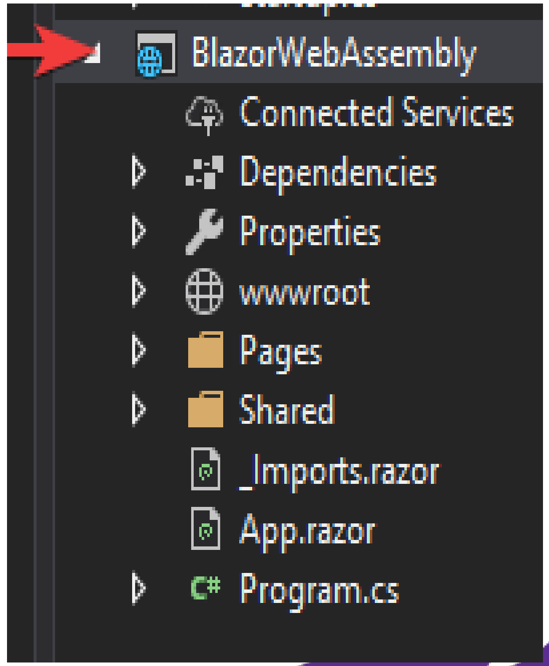
- **_Imports.razor:**
  - Includes common Razor directives to include in the app's components (.razor),
  - such as @using directives for namespaces.

- **Properties/launchSettings.json**:
  - Holds development environment configuration.



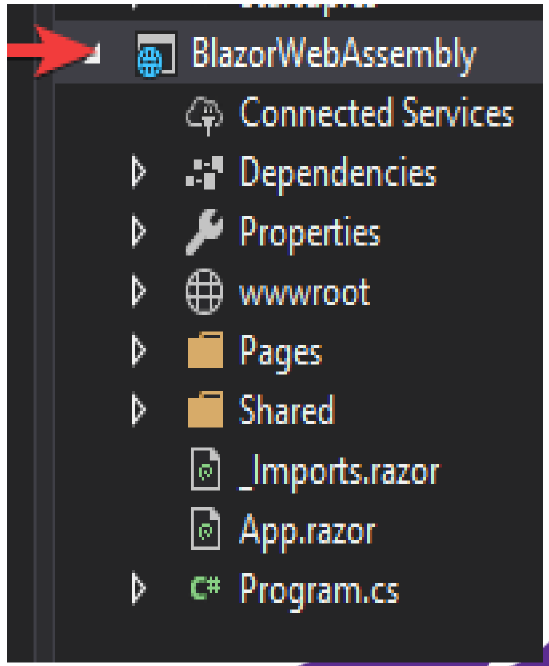
Blazor WebAssembly Project Structure

- **wwwroot**:
  - containing the app's **public static assets,**
  - including appsettings.json and environmental app settings files for configuration settings.
  - The index.html webpage is the root page of the app implemented as an HTML page:

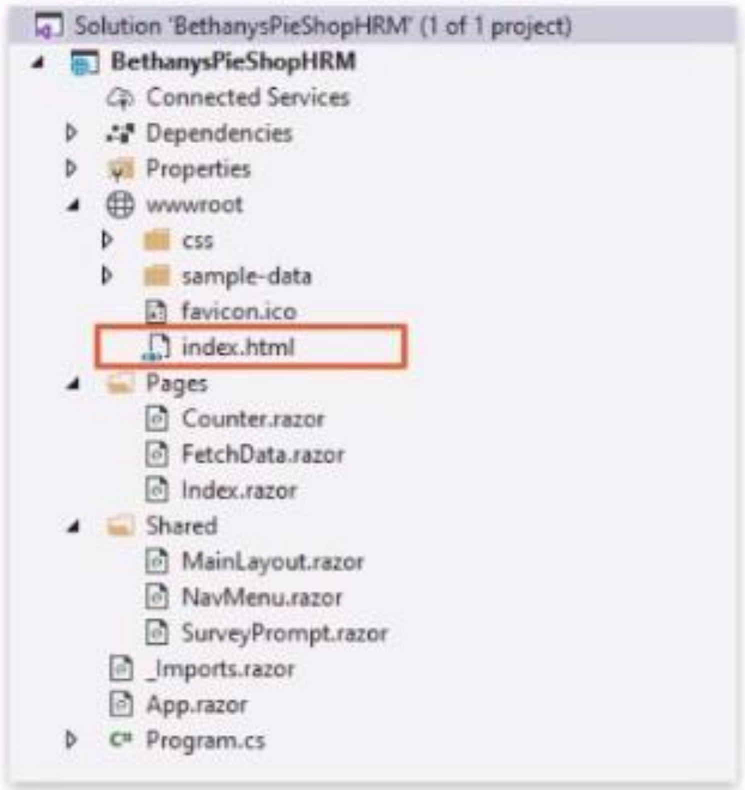


Blazor WebAssembly Project Structure

- **wwwroot**:
  - When any page of the app is initially requested, this page is rendered and returned in the response.
  - The page specifies where the root App component is rendered. The component is rendered at the location of the div DOM element with an id of app (<div id="app">Loading...</div>).



Index.html


Solution 'BethanysPieShopHRM' (1 of 1 project)
BethanysPieShopHRM
Connected Services
Dependencies
Properties
wwwroot
css
sample-data
favicon.ico
index.html
Pages
Counter.razor
FetchData.razor
Index.razor
Shared
MainLayout.razor
NavMenu.razor
SurveyPrompt.razor
_Imports.razor
App.razor
Program.cs

