FPT Education

FPT POLYTECHNIC

LẬP TRÌNH C# 6

BÀI 3: BLAZOR WEBASSEMBLY

P3.1

Education

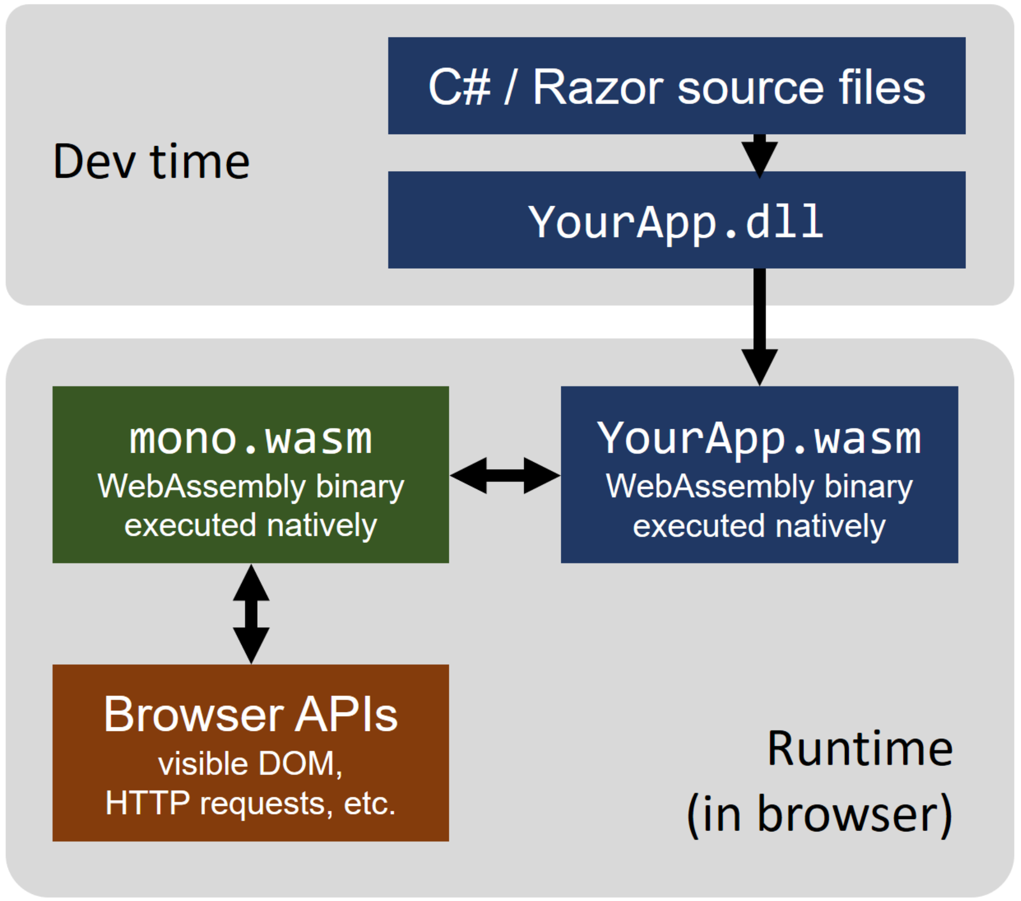
FPT POLYTECHNIC

MỤC TIÊU

- Blazor WebAssembly
- Blazor WebAssembly Gọi JavaScript
- JavaScript gọi C# method

BLAZOR WEBASSEMBLY

- ❑C# dịch sang WebAssembly và chạy trên trình duyệt như một ứng dụng native.
- ❑ Microsoft đưa .NET runtime lên WebAssembly gọi là Mono)
- ❑UI Framework giúp viết chương trình web client chạy trên trình duyệt
- ❑ Single-page applications (SPAs) dùng C# và .NET


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

BLAZOR WEBASSEMBLY

❑ Mô hình Blazor WebAssembly có những ưu điểm:

- Không phụ thuộc vào .NET server sau khi tải về client.
- Khai thác tài nguyên của client.
- Giảm tải cho server, đồng nghĩa với có thể phục vụ nhiều client hơn.
- Có thể triển khai mà không cần server (ví dụ, từ CDN – Content Delivery Network).

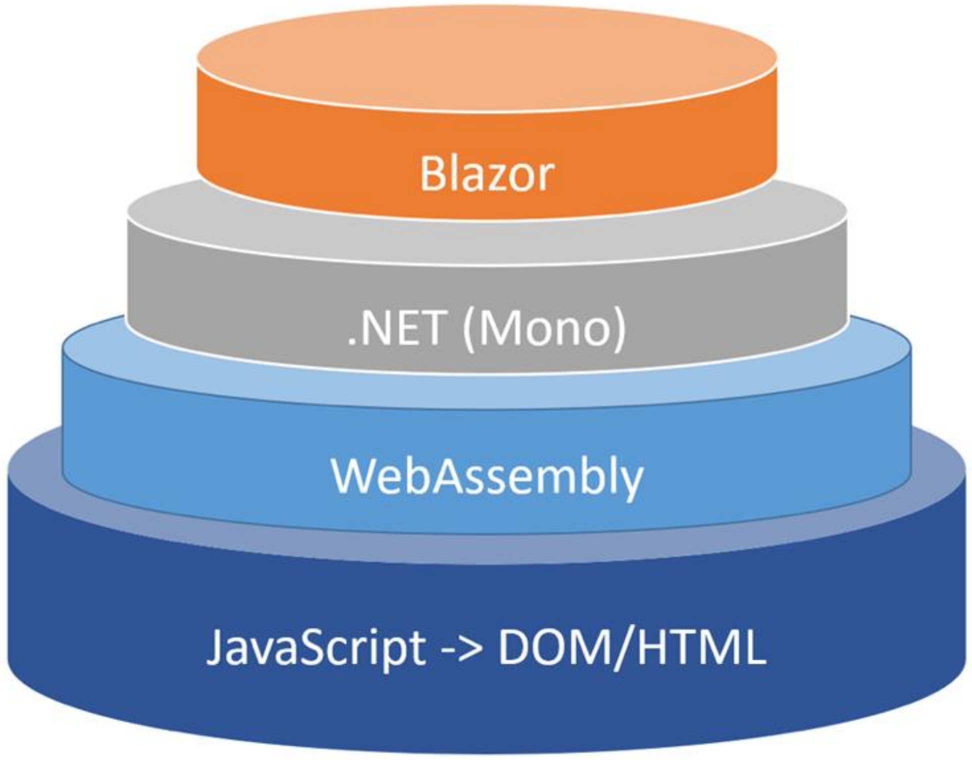
Blazor WebAssembly

❑ Nhược điểm của Blazor WebAssembly:

