FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

LẬP TRÌNH C# 6

BÀI 2: BLAZOR SERVER

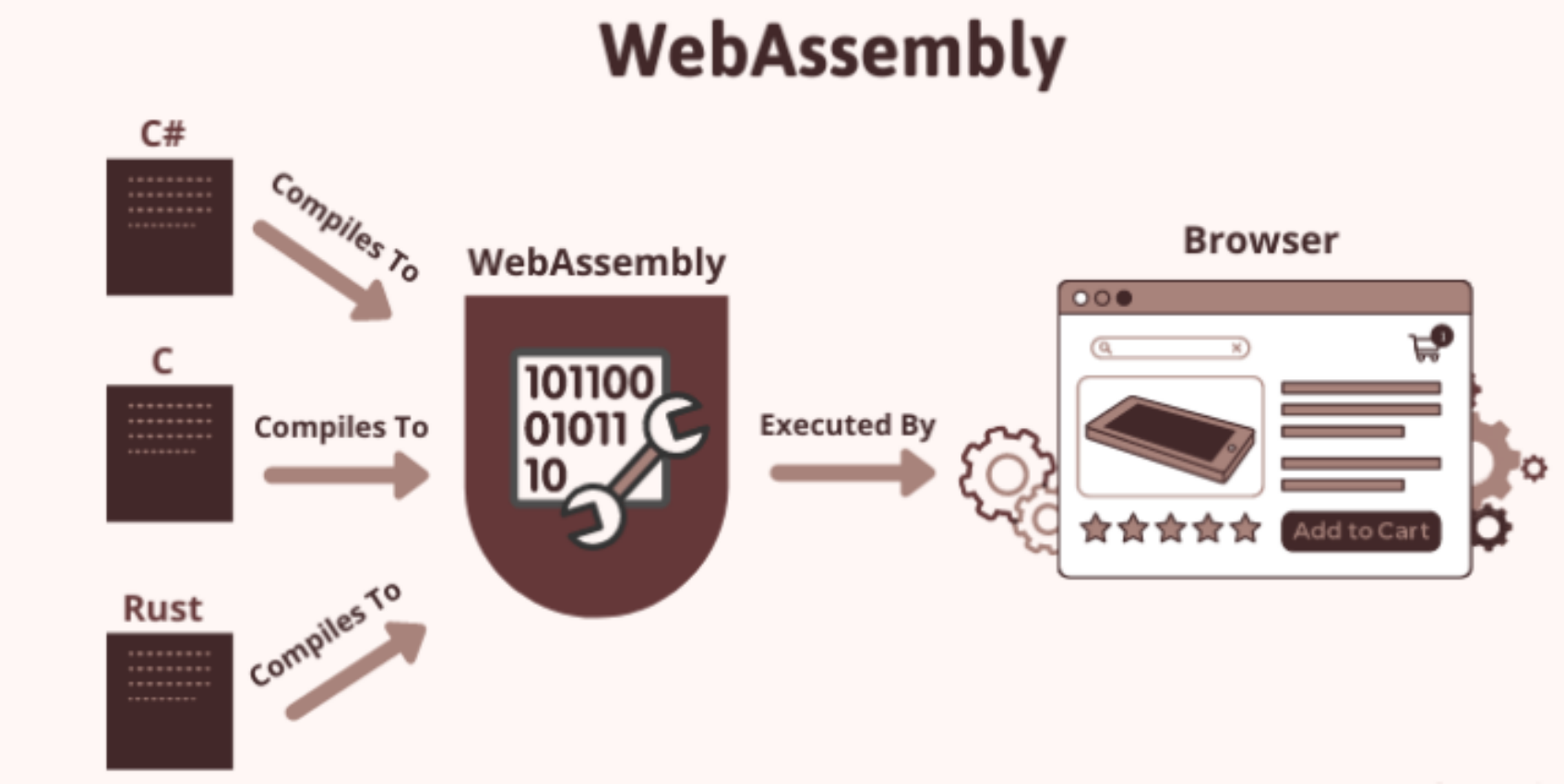
THỰC HỌC – THỰC NGHIỆP

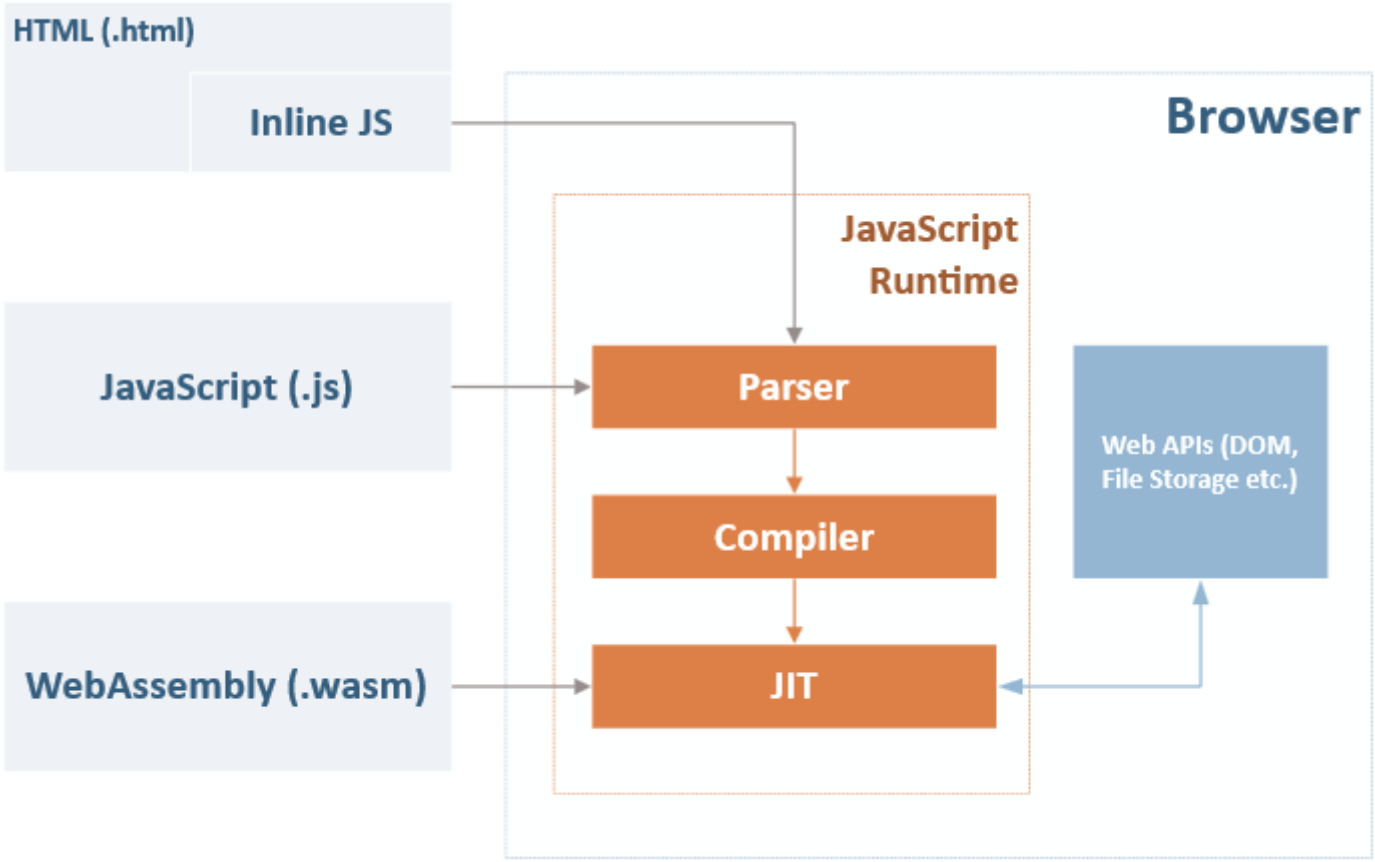
http://www.poly.edu.vn

Mục tiêu

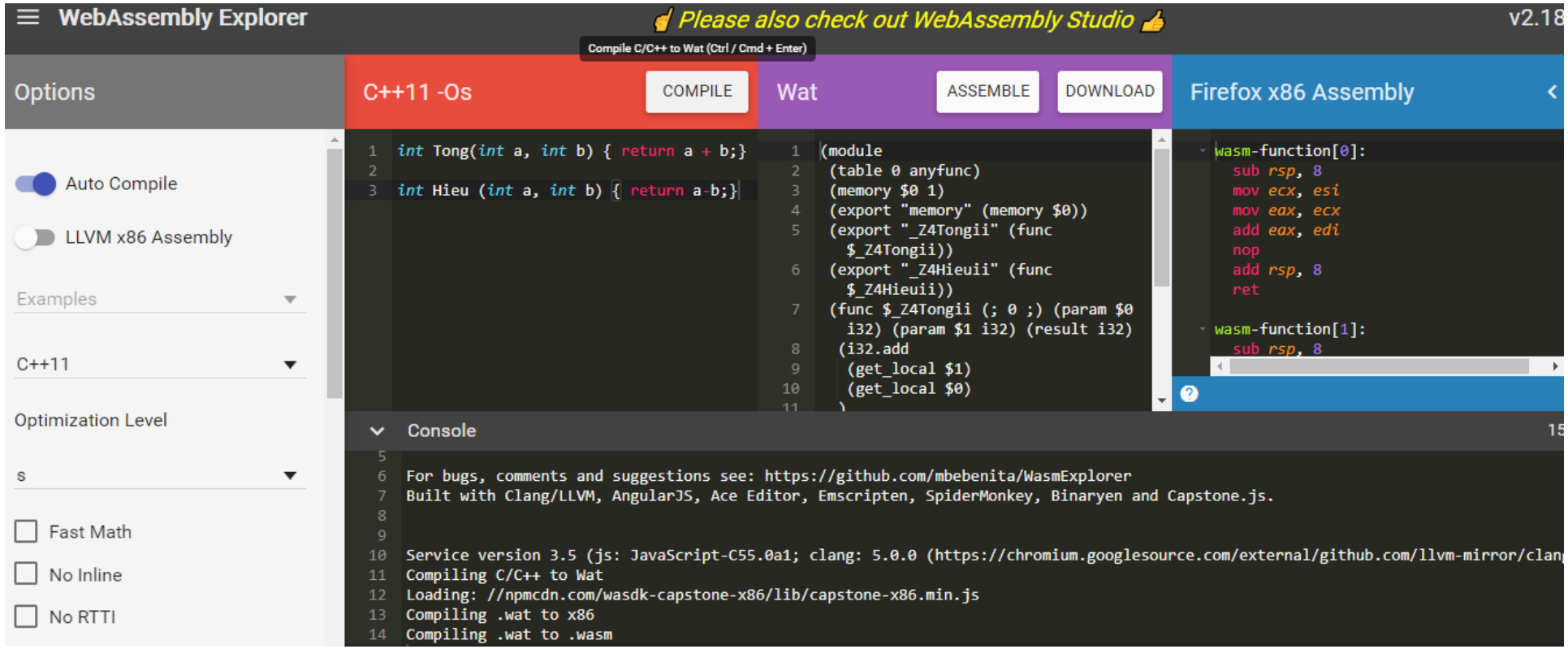
- WebAssembly
- WebAssembly và C#
- Blazor server

- Cùng với HTML, CSS và JavaScript, WebAssembly là ngôn ngữ thứ tư cho phép các đoạn mã thực thi trên các trình duyệt (browser).
- Định dạng nhị phân mức thấp cho web, dịch ra từ ngôn ngữ low-level khác như C/C++, C# và Rust
- Kết hợp bổ sung cho javascript
- Các trình duyệt phổ biến như Chrome, Firefox, Safari, Opera, Edge đều hỗ trợ WebAssembly.

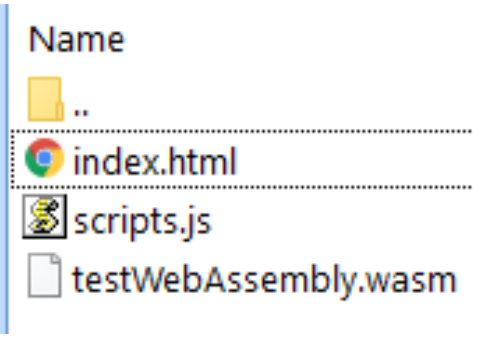

WebAssembly
C#
C
Rust
Compiles To
Compiles To
Compiles To
WebAssembly
101100
01011
10
Executed By
Browser
Add to Cart


HTML (.html)
Inline JS
Browser