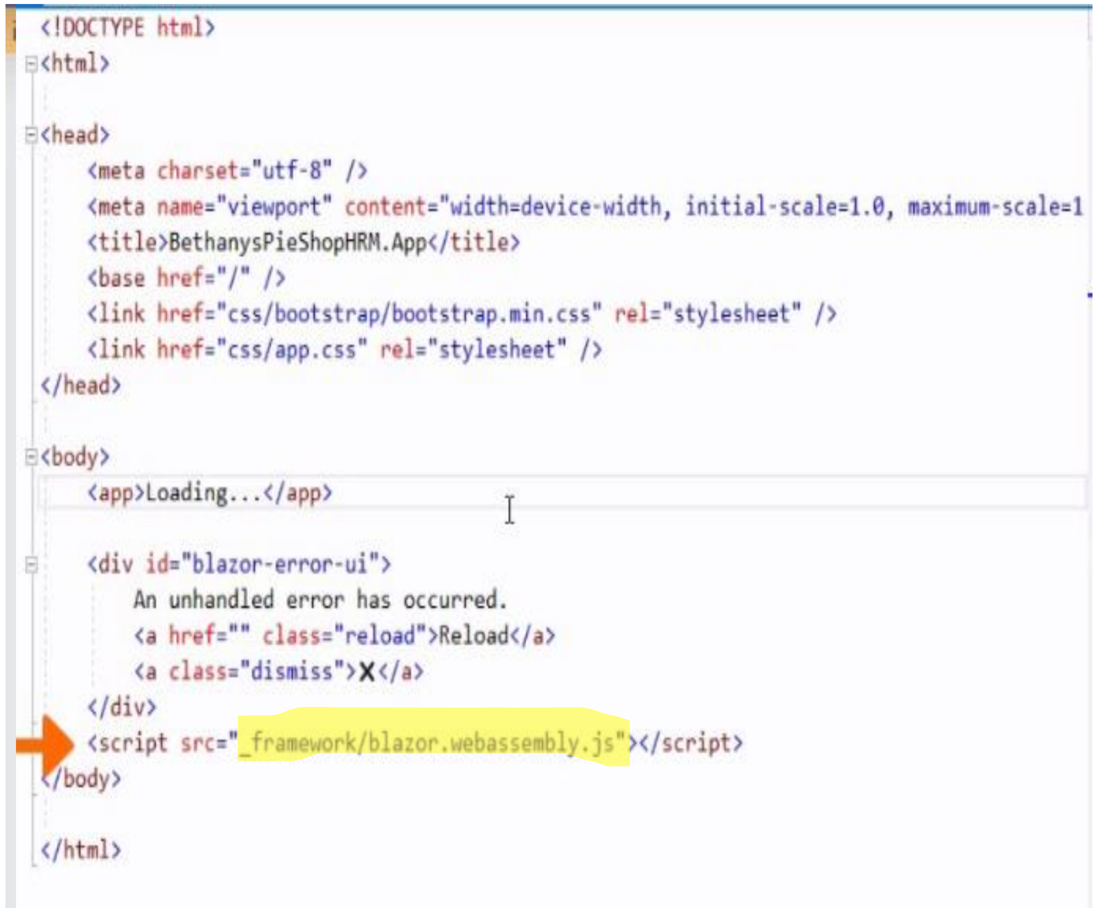
Hosting page

Plain HTML

Trigger loading of your Blazor app

- blazor.webassembly.js

Index.html



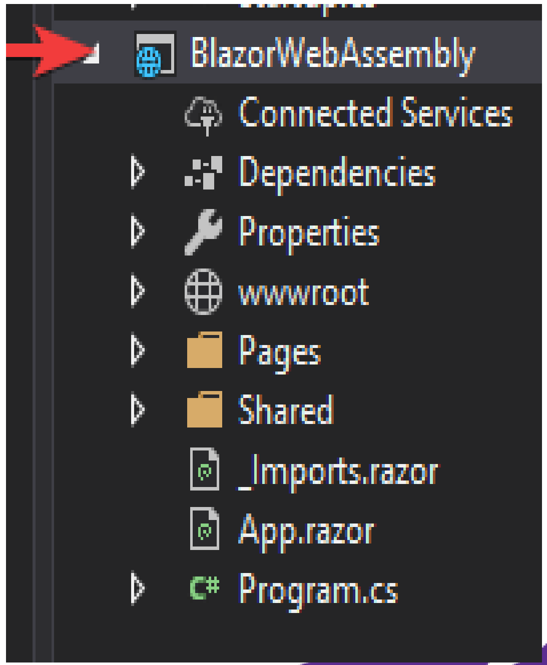
```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1
    <title>BethanysPieShopHRM.App</title>
    <base href="/" />
    <link href="css/bootstrap/bootstrap.min.css" rel="stylesheet" />
    <link href="css/app.css" rel="stylesheet" />
</head>

<body>
    <app>Loading...</app>

    <div id="blazor-error-ui">
        An unhandled error has occurred.
        <a href="" class="reload">Reload</a>
        <a class="dismiss">X</a>
    </div>
    <script src="_framework/blazor.webassembly.js"></script>
</body>

</html>
```

The `blazor.webassembly.js` script is provided by the framework and handles:

- Downloading the .NET runtime, the app, and the app's dependencies.
- Initialization of the runtime to run the app.

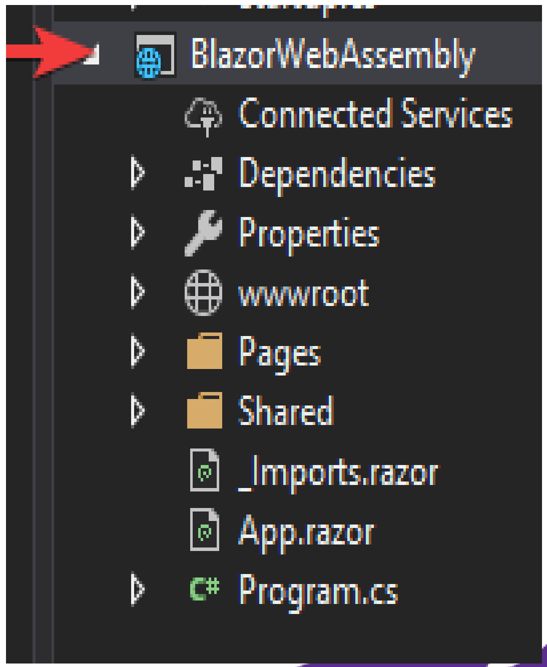
Blazor WebAssembly Project Structure

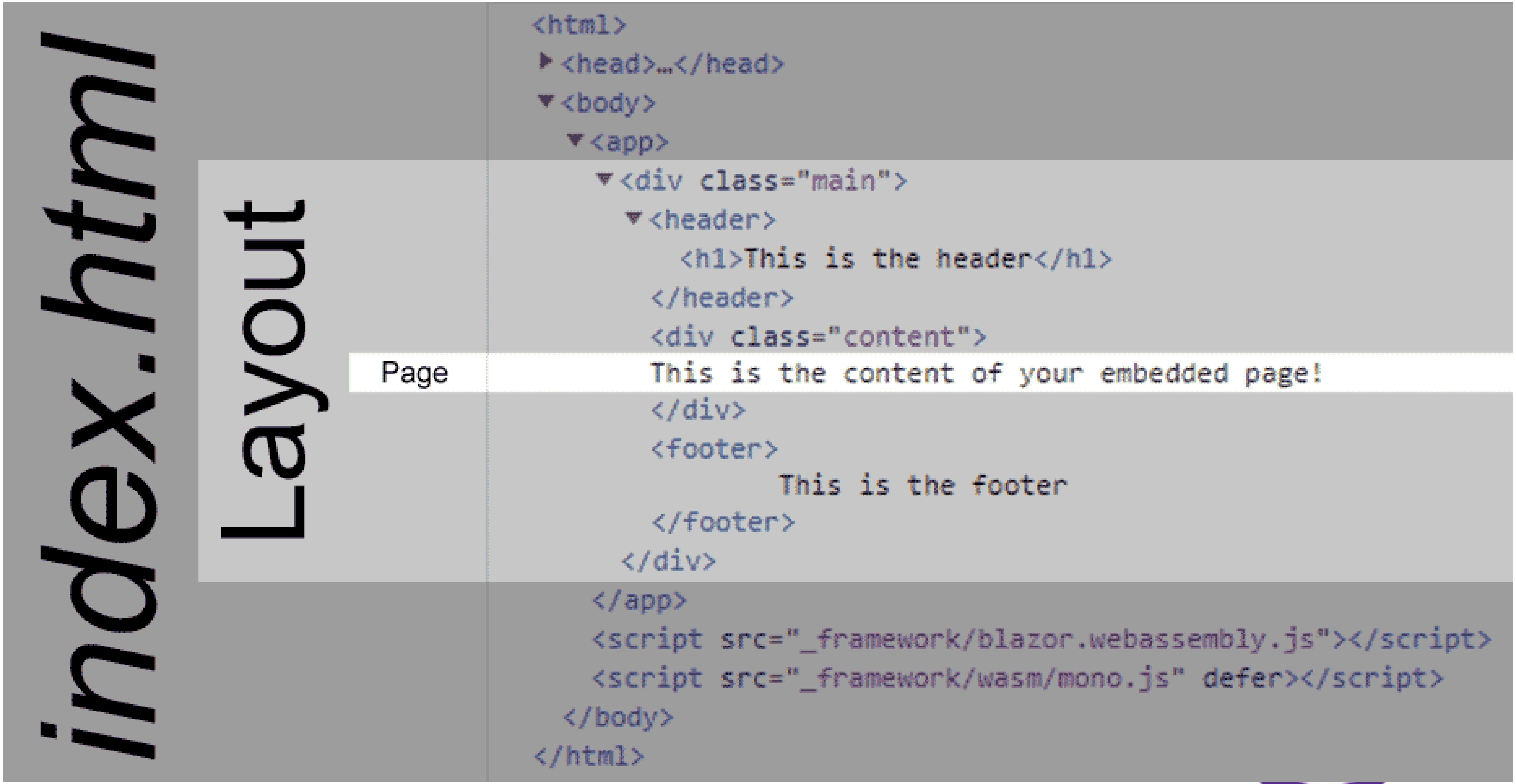
- **Shared folder:** Contains the following shared components and stylesheets:
  - **MainLayout component** (MainLayout.razor): The app's layout component.
  - **MainLayout.razor.css**: Stylesheet for the app's main layout.



Blazor WebAssembly Project Structure

- **Shared folder:** Contains the following shared components and stylesheets:
  - **NavMenu component** (NavMenu.razor): Implements sidebar navigation.
    - Includes the **NavLink component** (NavLink), which renders navigation links to other Razor components.
    - The NavLink component automatically indicates a selected state when its component is loaded, which helps the user understand which component is currently displayed.
  - **NavMenu.razor.css**: Stylesheet for the app's navigation menu.