- Phụ thuộc vào khả năng của trình duyệt và hiệu suất của client.
- Có yêu cầu cao hơn đối với thiết bị client.
- Kích thước tải về lớn, tốc độ load (lần đầu) chậm hơn.

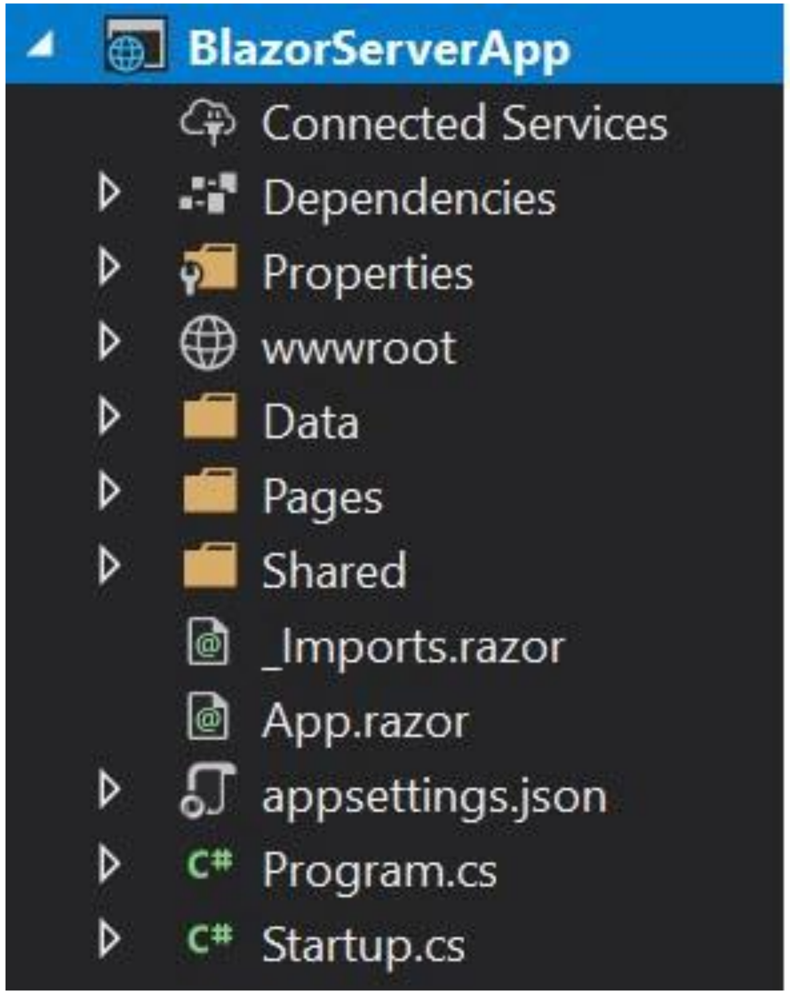
Full Stack Development

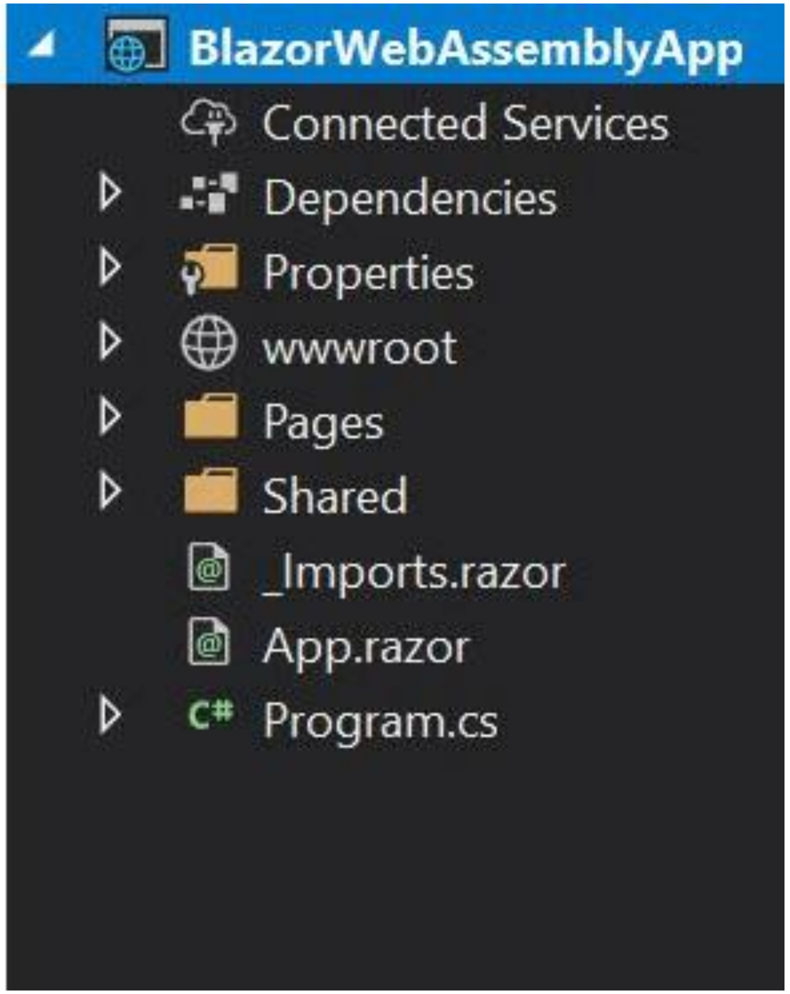
- ❑ Layouts
- ❑ IntelliSense and tooling
- ❑ JavaScript interoperability
- ❑ Debugging
- ❑ Dependency injection
- ❑ Routing services
- ❑ Forms and validation
- ❑ Server-side rendering
- ❑ Static file publishing
- ❑ Unit testing