JavaScript
Runtime
JavaScript (.js)
Parser
Compiler
JIT
Web APIs (DOM,
File Storage etc.)
WebAssembly (.wasm)

- https://mbebenita.github.io/WasmExplorer/



- Tích hợp webAssemly vào website

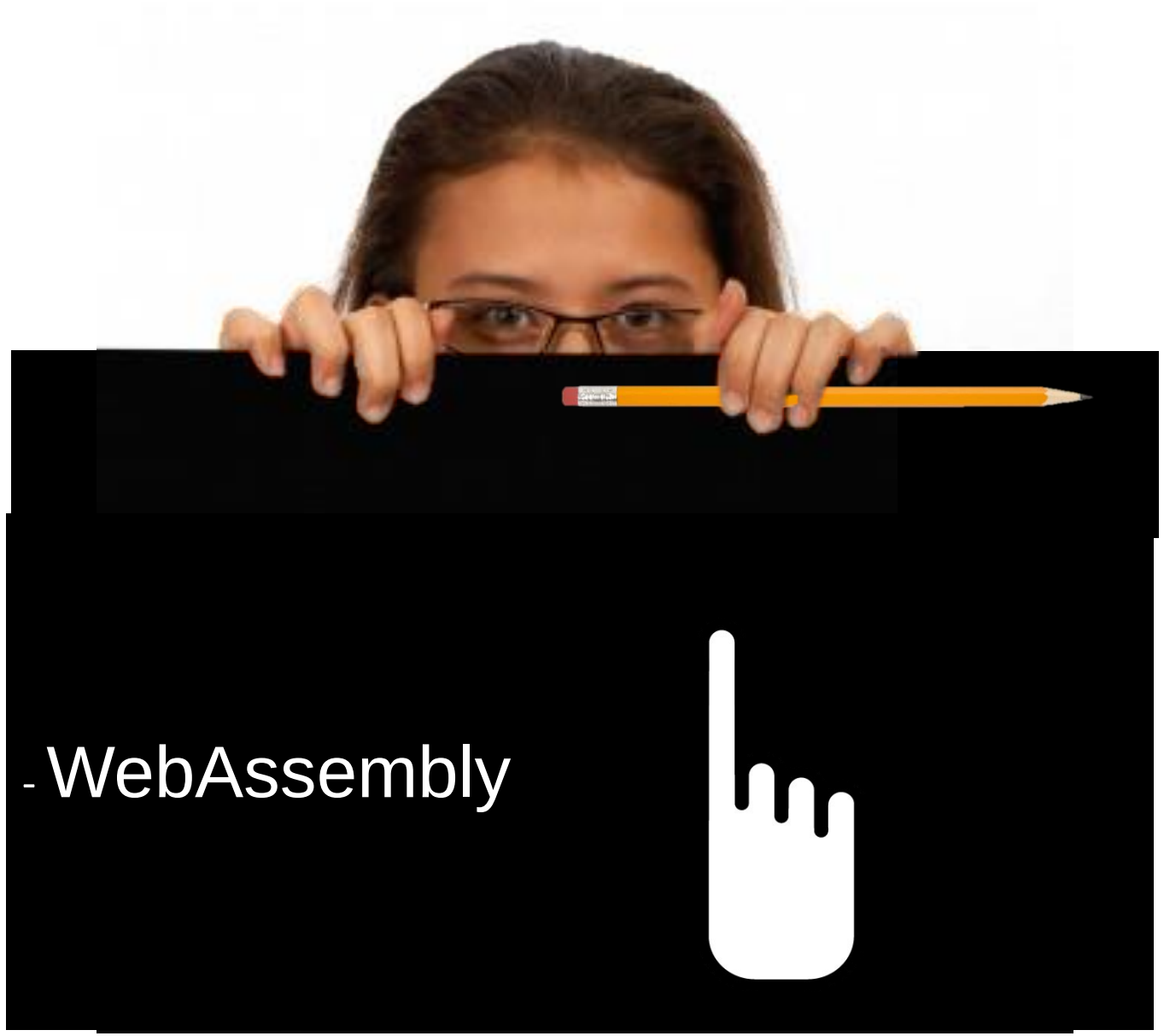


scripts

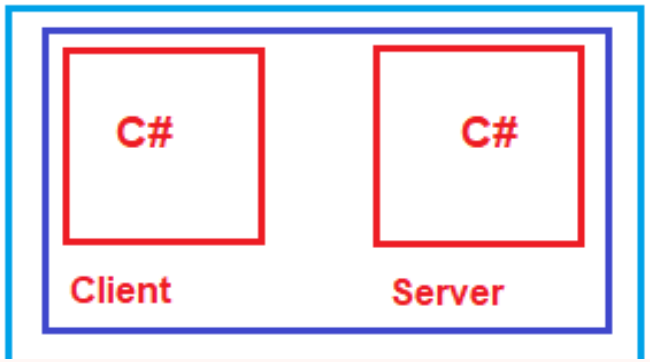
```
let tong, hieu;
function loadWebAssembly(fileName) {
    return fetch(fileName)
        .then(response => response.arrayBuffer())
        .then(buffer => WebAssembly.compile(buffer))
        .then(module => {return new WebAssembly.Instance(module) });
};

loadWebAssembly('http://localhost:80/testWebAssembly.wasm')
    .then(instance => {
    tong = instance.exports._Z4Tongii;
    hieu = instance.exports._Z4Hieuii;
    document.getElementById("tong").innerHTML = "Tong 2 va 5 la: " + tong(2,5);
    document.getElementById("hieu").innerHTML = "Hieu 2 va 5 la: " + hieu(2,5);
});
```