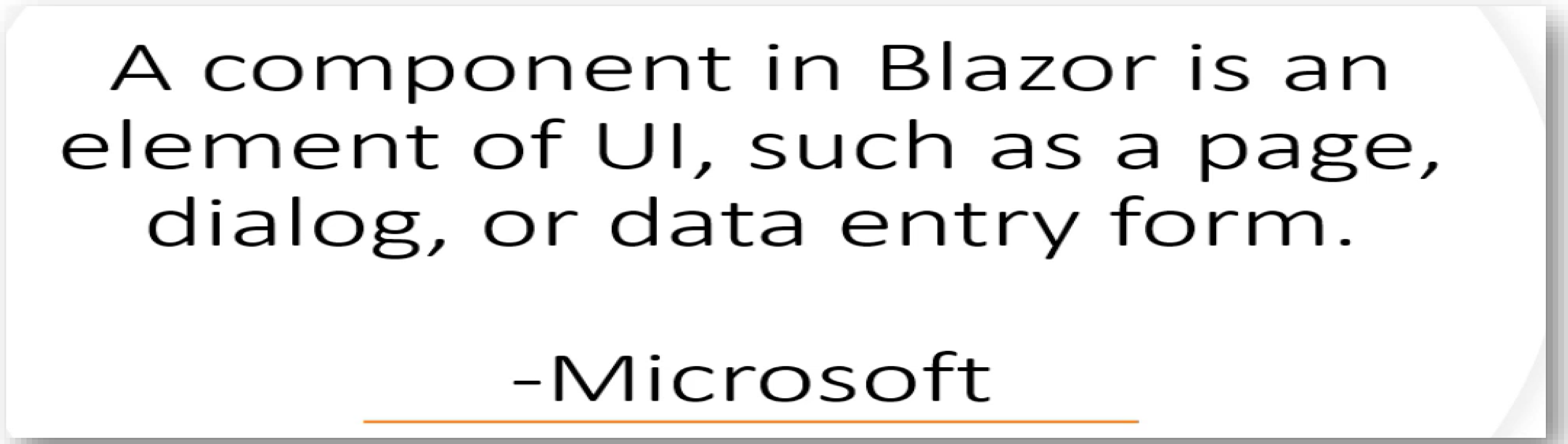


```
<html>
 ▶<head>…</head>
 ▼<body>
   ▼<app>
     ▼<div class="main">
       ▼<header>
          <h1>This is the header</h1>
        </header>
        <div class="content">
        This is the content of your embedded page!
        </div>
        <footer>
                This is the footer
        </footer>
      </div>
    </app>
    <script src="_framework/blazor.webassembly.js"></script>
    <script src="_framework/wasm/mono.js" defer></script>
  </body>
</html>
```

Part2 : Blazor Component

Demo Steps

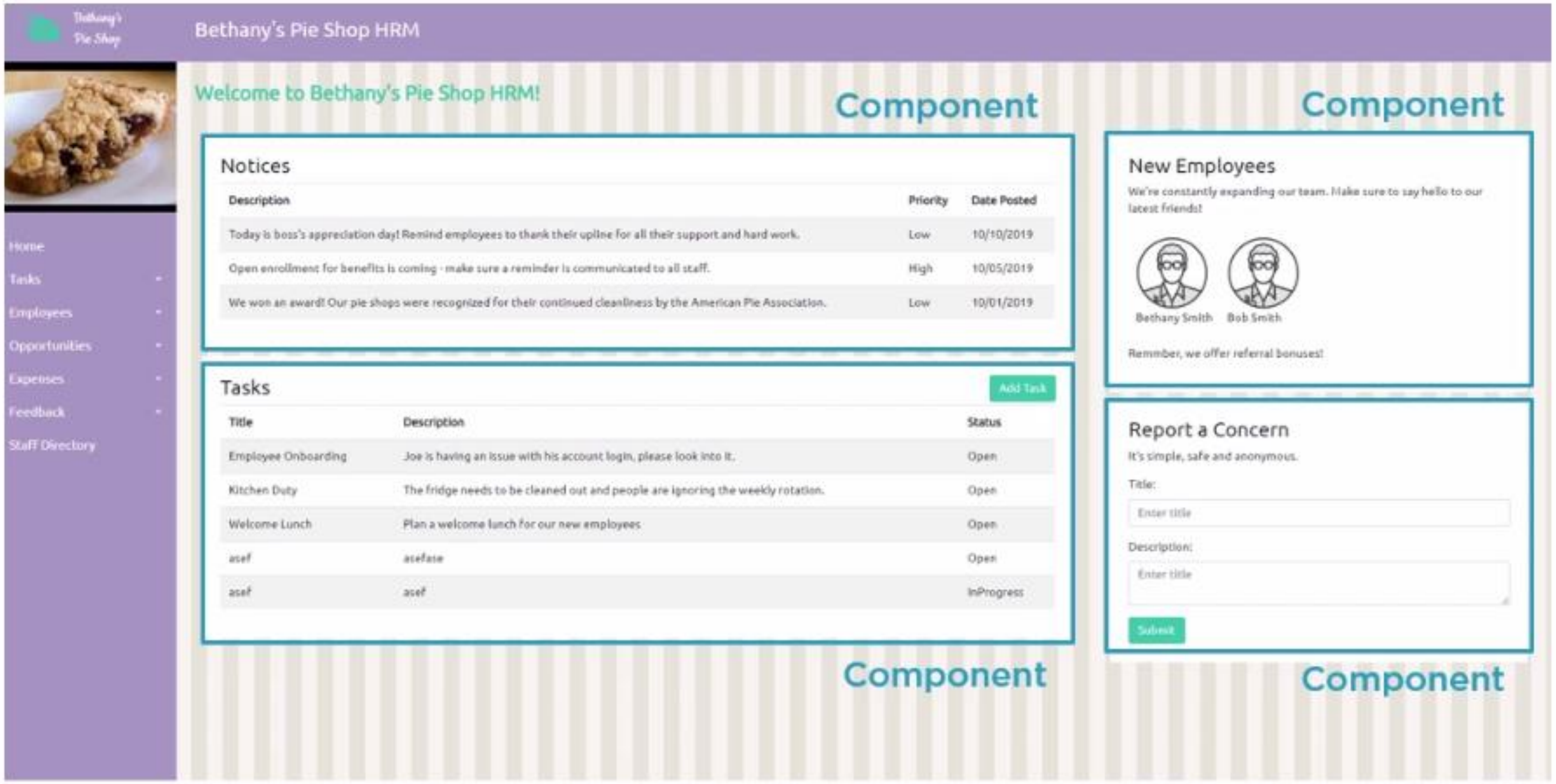
- Create component with static
- Make rout to this component using @page
- Go to navbar make link
- Used this component in another component like parent & child
- Make property to change static data to dynamic
- Put [parameter] to change property value from parnt component
- Change rout to send paremeter values from route attribute
- Add event to send data from Child to parent


A component in Blazor is an element of UI, such as a page, dialog, or data entry form.

-Microsoft

- Blazor consider component base framerwork which means that a component is the main building block of asp.net core blazor app

Component driven design



Bethany's Pie Shop HRM

Welcome to Bethany's Pie Shop HRM!

Notices

Description	Priority	Date Posted
Today is boss's appreciation day! Remind employees to thank their upline for all their support and hard work.	Low	10/10/2019
Open enrollment for benefits is coming - make sure a reminder is communicated to all staff.	High	10/05/2019
We won an award! Our pie shops were recognized for their continued cleanliness by the American Pie Association.	Low	10/01/2019

Tasks

Title	Description	Status
Employee Onboarding	Joe is having an issue with his account login, please look into it.	Open
Kitchen Duty	The fridge needs to be cleaned out and people are ignoring the weekly rotation.	Open
Welcome Lunch	Plan a welcome lunch for our new employees	Open
asef	asefase	Open
asef	asef	InProgress

New Employees

We're constantly expanding our team. Make sure to say hello to our latest friends!

Bethany Smith Bob Smith

Remmber, we offer referral bonuses!