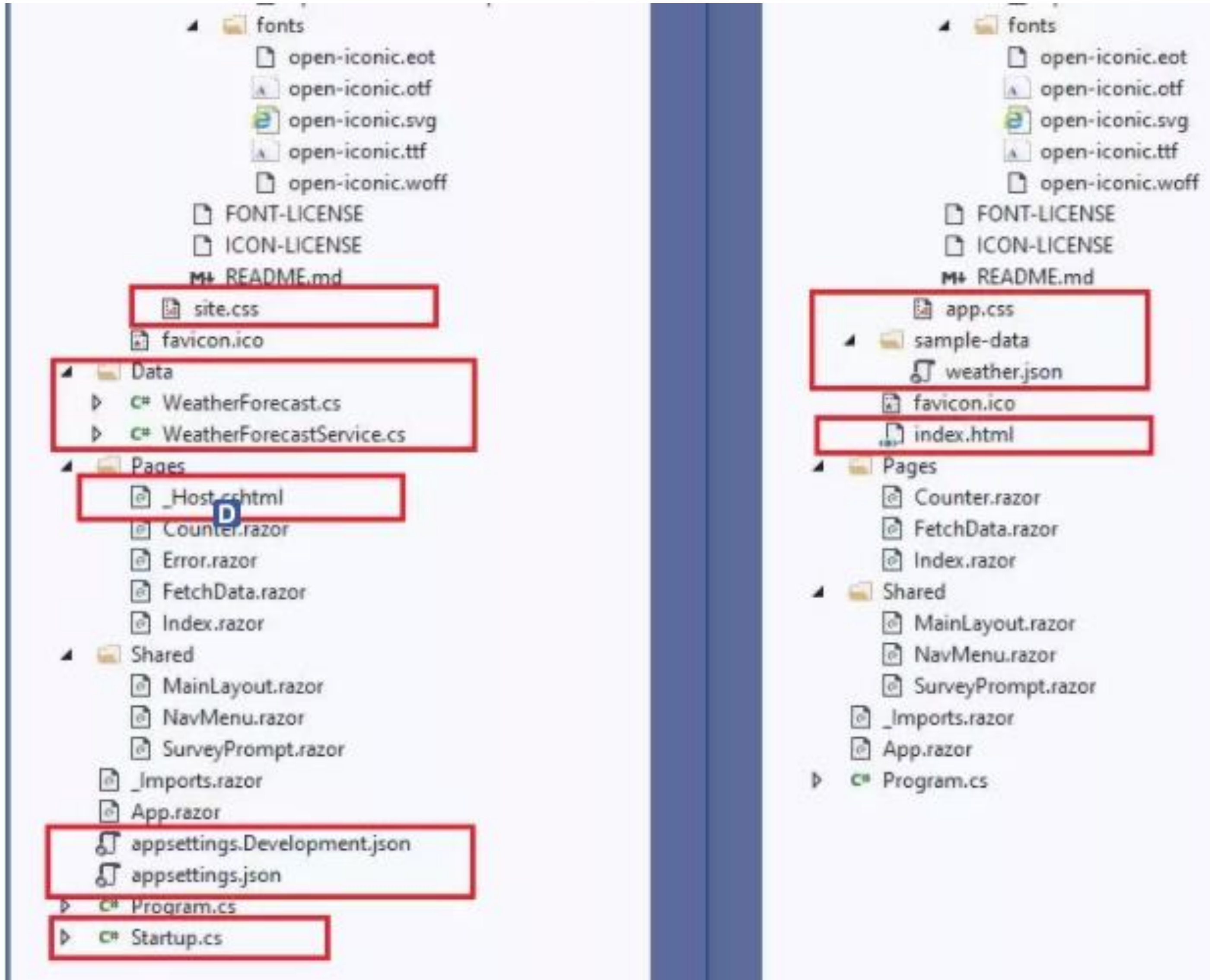
BLAZOR SERVER - WEBASSEMBLY

- Hosting Models (SignalR và WebAssembly)
- Project Structure Differences


BlazorServerApp
Connected Services
Dependencies
Properties
wwwroot
Data
Pages
Shared
_Imports.razor
App.razor
appsettings.json
Program.cs
Startup.cs


BlazorWebAssemblyApp
Connected Services
Dependencies
Properties
wwwroot
Pages
Shared
_Imports.razor
App.razor
Program.cs