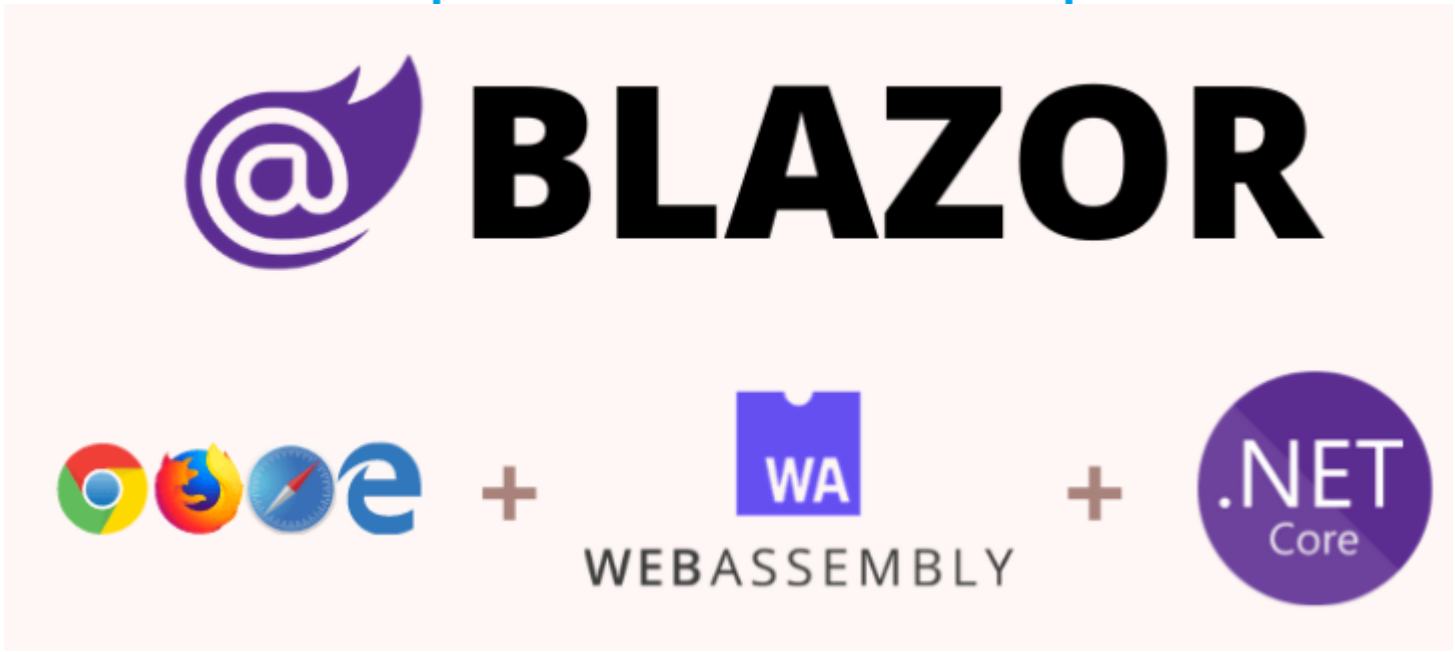

```
<html>
  <head>
      <script src="scripts.js"></script>
  </head>
  <body>
      <h1>Hi Thepv,Demo WebAssembly! </h1>
      <h3 id="tong"></h3>
      <h3 id="hieu"></h3>
  </body>
</html>
```

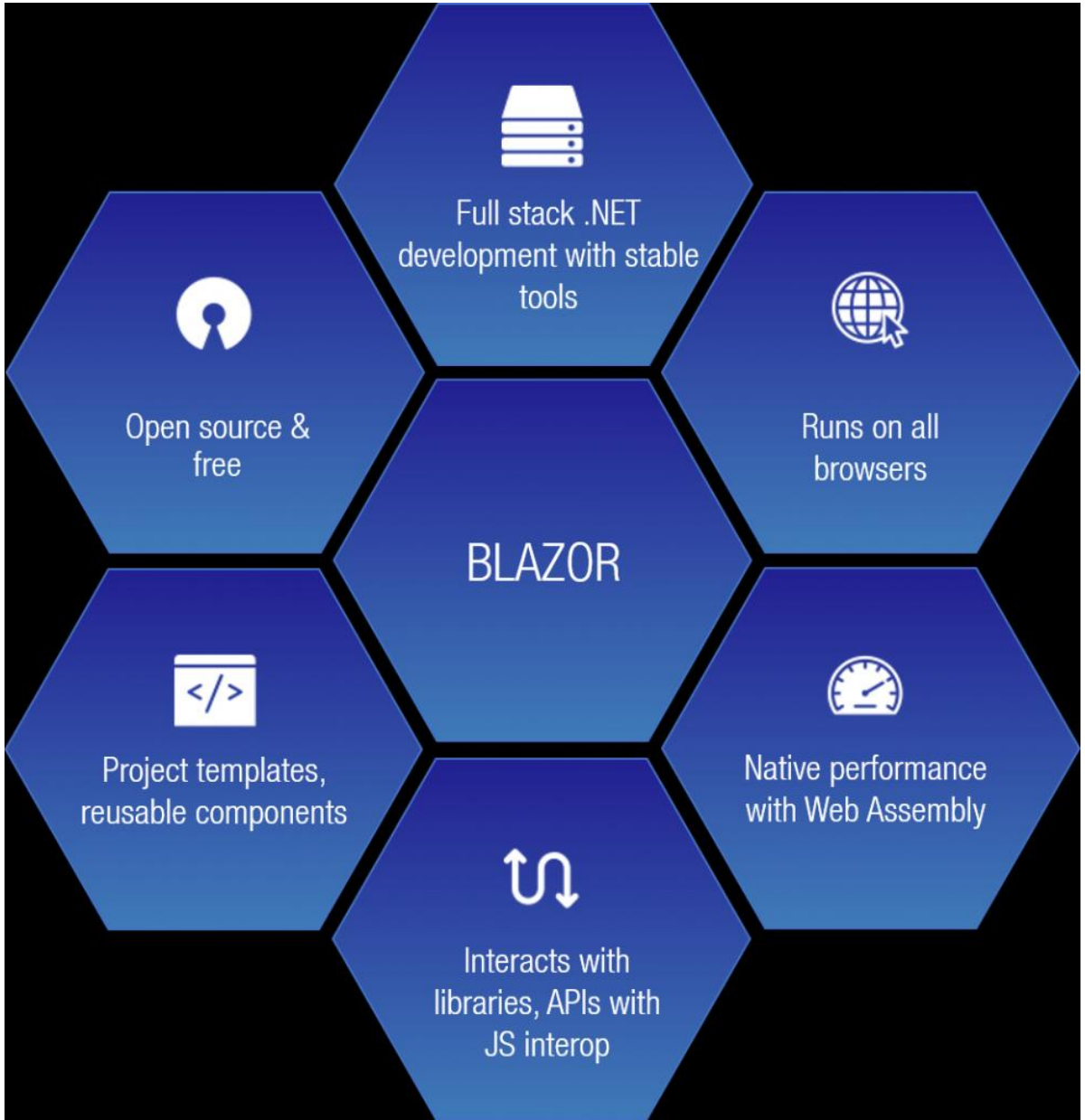
DEMO 2.1



- C# dịch sang WebAssembly và chạy trên trình duyệt như một ứng dụng native: Blazor