Report a Concern

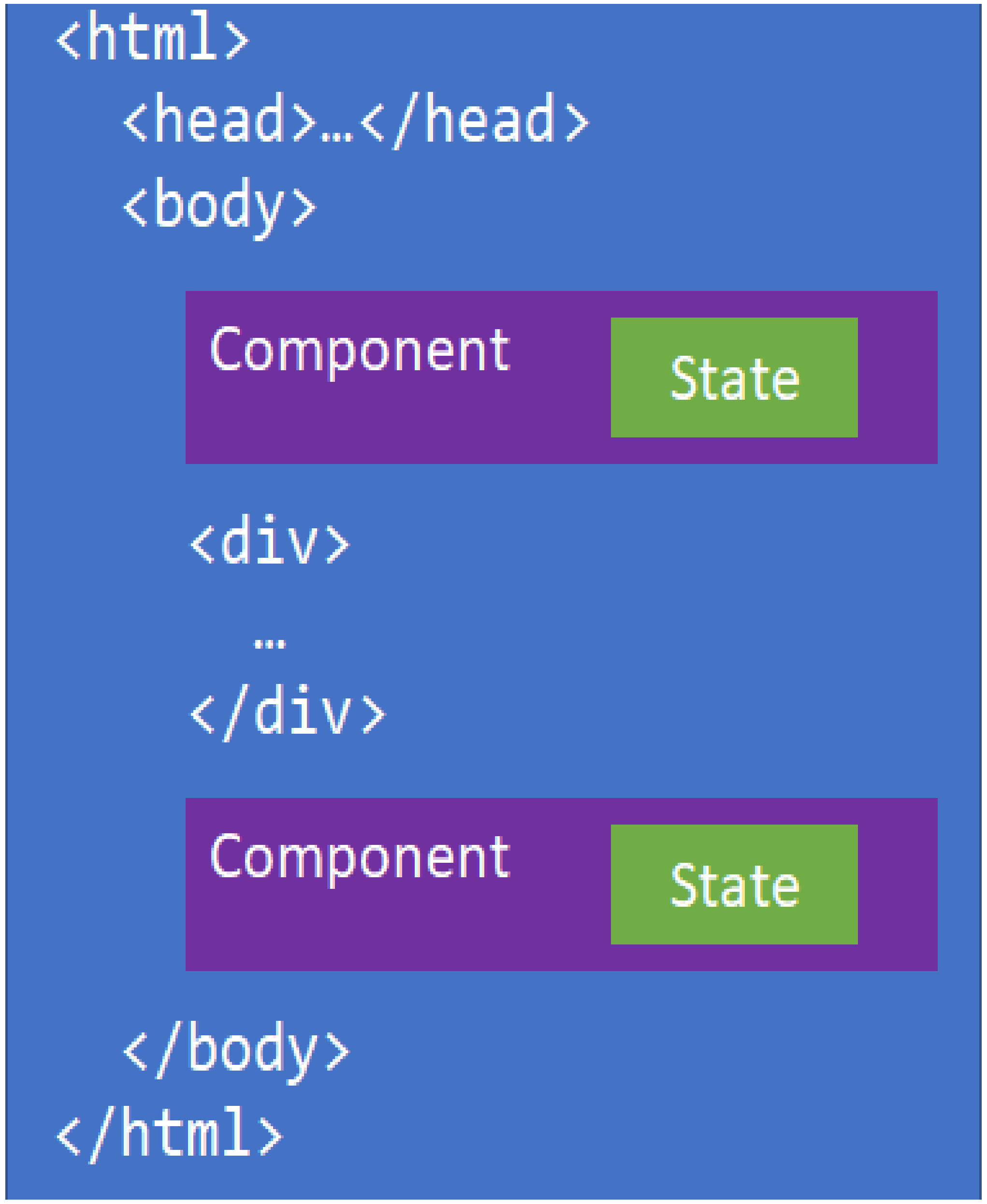
It's simple, safe and anonymous.

Title:

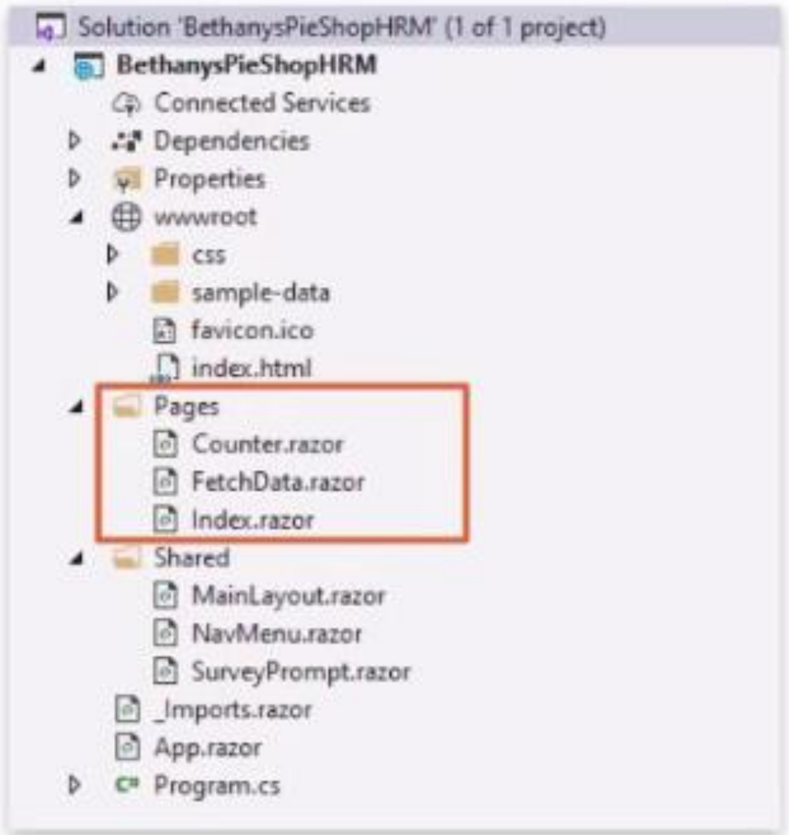
Description:

Blazor Component

- .NET classes that represent a reusable piece of UI.
- Each component maintains its own state and specifies its own rendering logic, which can include rendering other components.
- Components specify event handlers for specific user interactions to update the component's state.