❑ Project Structure Differences

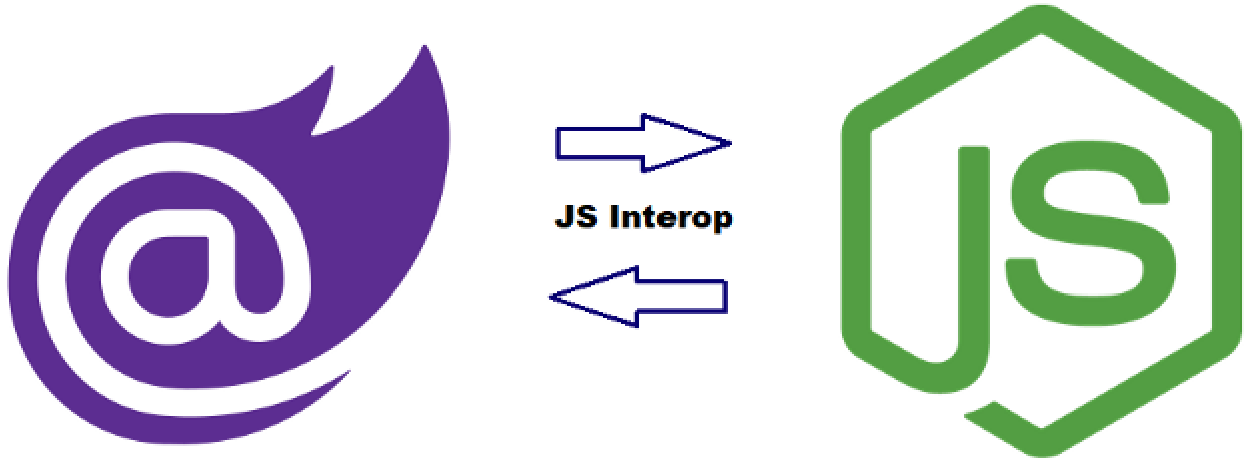


```
◢ fonts
    open-iconic.eot
    open-iconic.otf
    open-iconic.svg
    open-iconic.ttf
    open-iconic.woff
  FONT-LICENSE
  ICON-LICENSE
  README.md
 site.css
 favicon.ico
◢ Data
  ▷ WeatherForecast.cs
  ▷ WeatherForecastService.cs
◢ Pages
    _Host.cshtml
    Counter.razor
    Error.razor
    FetchData.razor
    Index.razor
◢ Shared
    MainLayout.razor
    NavMenu.razor
    SurveyPrompt.razor
  _Imports.razor
  App.razor
  appsettings.Development.json
  appsettings.json
▷ Program.cs
▷ Startup.cs
```

```
◢ fonts
    open-iconic.eot
    open-iconic.otf
    open-iconic.svg
    open-iconic.ttf
    open-iconic.woff
  FONT-LICENSE
  ICON-LICENSE
  README.md
 app.css
◢ sample-data
    weather.json
 favicon.ico
 index.html
◢ Pages
    Counter.razor
    FetchData.razor
    Index.razor
◢ Shared
    MainLayout.razor
    NavMenu.razor
    SurveyPrompt.razor
  _Imports.razor
  App.razor
▷ Program.cs
```

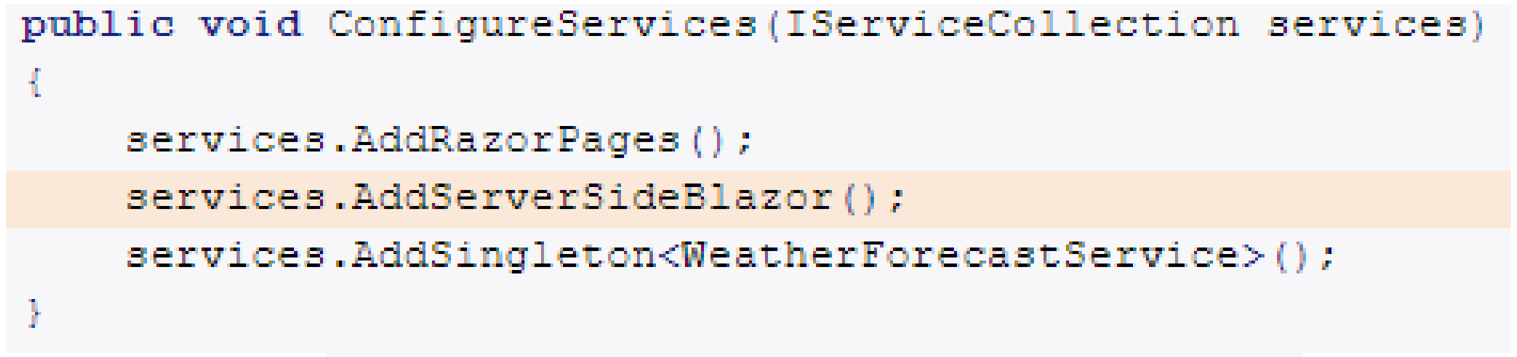
BLAZOR SERVER - WEBASSEMBLY

- The JavaScript Reference
  - _framework/blazor.webassembly.js
  - _framework/blazor.server.js