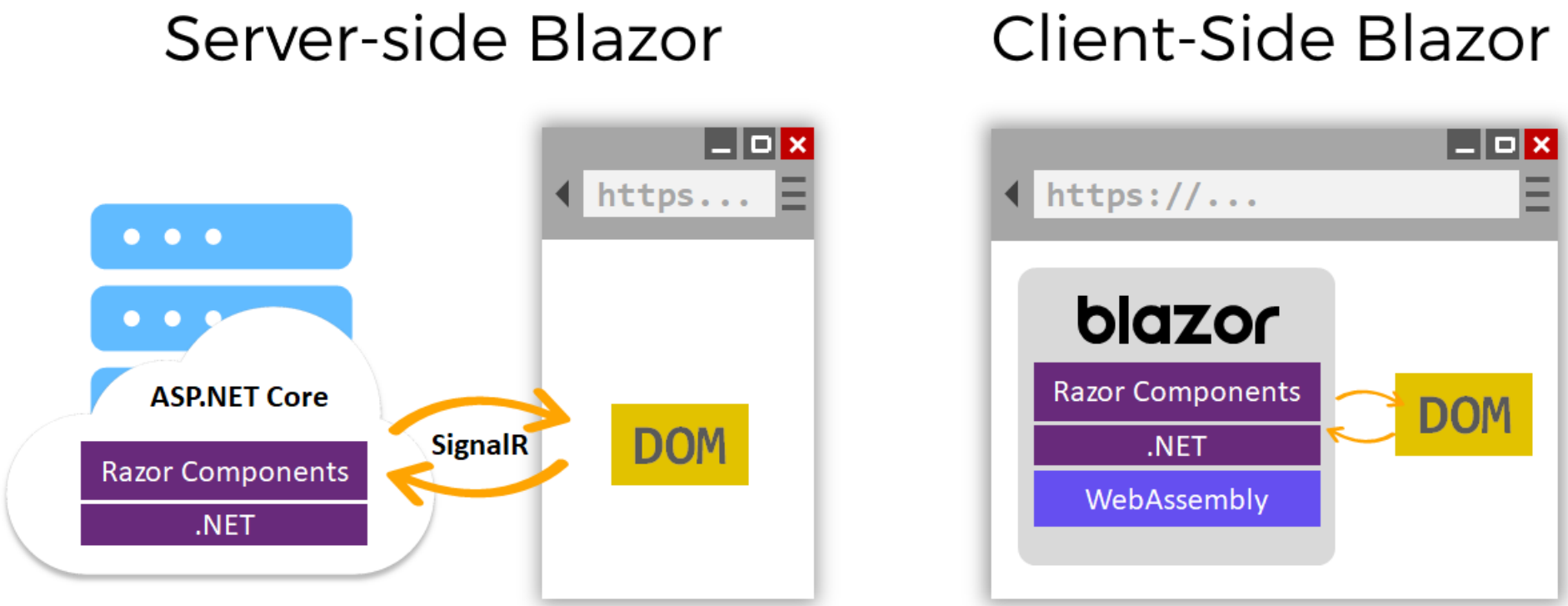




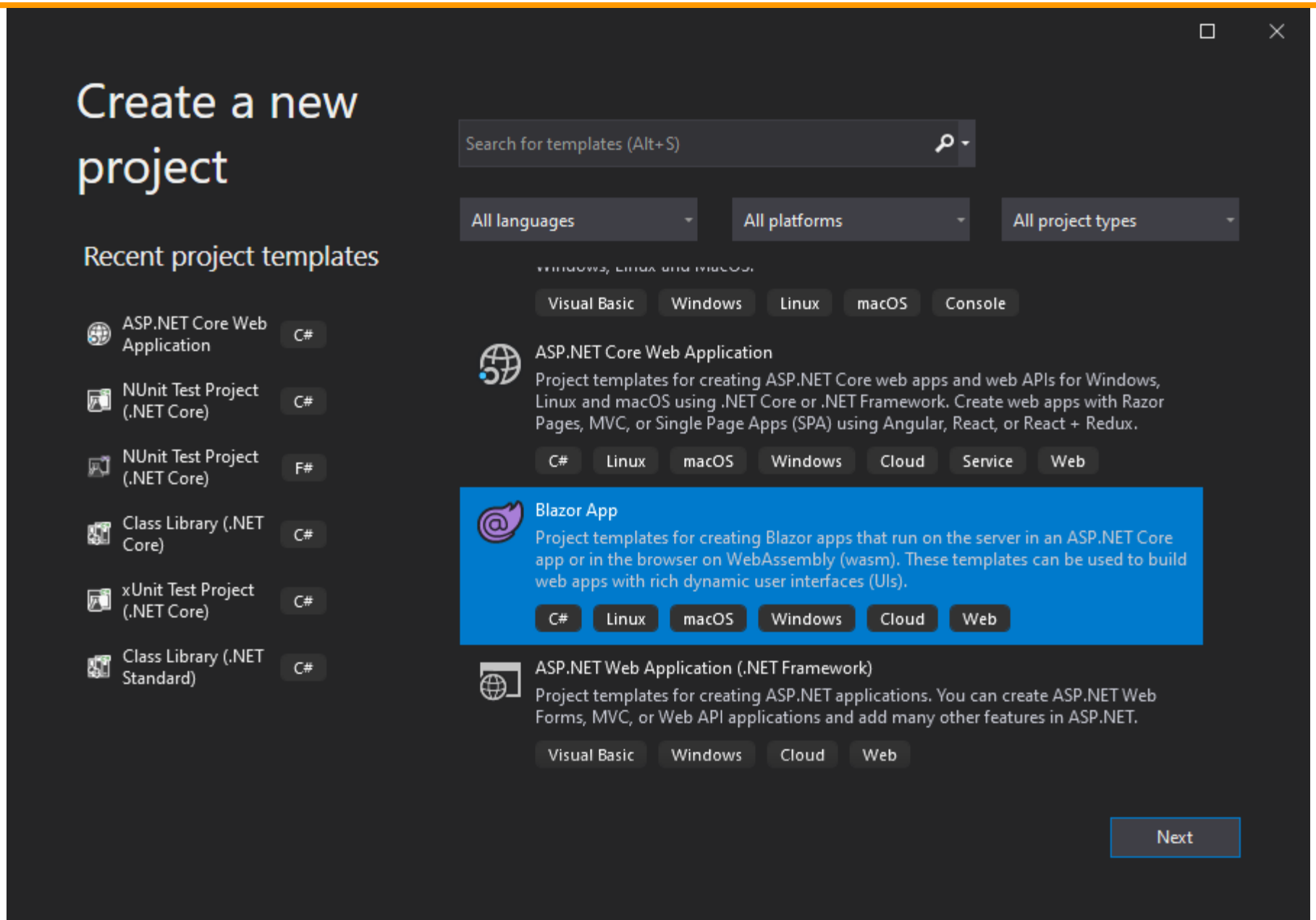

Full stack .NET development with stable tools
Open source & free
Runs on all browsers
BLAZOR
Project templates, reusable components
Native performance with Web Assembly
Interacts with libraries, APIs with JS interop

Blazor Hosting Models

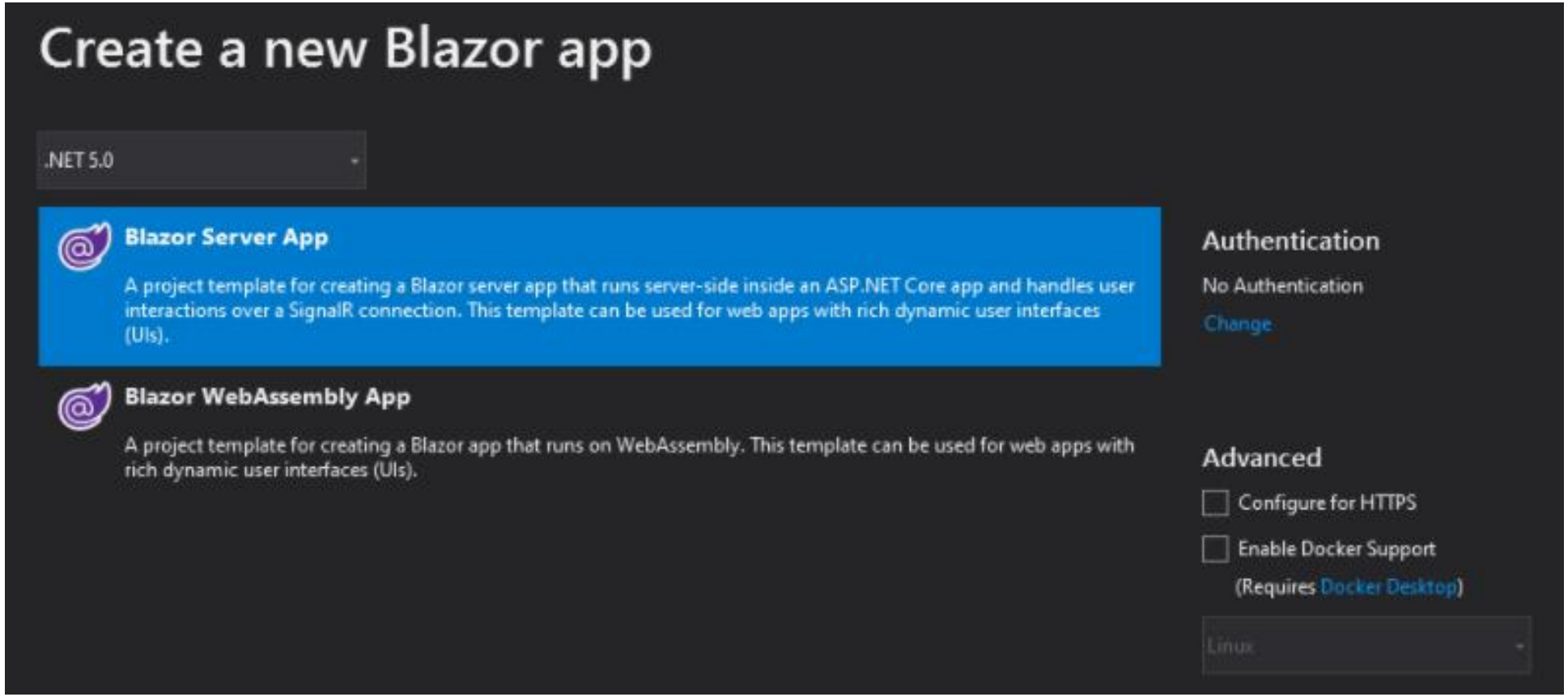
1. Blazor Server
2. Blazor WebAssembly
3. Blazor Electron
4. Mobile Blazor Bindings