Blazor Component



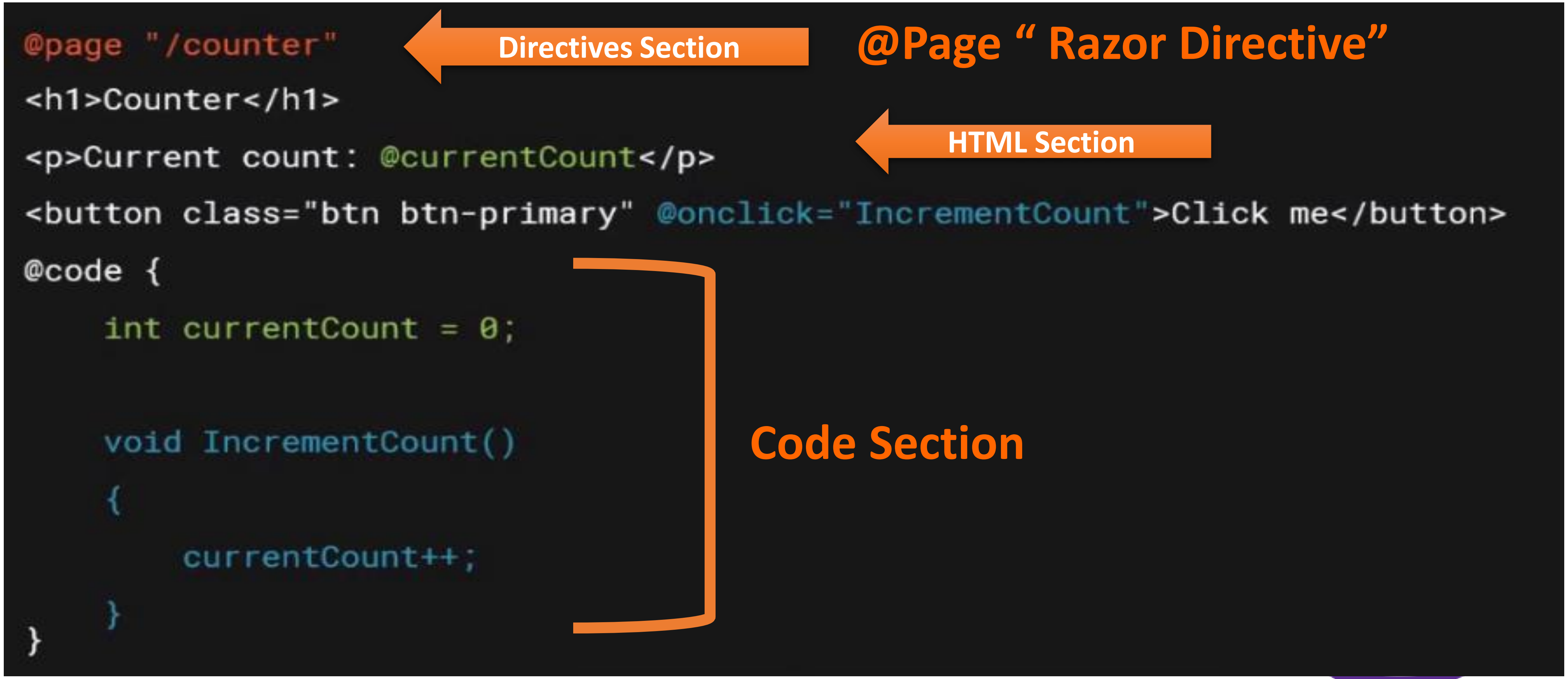
*.razor files

Components are building blocks

Name must start with uppercase

Class generated upon compilation

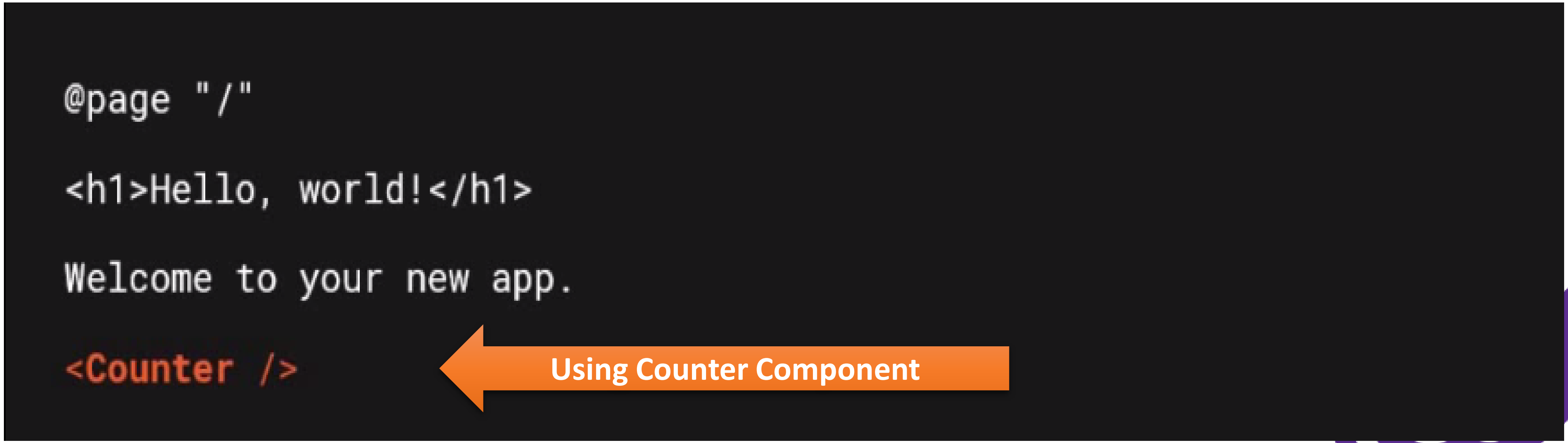
First Component



```
@page "/counter"
<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
@code {
    int currentCount = 0;

    void IncrementCount()
    {
        currentCount++;
    }
}
```

Using Component Inside Component

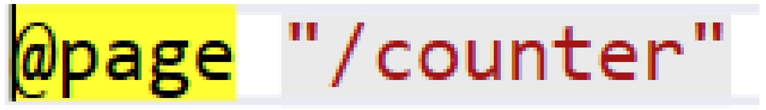
- Components can include other components by declaring them using HTML element syntax



```
@page "/"

<h1>Hello, world!</h1>

Welcome to your new app.

<Counter />
```

Razor Directives

- control many aspects of how a Razor component is compiled. Examples include the component's:
  - Namespace , Base class , Implemented interfaces ,Generic parameters, Imported namespaces, Routes
- Razor directives start with the @ character and are typically used at the start of a new line at the start of the file.