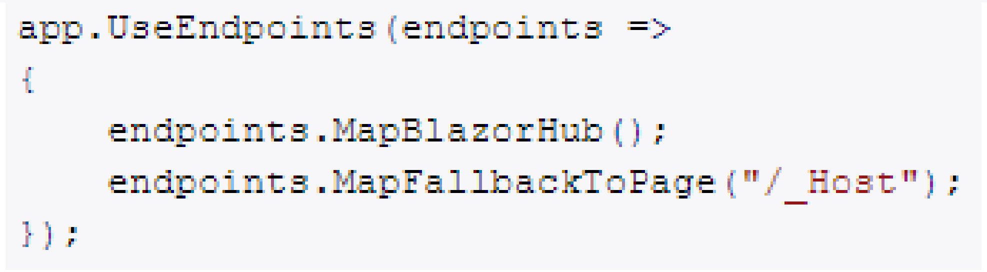


❑ Application Initialization

❖ Startup.cs



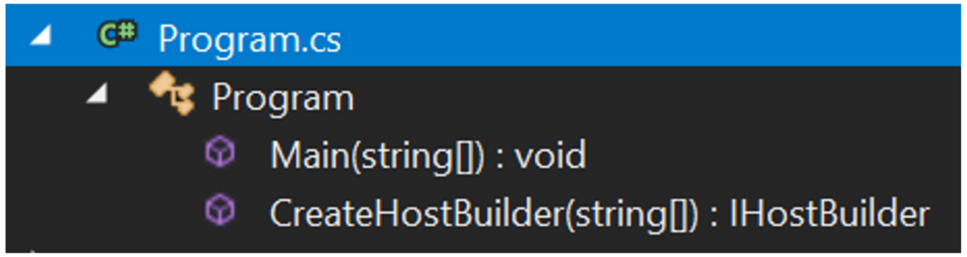
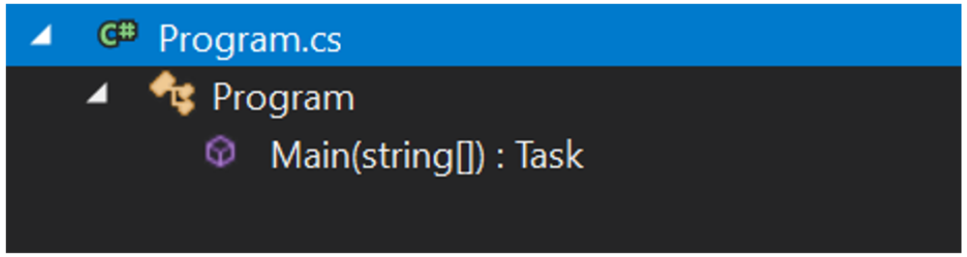
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddRazorPages();
    services.AddServerSideBlazor();
    services.AddSingleton<WeatherForecastService>();
}
```