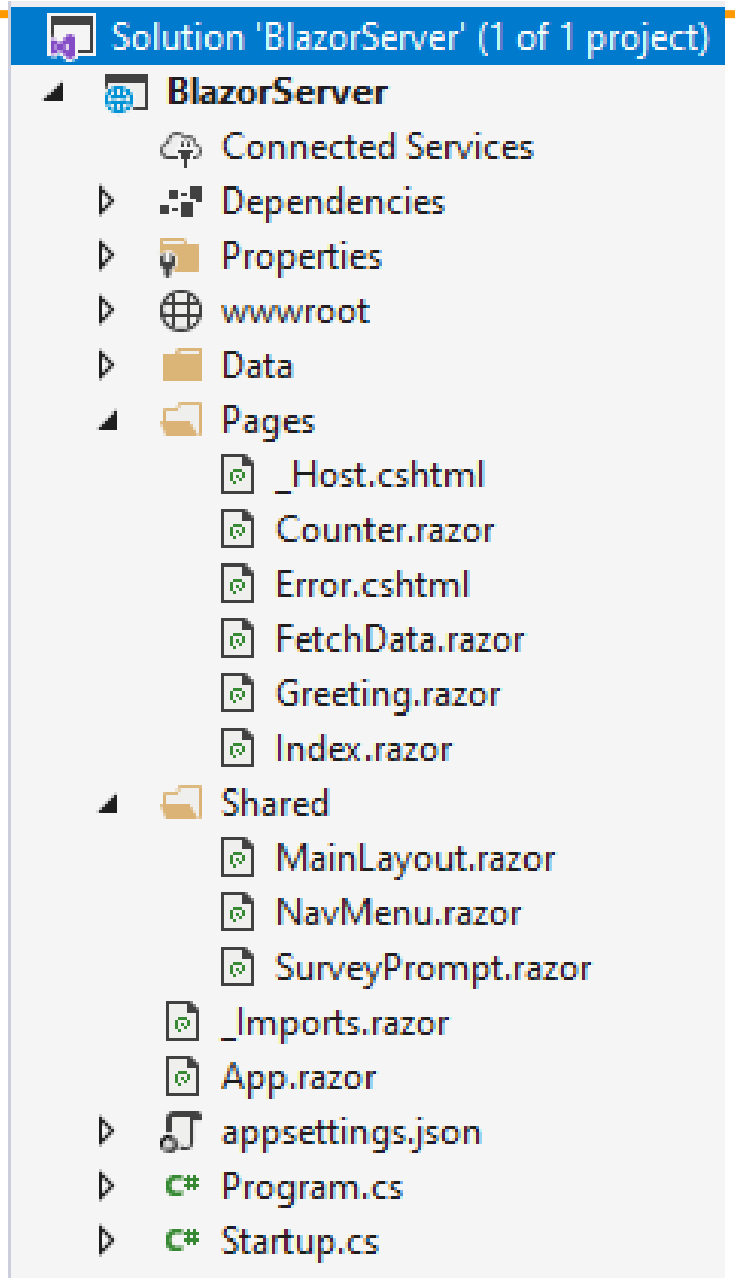
BLAZOR SERVER



BLAZOR SERVER



Blazor Server Project Structure



BLAZOR SERVER PROJECT STRUCTURE

- Program.cs

```
public static void Main(string[] args)
{
    CreateHostBuilder(args).Build().Run();
}

public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();
        });
```

BLAZOR SERVER PROJECT STRUCTURE

- Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
    services.AddServerSideBlazor();
    services.AddSingleton<WeatherForecastService>();
}
```

```
1  app.UseEndpoints(endpoints =>
2  {
3      endpoints.MapBlazorHub();
4      endpoints.MapFallbackToPage("/_Host");
5  });
```

BLAZOR SERVER PROJECT STRUCTURE

_Host.cshtml

```
<component type="typeof(App)" render-mode="ServerPrerendered" />
```

```
<script src="_framework/blazor.server.js"></script>
```

App.razor

```
<Router AppAssembly="@typeof(Program).Assembly" PreferExactMatches="@true">
    <Found Context="routeData">
        <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
    </Found>
    <NotFound>
        <LayoutView Layout="@typeof(MainLayout)">
            <p>Sorry, there's nothing at this address.</p>
        </LayoutView>
    </NotFound>
</Router>
```

BLAZOR SERVER PROJECT STRUCTURE

- MainLayout.cshtml

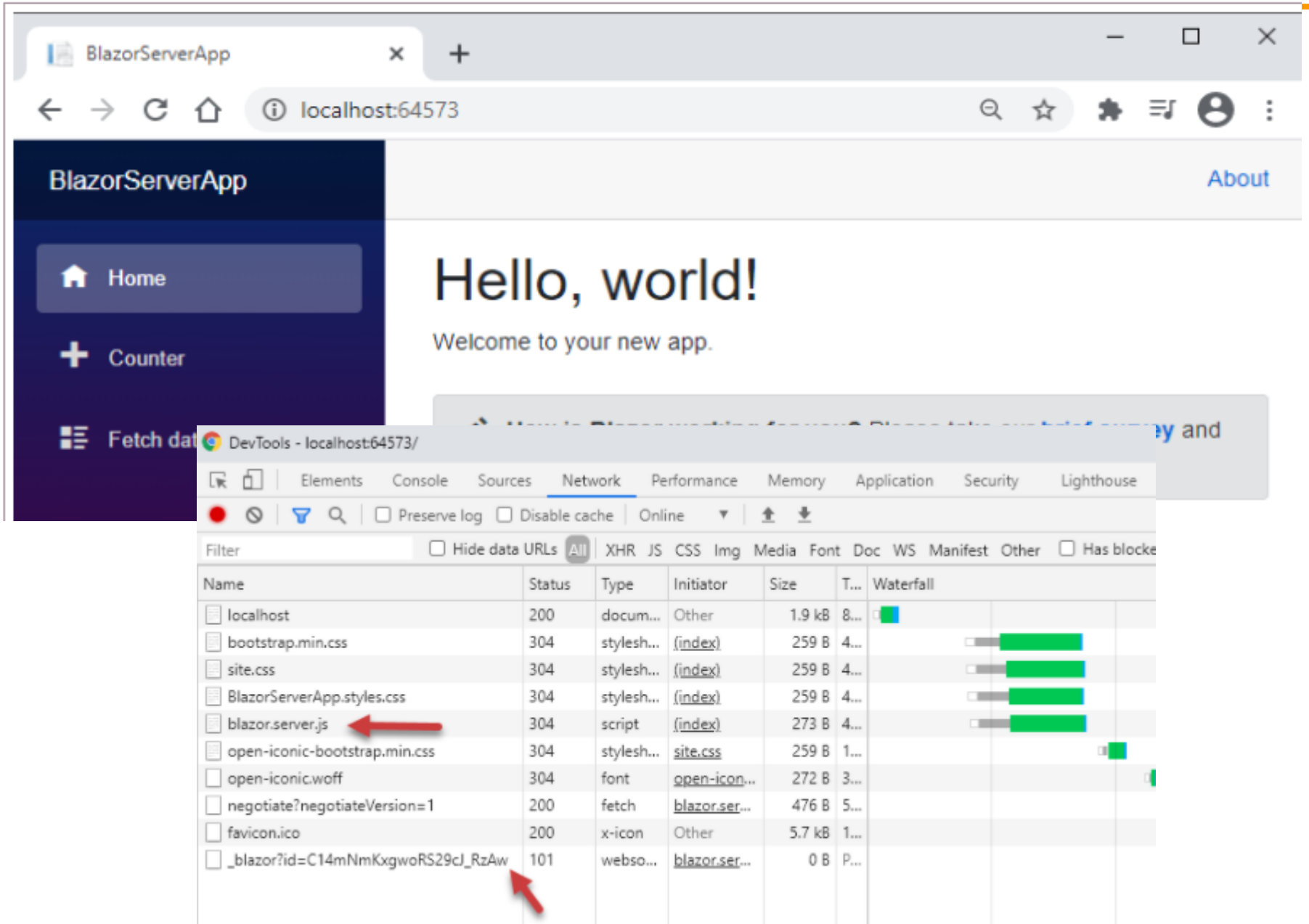
```
@inherits LayoutComponentBase