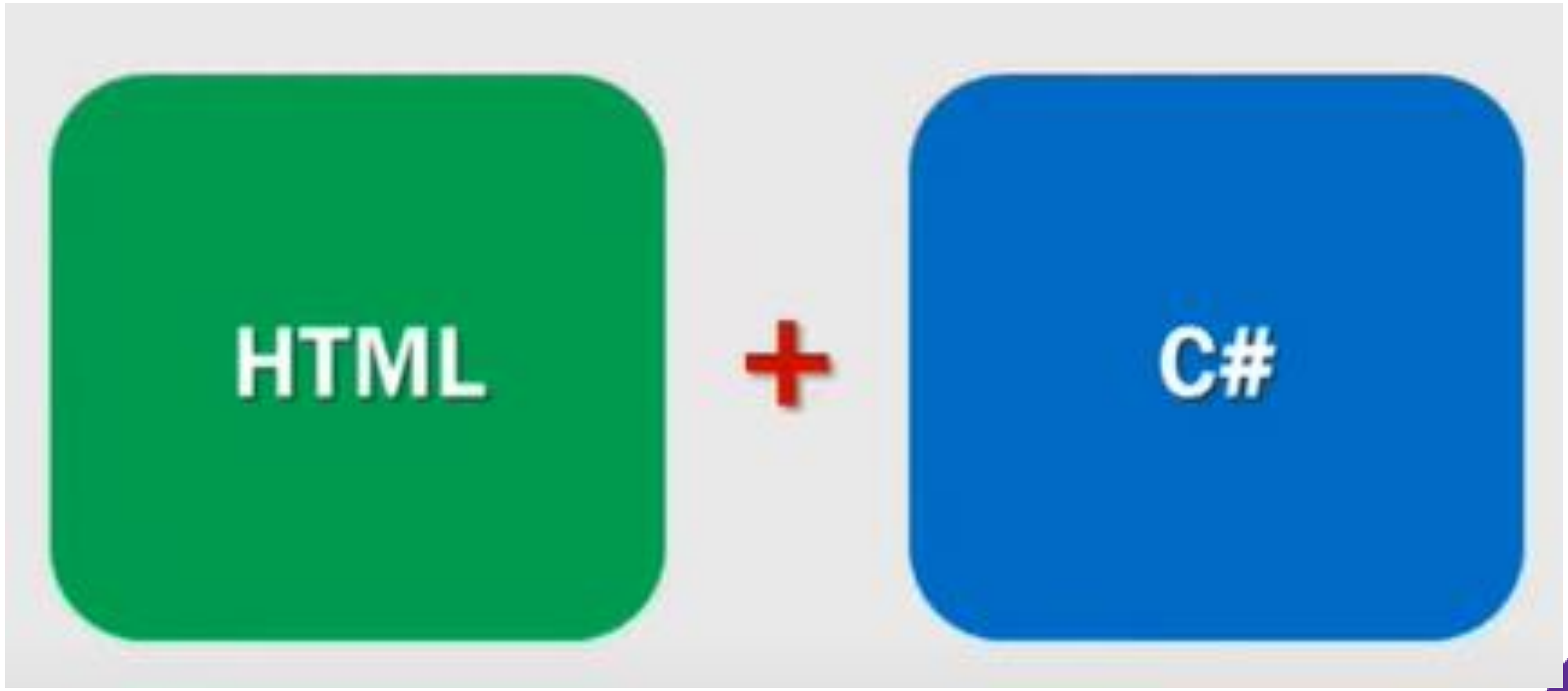


```
@page "/counter"
```

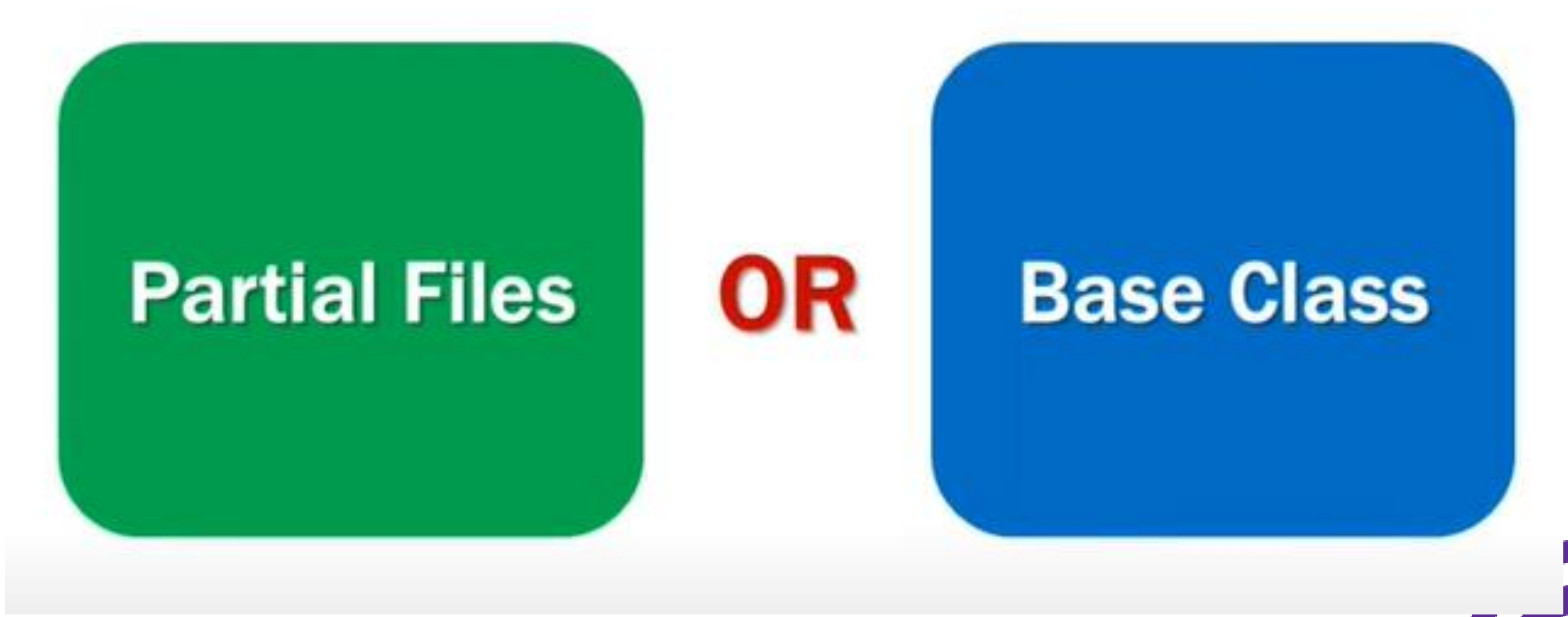
Razor Directive

Directive	Description	Example
@attribute	Adds a class-level attribute to the component	@attribute [Authorize]
@code	Adds class members to the component	@code { ... }
@implements	Implements the specified interface	@implements IDisposable
@inherits	Inherits from the specified base class	@inherits MyComponentBase
@inject	Injects a service into the component	@inject IJSRuntime JS
@layout	Specifies a layout component for the component	@layout MainLayout
@namespace	Sets the namespace for the component	@namespace MyNamespace
@page	Specifies the route for the component	@page "/product/{id}"
@typeparam	Specifies a generic type parameter for the component	@typeparam Titem
@using	Specifies a namespace to bring into scope	@using MyComponentNamespace