```
app.UseEndpoints(endpoints =>
{
    endpoints.MapBlazorHub();
    endpoints.MapFallbackToPage("/_Host");
});
```

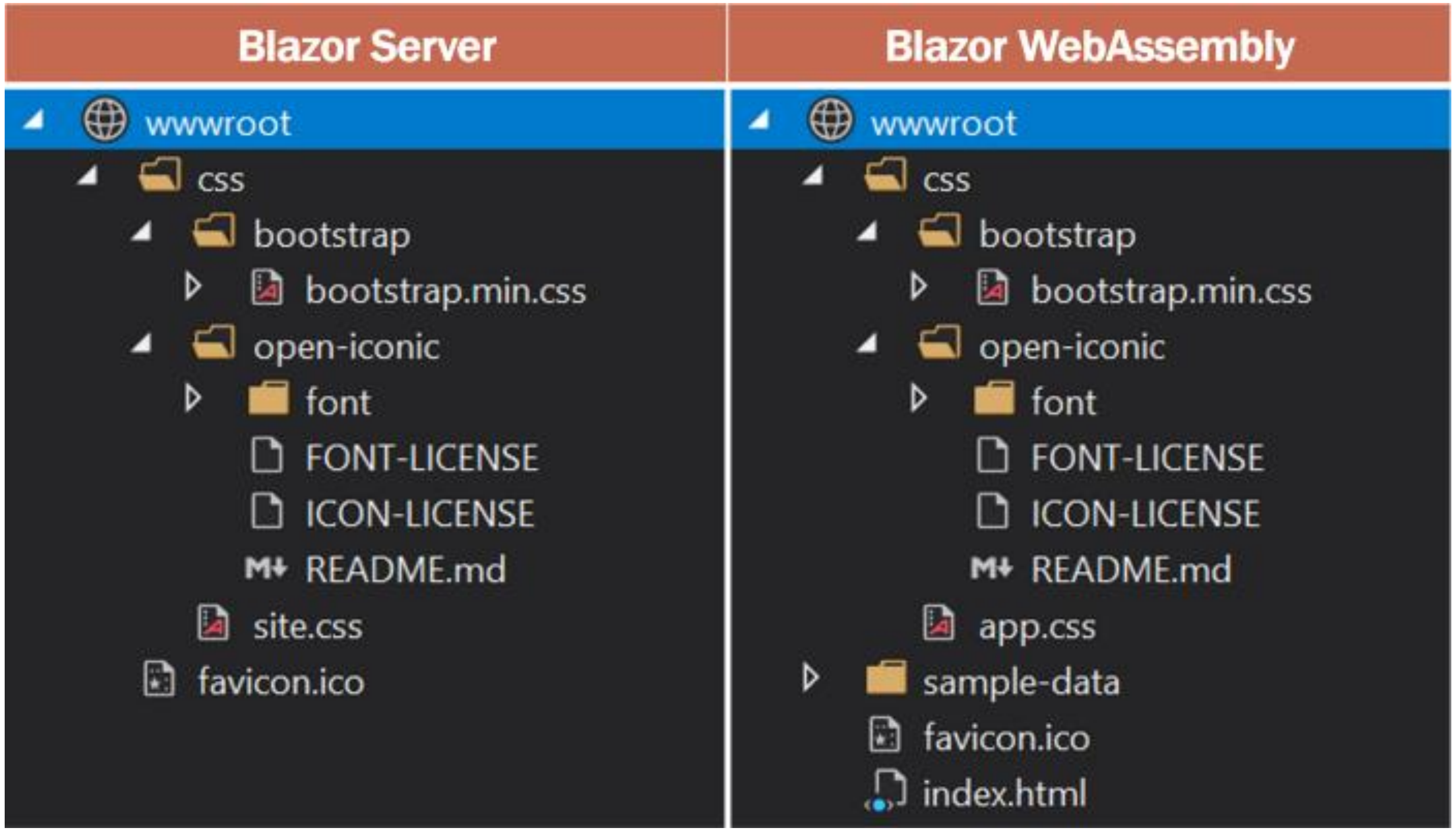
BLAZOR SERVER - WEBASSEMBLY

- Application Initialization
  - Program.cs

Blazor Server	Blazor WebAssembly
 Program.cs Program Main(string[]) : void CreateHostBuilder(string[]) : IHostBuilder	 Program.cs Program Main(string[]) : Task
The `Main()` method calls the `CreateHostBuilder()` method which sets the ASP.NET Core hosts.	The `Main()` method determines the basic components of the application. The component is in the `App.razor file`.

Blazor Server - WebAssembly

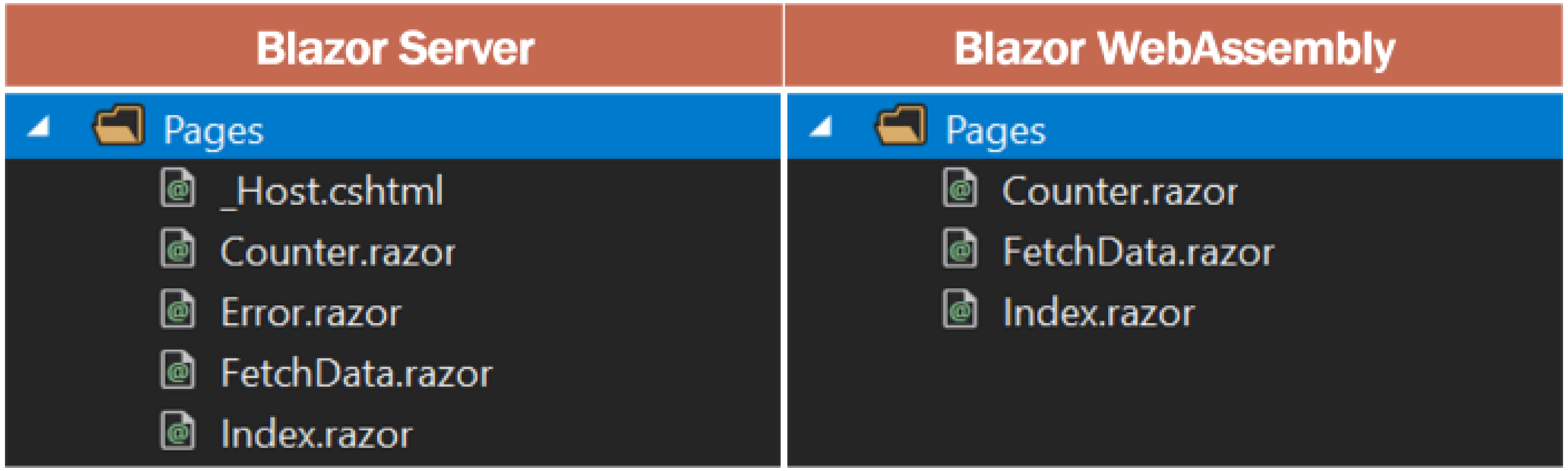
❑ wwwroot Folder



Blazor Server	Blazor WebAssembly
wwwroot	wwwroot
css	css
bootstrap	bootstrap
bootstrap.min.css	bootstrap.min.css
open-iconic	open-iconic
font	font
FONT-LICENSE	FONT-LICENSE
ICON-LICENSE	ICON-LICENSE
README.md	README.md
site.css	app.css
favicon.ico	sample-data
	favicon.ico
	index.html

BLAZOR SERVER - WEBASSEMBLY

❑ Pages Folder



Blazor Server	Blazor WebAssembly
Pages	Pages
_Host.cshtml	Counter.razor
Counter.razor	FetchData.razor
Error.razor	Index.razor
FetchData.razor	
Index.razor	

DEMO 3.1

Blazor WebAssembly Project Structure

FPT Education

FPT POLYTECHNIC

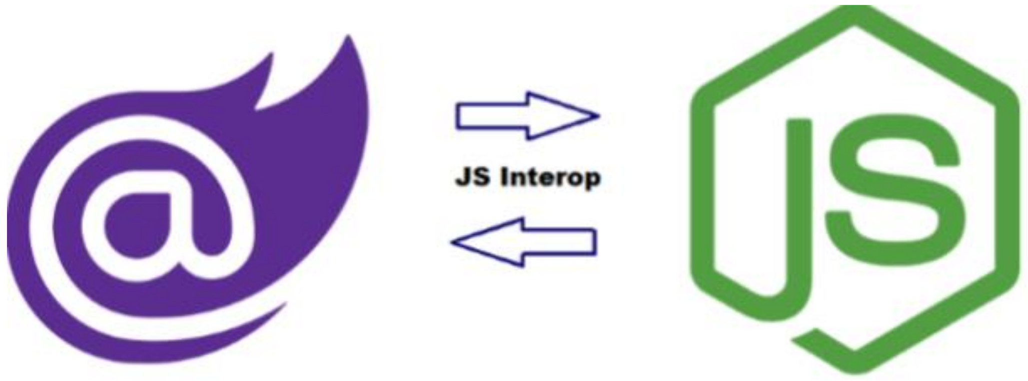
LẬP TRÌNH C# 6

BÀI 3: BLAZOR WEBASSEMBLY

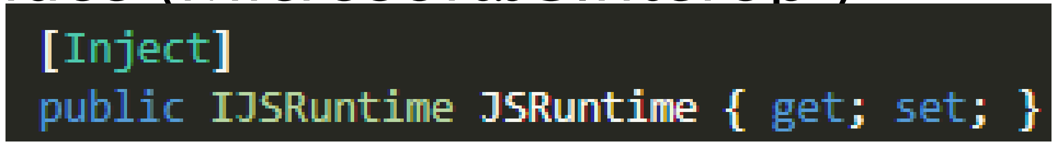
P3.2

FPT Education