<div class="page">
    <div class="sidebar">
        <NavMenu />
    </div>

    <div class="main">
        <div class="top-row px-4">
            <a href="https://docs.microsoft.com/aspnet/" target="_blank">About</a>
        </div>

        <div class="content px-4">
            @Body
        </div>
    </div>
</div>
```

Blazor Server Project Structure

- _Imports.razor

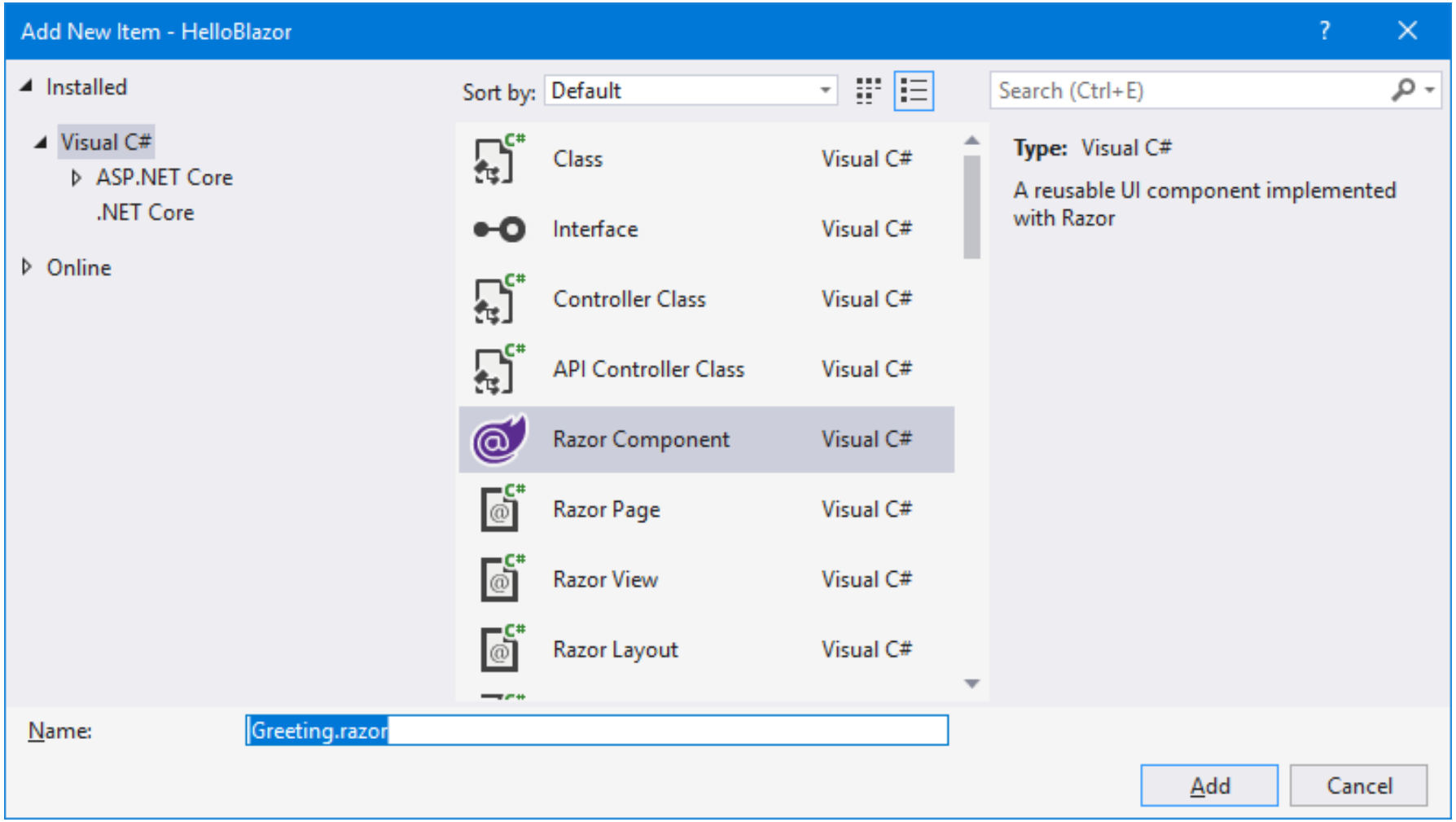
```
@using System.Net.Http
@using Microsoft.AspNetCore.Authorization
@using Microsoft.AspNetCore.Components.Authorization
@using Microsoft.AspNetCore.Components.Forms
@using Microsoft.AspNetCore.Components.Routing
@using Microsoft.AspNetCore.Components.Web
@using Microsoft.AspNetCore.Components.Web.Virtualization
@using Microsoft.JSInterop
```

RUN THE BLAZOR SERVER APP



RAZOR COMPONENT

- Tạo mới page trong blazor



RAZOR COMPONENT

- Tạo mới page trong blazor

Greeting.razor | Counter.razor | NavMenu.razor | BlazorServer

```
@page "/greeting"
<h3> Hello Blazor server demo</h3>

@code {
    private string title = "Thepv, hi Blazor server";
    private string name;
    private string address;
    private string greeting;