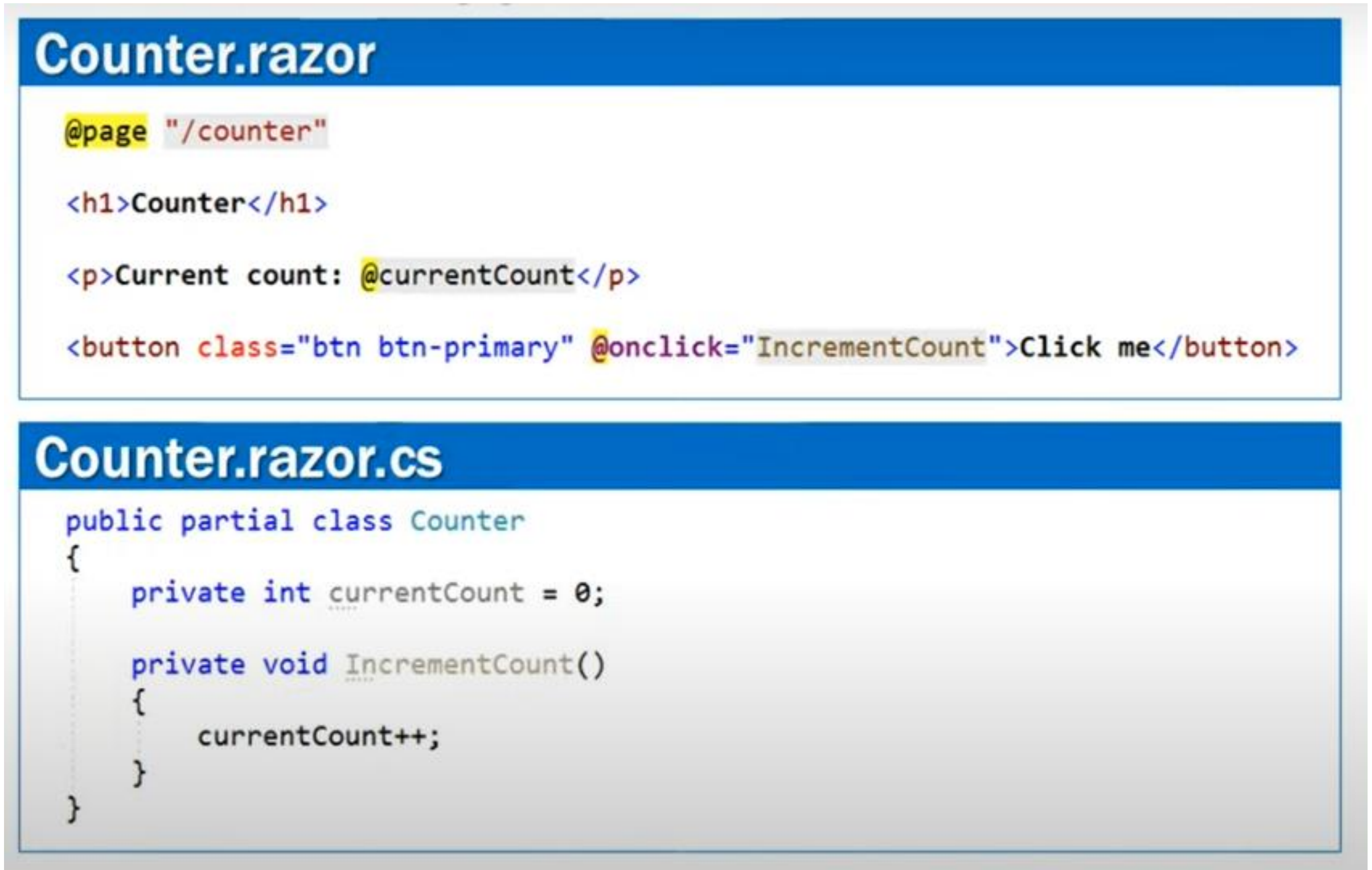
Razor Component



Split Component HTML and C# Code



Partial Files Approach



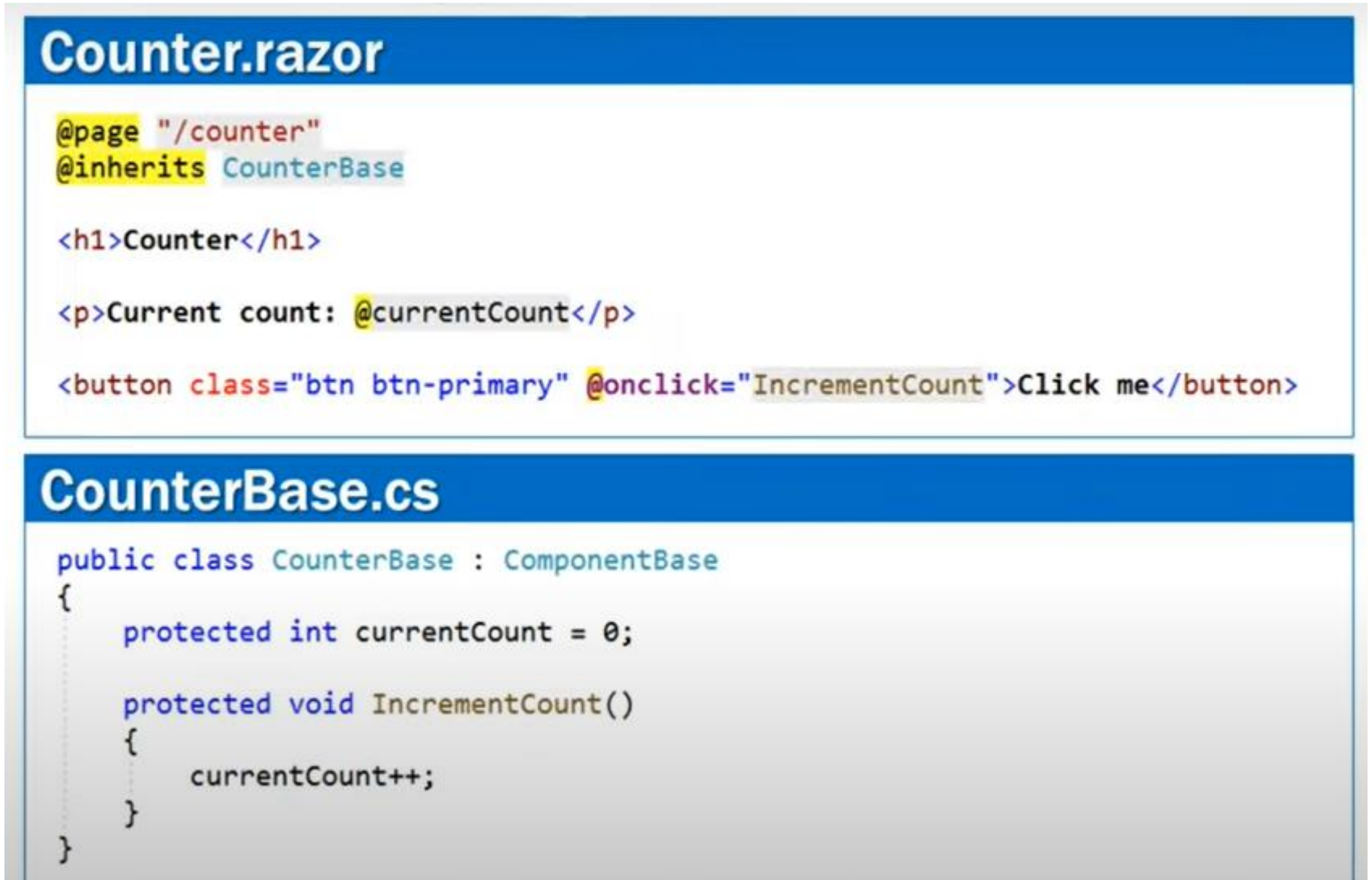
Counter.razor

```
@page "/counter"

<h1>Counter</h1>

<p>Current count: @currentCount</p>

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

Counter.razor.cs

```
public partial class Counter
{
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

Base Class Approach



Counter.razor

```
@page "/counter"
@inherits CounterBase

<h1>Counter</h1>

<p>Current count: @currentCount</p>

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

CounterBase.cs

```
public class CounterBase : ComponentBase
{
    protected int currentCount = 0;

    protected void IncrementCount()
    {
        currentCount++;
    }
}
```

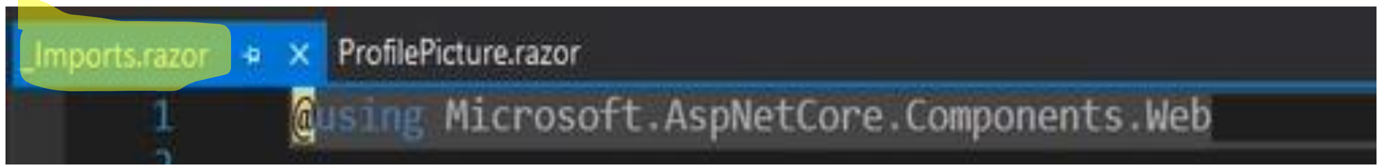
Event Handler

- Each htmt element has number of blazer event start with **@on**

```
<img .. @onclick="ProfileClick"/>
```

C# Method

- All Blazor Event Handler Declare in this namespace



_Imports.razor ProfilePicture.razor

```
@using Microsoft.AspNetCore.Components.Web
```

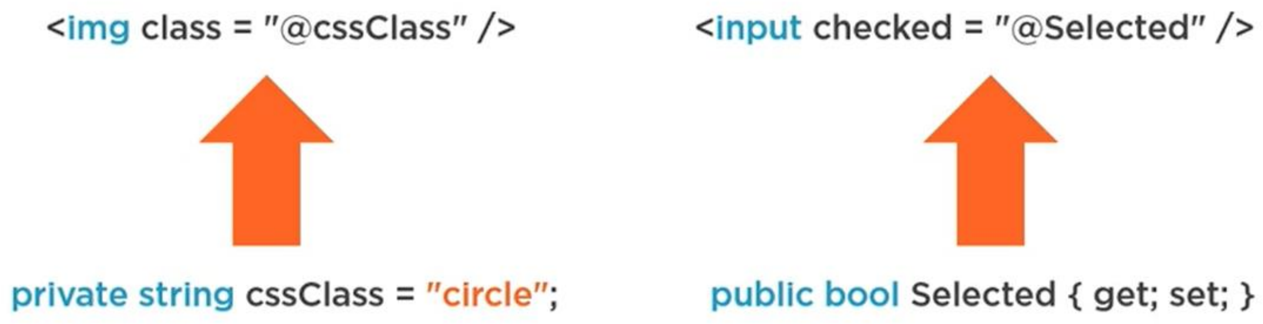
DataBinding

Data binding support in Blazor

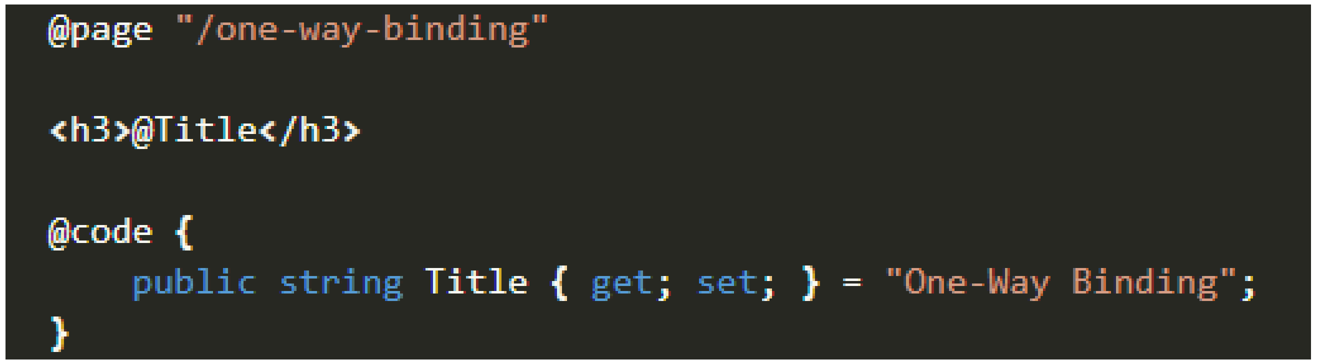
- One-way
- Two-way
- Component parameter

One-way DataBinding

- One-Way binding has a one-directional flow.
- This means that the value is set by the application and then rendered on the page.
- Basically, the user can't modify the value directly on the page since this value can only be set by the application itself.



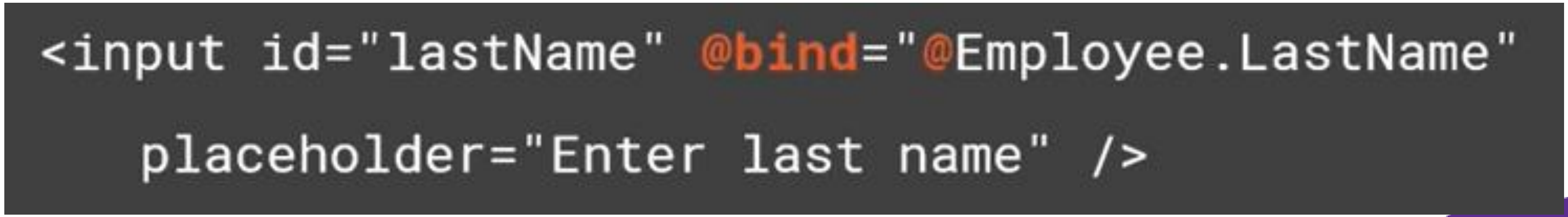
<img class = "@cssClass" />
private string cssClass = "circle";
<input checked = "@Selected" />
public bool Selected { get; set; }