FPT POLYTECHNIC

BLAZOR WEBASSEMBLY GỌI JAVASCRIPT



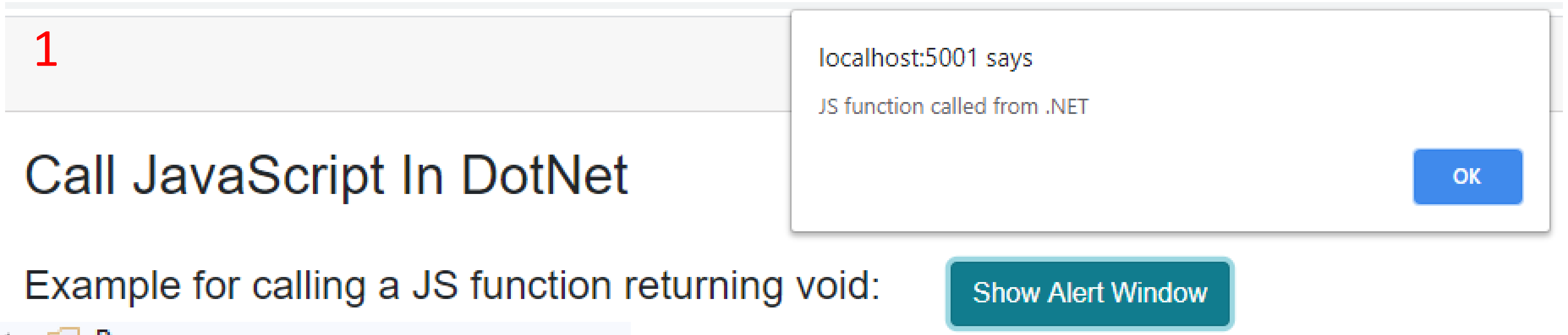
- IJSRuntime Interface (Microsoft.JSInterop)



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

- InvokeAsync<T>(name, args): JS function will return Value
- InvokeVoidAsync(name, args):JS function will return Void

INVOKEVOIDASYNC(NAME, ARGS)

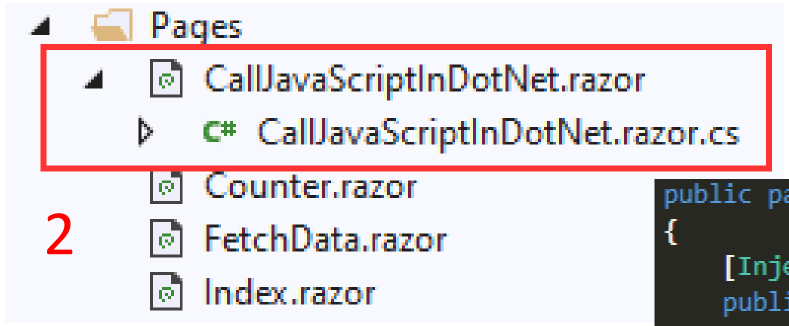

1

localhost:5001 says

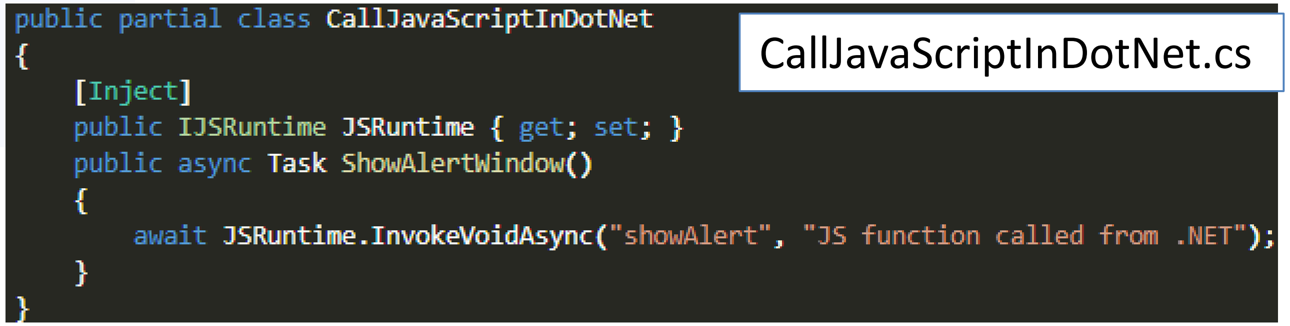
JS function called from .NET

Call JavaScript In DotNet

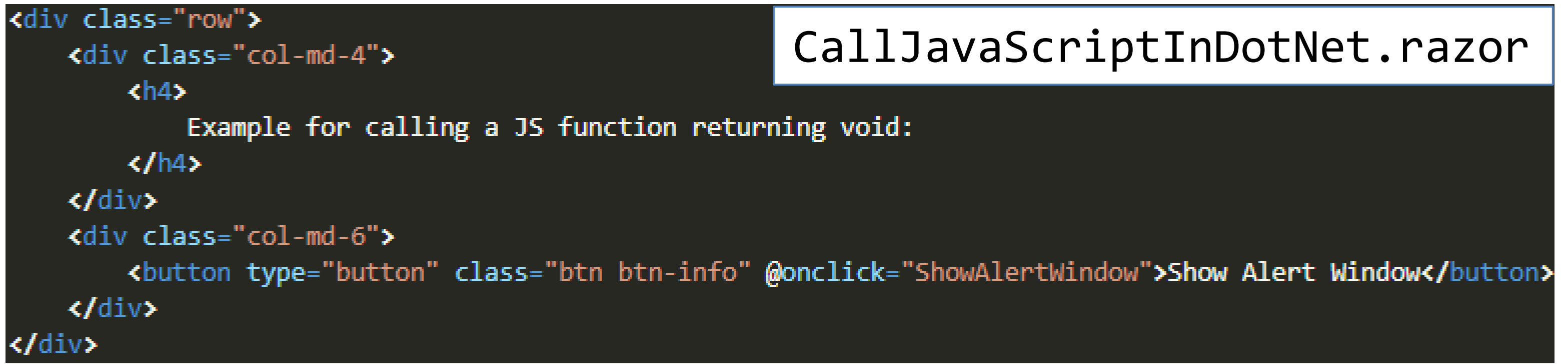
Example for calling a JS function returning void:



2


CallJavaScriptInDotNet.cs

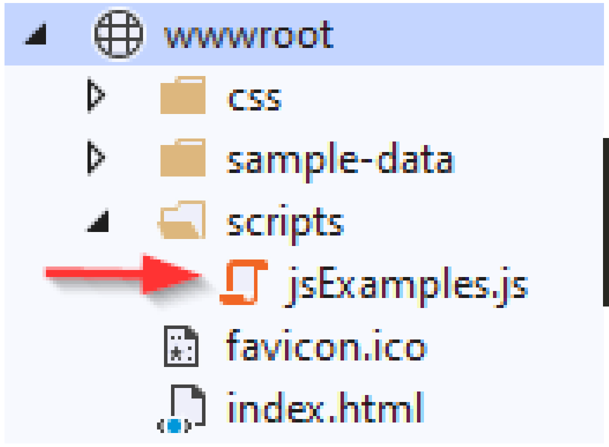
```
public partial class CallJavaScriptInDotNet
{
    [Inject]
    public IJSRuntime JSRuntime { get; set; }
    public async Task ShowAlertWindow()
    {
        await JSRuntime.InvokeVoidAsync("showAlert", "JS function called from .NET");
    }
}
```


CallJavaScriptInDotNet.razor

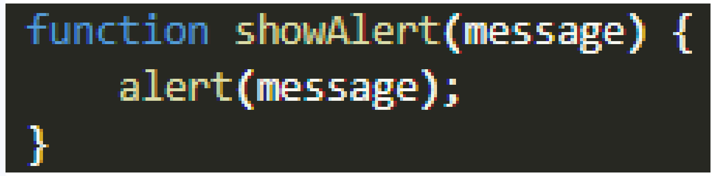
```
<div class="row">
    <div class="col-md-4">
        <h4>
            Example for calling a JS function returning void:
        </h4>
    </div>
    <div class="col-md-6">
        <button type="button" class="btn btn-info" @onclick="ShowAlertWindow">Show Alert Window</button>
    </div>
</div>
```

FPT POLYTECHNIC

InvokeVoidAsync(name, args)

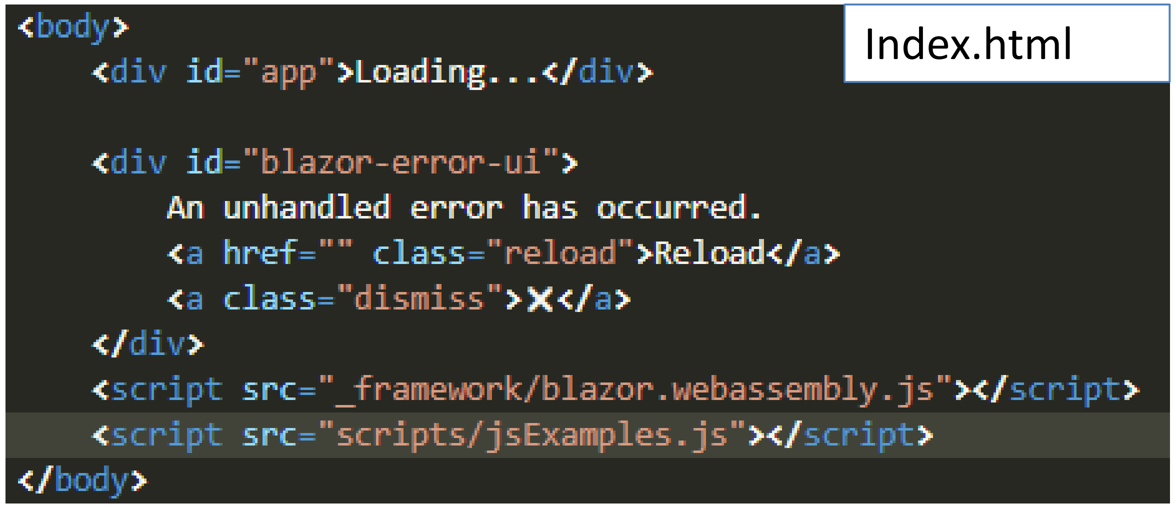


3