    private void welcome()
    {
        if (!string.IsNullOrEmpty(name) && !string.IsNullOrEmpty(address))
        {
            greeting = $"Welcome {name} from {address} go to blazor app ";
        }
    }
}
<div class="form-group">
    <input type="text" placeholder="your name" @bind="name" class="form-control" />
</div>
<div class="form-group">
    <input type="text" placeholder="your address" @bind="address" class="form-control" />
</div>
<div class="form-group">
    <button @onclick="welcome" class="btn btn-primary">Welcome</button>
</div>
<h3>@greeting</h3>
```

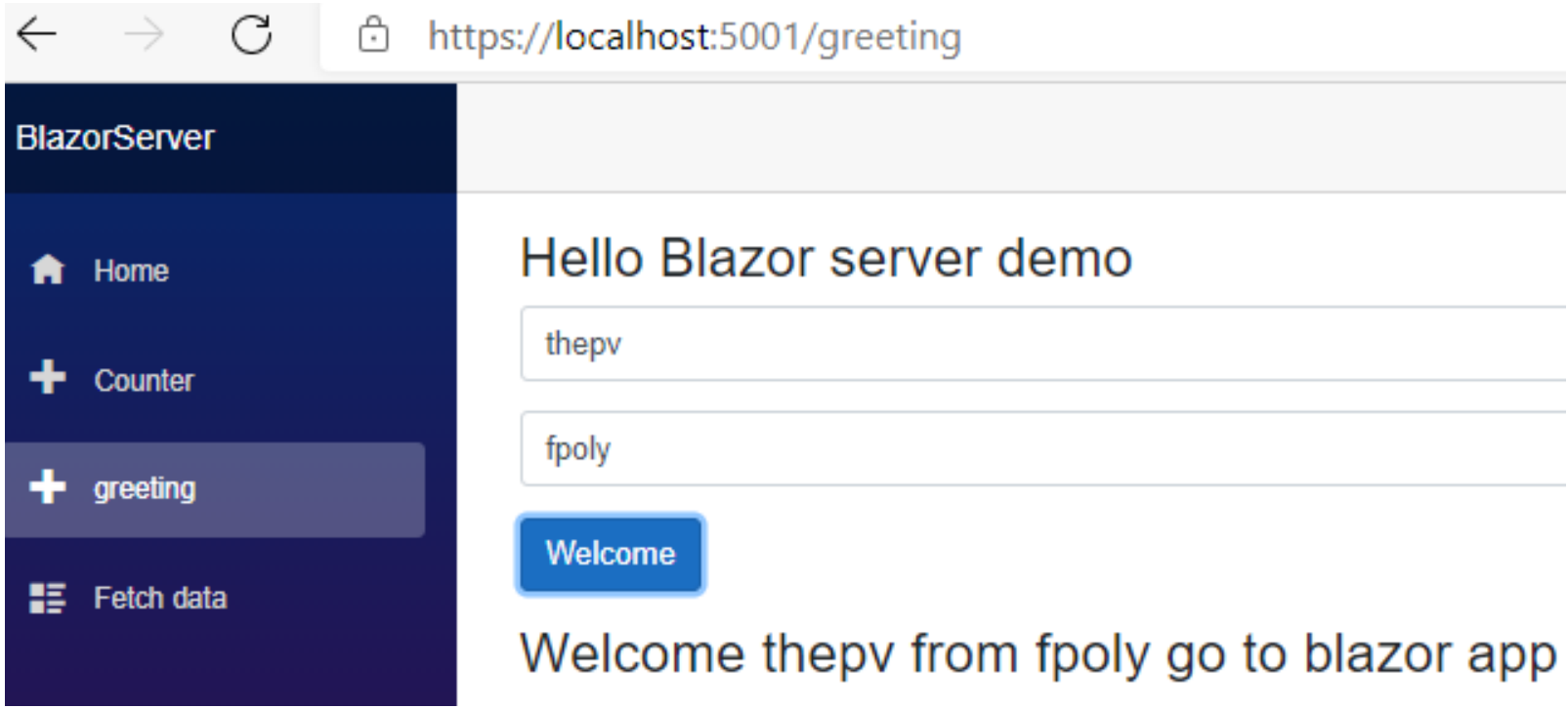
RAZOR COMPONENT

- Tạo mới page trong blazor

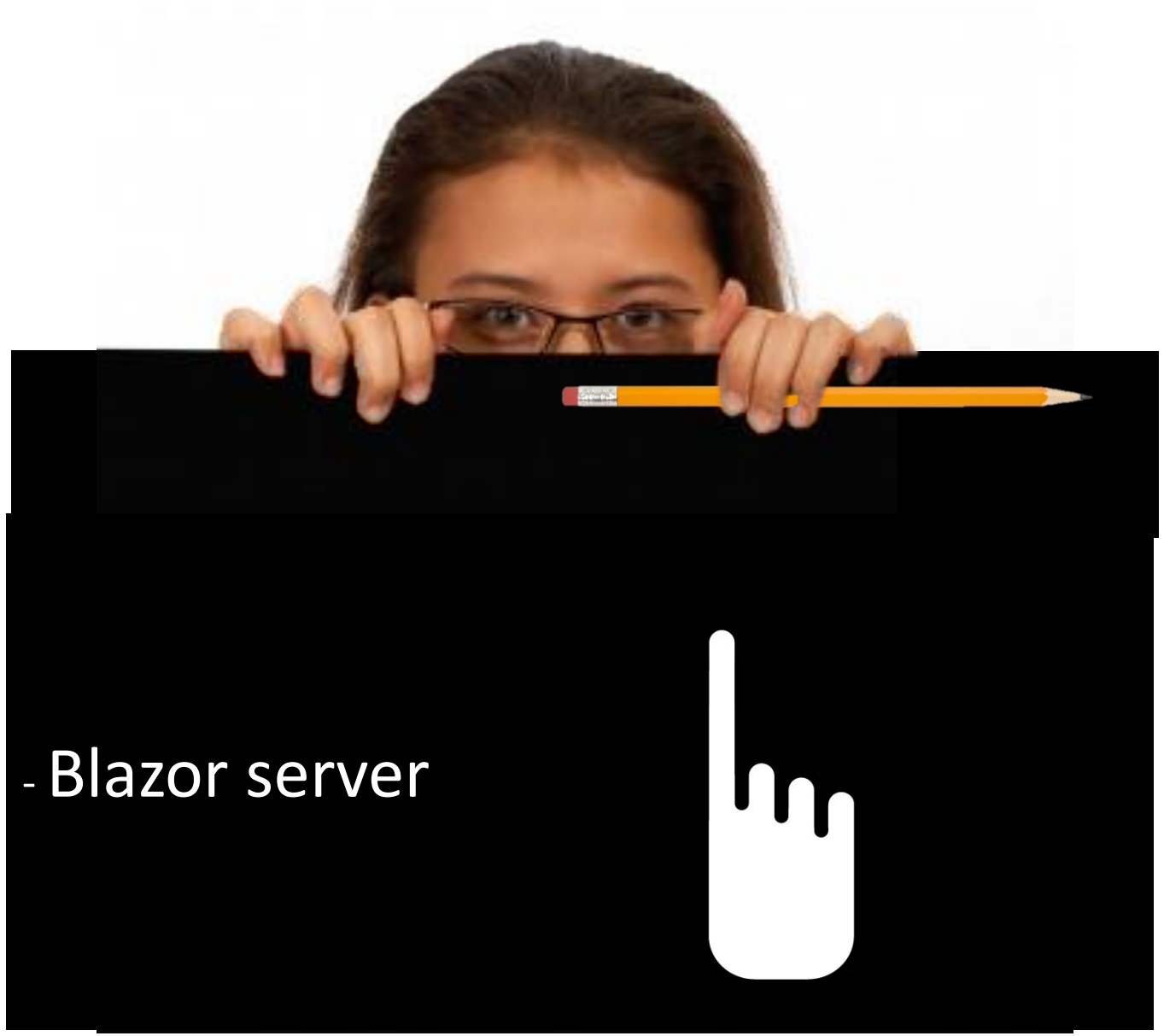
NavMenu.razor | Greeting.razor | Counter.razor | BlazorServer

```
</div>

<div class="@NavMenuCssClass" @onclick="ToggleNavMenu">
    <ul class="nav flex-column">
        <li class="nav-item px-3">
            <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
                <span class="oi oi-home" aria-hidden="true"></span> Home
            </NavLink>
        </li>
        <li class="nav-item px-3">
            <NavLink class="nav-link" href="counter">
                <span class="oi oi-plus" aria-hidden="true"></span> Counter
            </NavLink>
        </li>
        <li class="nav-item px-3">
            <NavLink class="nav-link" href="greeting">
                <span class="oi oi-plus" aria-hidden="true"></span> greeting
            </NavLink>
        </li>
        <li class="nav-item px-3">
            <NavLink class="nav-link" href="fetchdata">
                <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
            </NavLink>
        </li>
    </ul>
</div>
```

- Tạo mới page trong blazor



DEMO 2.2



FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

LẬP TRÌNH C# 6

BÀI 2: BLAZOR SERVER

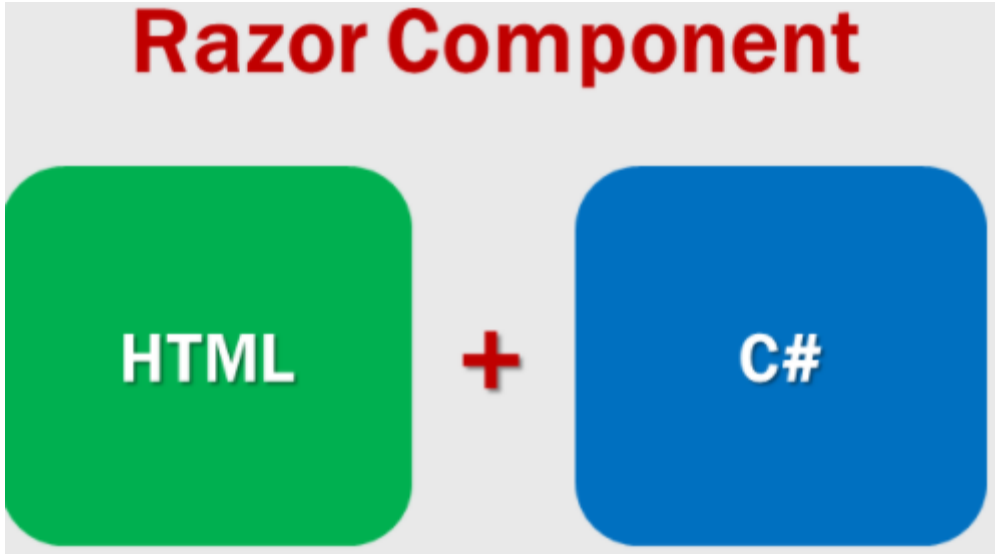
P2.3

THỰC HỌC – THỰC NGHIỆP

http://www.poly.edu.vn

Blazor Components

- Blazor is a component driven framework
- Components can be nested, reused, and shared across multiple projects
- Component files have the extension .razor
- Razor components or Blazor components
- Component name must start with an uppercase character



BLAZOR COMPONENTS

Directive

```
@page "/fetchdata"
@using WebApplication33.Data
@inject WeatherForecastService ForecastService
```

Markup / Razor

```
<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@if (forecasts == null)
{
    <p><em>Loading...</em></p>
}
else
{
    <table class="table">...</table>
}
```

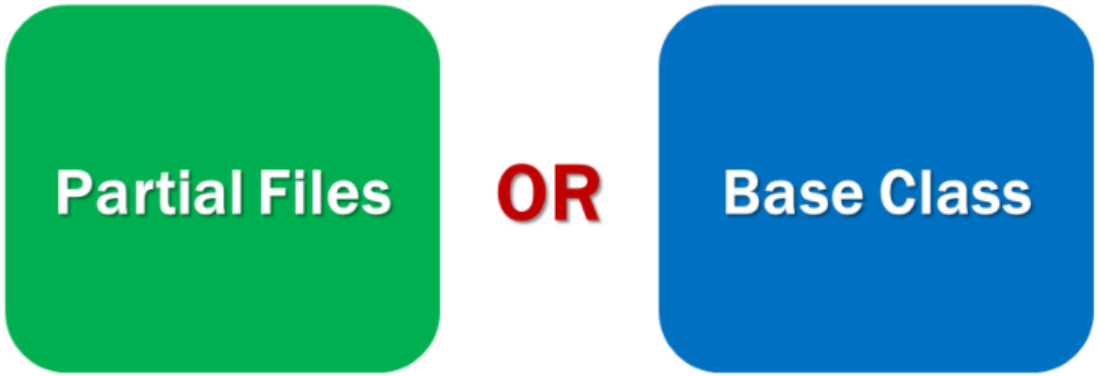
Component Logic