```
@page "/one-way-binding"

<h3>@Title</h3>

@code {
    public string Title { get; set; } = "One-Way Binding";
}
```

Two Way Binding

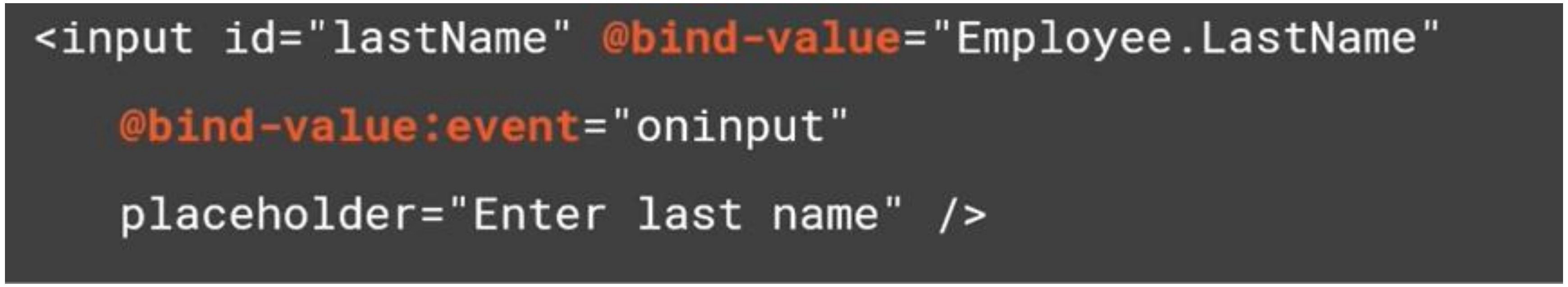
- Binding in two Direction from Model to UI and from UI to Model
- Default binding work when user tabs out of the input



```
<input id="lastName" @bind="@Employee.LastName"
   placeholder="Enter last name" />
```

Two Way Binding(Con.)

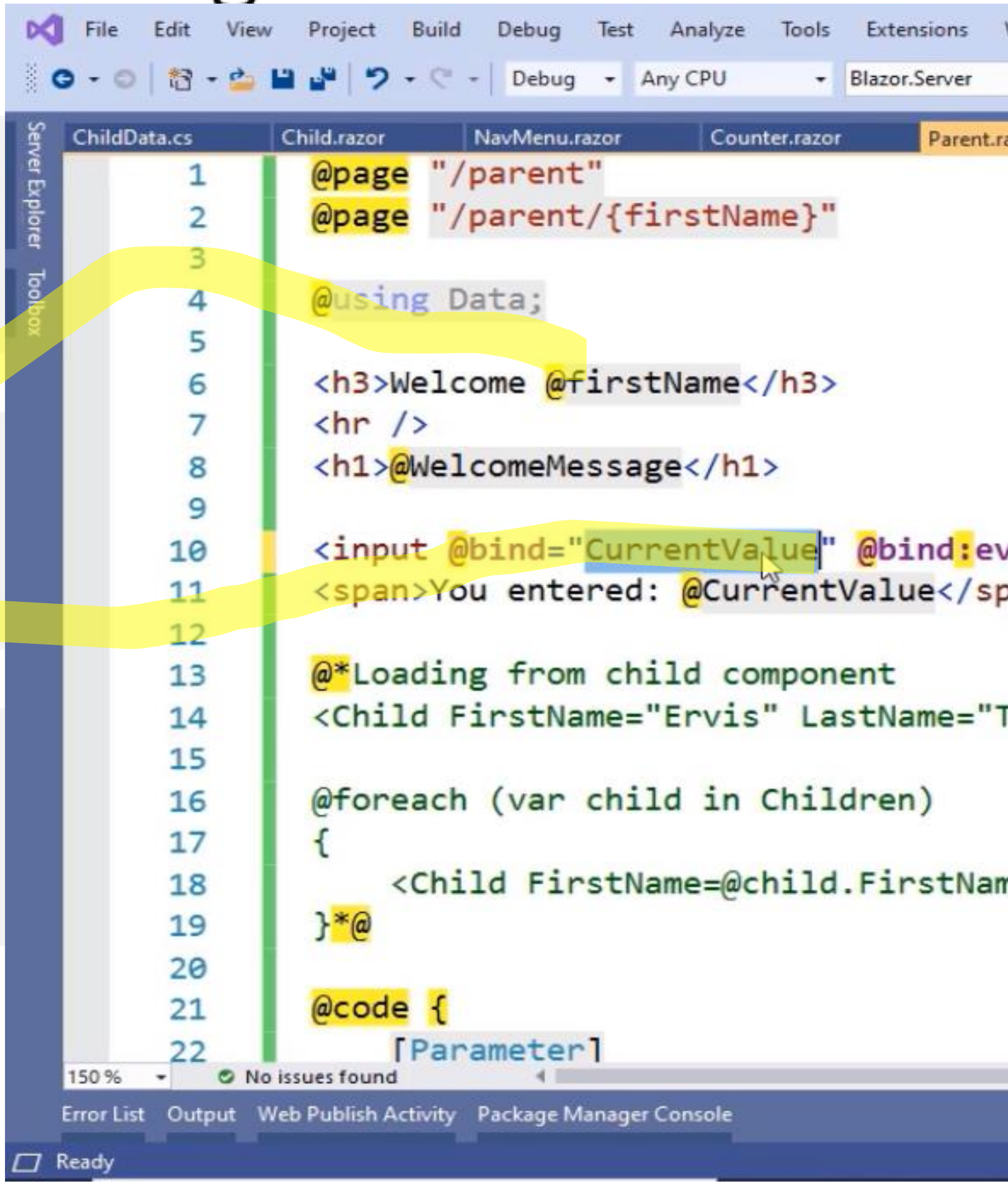
- To change the default behavior of binding to Different event



```
<input id="lastName" @bind-value="Employee.LastName"
   @bind-value:event="oninput"
   placeholder="Enter last name" />
```

Blazor Data Binding

- One-way data binding
- Two-way data binding
- Event binding



```
@page "/parent"
@page "/parent/{firstName}"

@using Data;

<h3>Welcome @firstName</h3>
<hr />
<h1>@WelcomeMessage</h1>

<input @bind="CurrentValue" @bind:ev
<span>You entered: @CurrentValue</sp

@*Loading from child component
<Child FirstName="Ervis" LastName="T

@foreach (var child in Children)
{
    <Child FirstName=@child.FirstNam
}*@

@code {
    [Parameter]
```