```
@code {
    WeatherForecast[] forecasts;

    protected override async Task OnInitAsync()
    {
        forecasts = await ForecastService.GetForecastAsync(DateTime.Now);
    }
}
```

FPT Education
FPT POLYTECHNIC

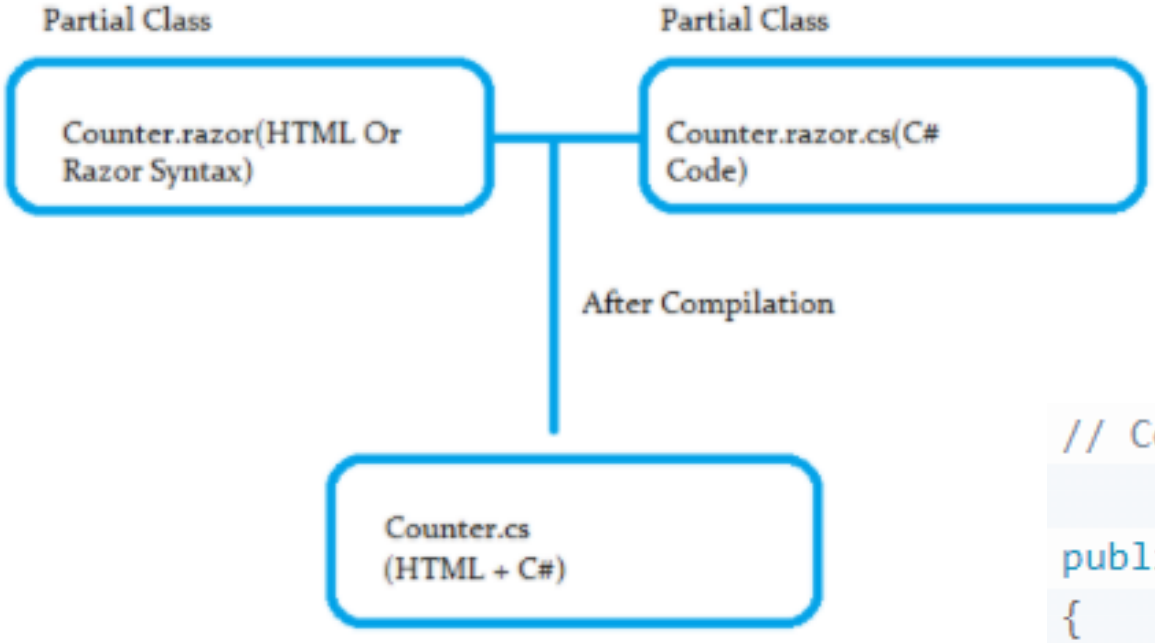
Splitting Razor Components in Blazor Application

- reuse parts of your code for multiple components
- readability and maintainability
- code-behind



SPLITTING RAZOR COMPONENTS IN BLAZOR APPLICATION

- Partial files approach




```
// Counter.razor.cs

public partial class Counter
{
    private int currentCount = 0;

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

SPLITTING RAZOR COMPONENTS IN BLAZOR APPLICATION

- Code-behind trong Blazor sử dụng kế thừa

FetchData.razor.cs

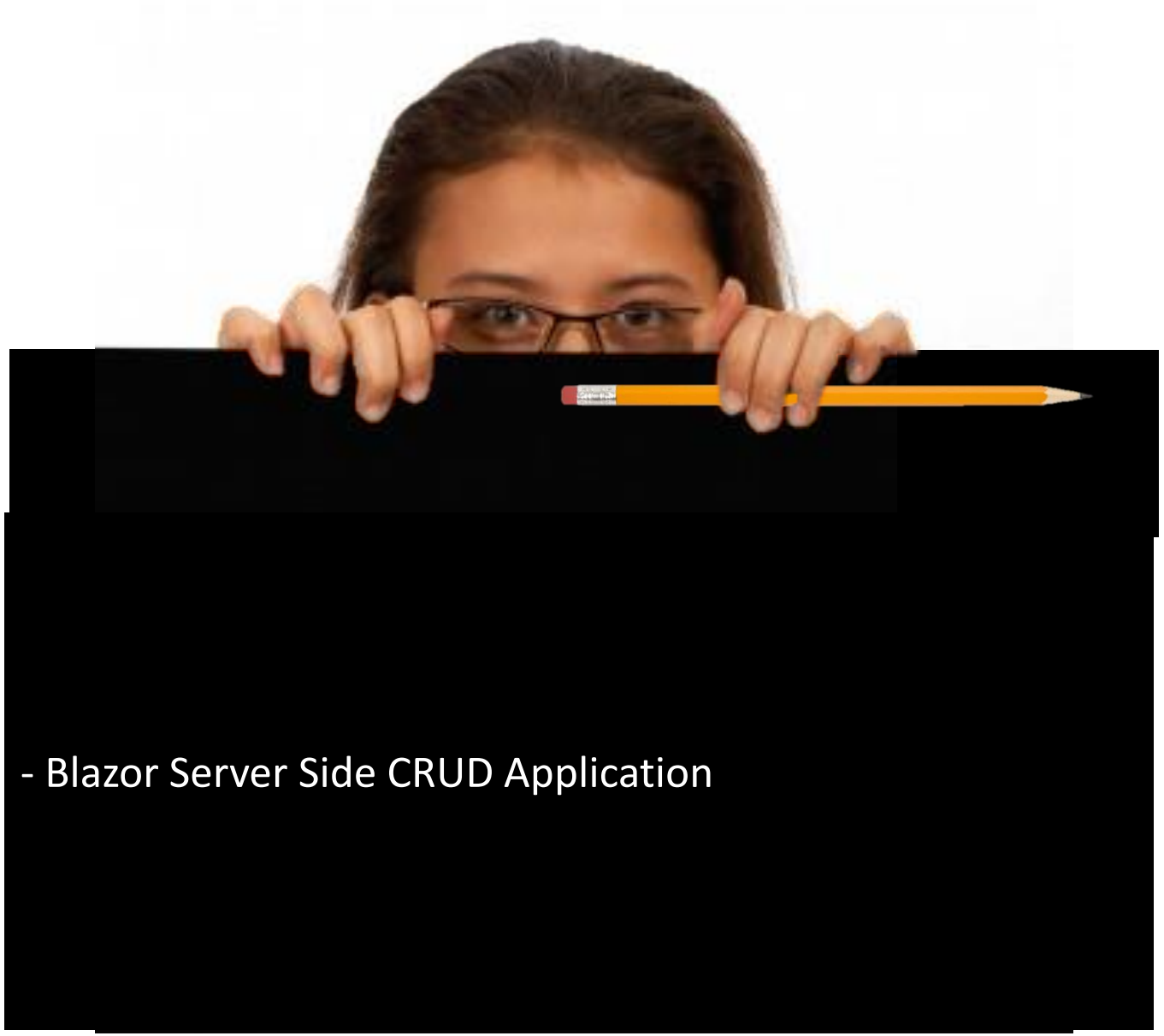
```
BlazorServerApp | BlazorServerApp.Pages.FetchDa | forecasts

using System;
using System.Threading.Tasks;

using BlazorServerApp.Data;

using Microsoft.AspNetCore.Components;

namespace BlazorServerApp.Pages {
    2 references
    public class FetchDataModel : ComponentBase {
        protected WeatherForecast[] forecasts;

        1 reference
        [Inject] WeatherForecastService ForecastService { get;
          set; }

        0 references
        protected override async Task OnInitializedAsync() {
            forecasts = await ForecastService.GetForecastAsync
              (DateTime.Now);
        }
    }
}
```

FetchData.razor

```
@inherits FetchDataModel
@page "/fetchdata"

<h1>Weather forecast</h1>

<p>This component demonstrates fetching data from a service.</p>

@if (forecasts == null) {
    <p><em>Loading...</em></p>
}
else {
    <table class="table">
        <thead>
            <tr>
                <th>Date</th>
                <th>Temp. (C)</th>
                <th>Temp. (F)</th>
                <th>Summary</th>
            </tr>
        </thead>
        <tbody>
            @foreach (var forecast in forecasts) {
                <tr>
                    <td>@forecast.Date.ToShortDateString()</td>
                    <td>@forecast.TemperatureC</td>
                    <td>@forecast.TemperatureF</td>
                    <td>@forecast.Summary</td>
                </tr>
            }
        </tbody>
    </table>
}
```

DEMO 2.3

DEMO 2.4



- Blazor Server Side CRUD Application

Tổng kết bài học

- WebAssembly
- WebAssembly và C#
- Blazor server

KẾT THÚC