```
function showAlert(message) {
    alert(message);
}
```

4

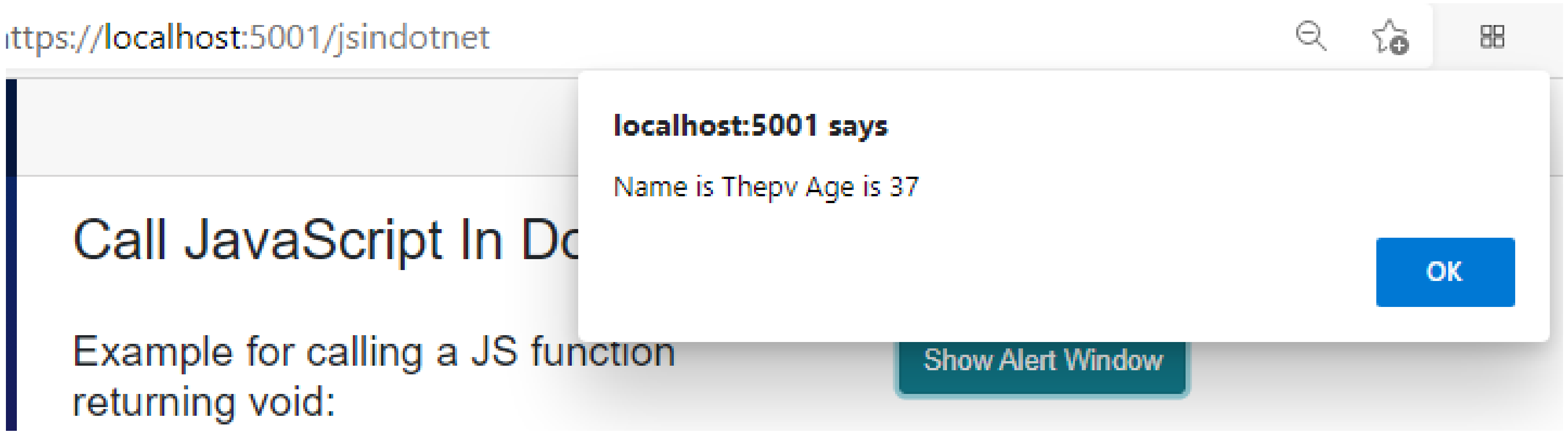


Index.html

```
<body>
    <div id="app">Loading...</div>

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

InvokeVoidAsync(name, args)

- Sending Parameters to JS Functions



ttps://localhost:5001/jsindotnet

localhost:5001 says

Name is Thepv Age is 37

1

```
private async Task ShowAlertWindow() =>
    await _jsModule.InvokeVoidAsync("showAlert", new { Name = "Thepv", Age = 37 });
```

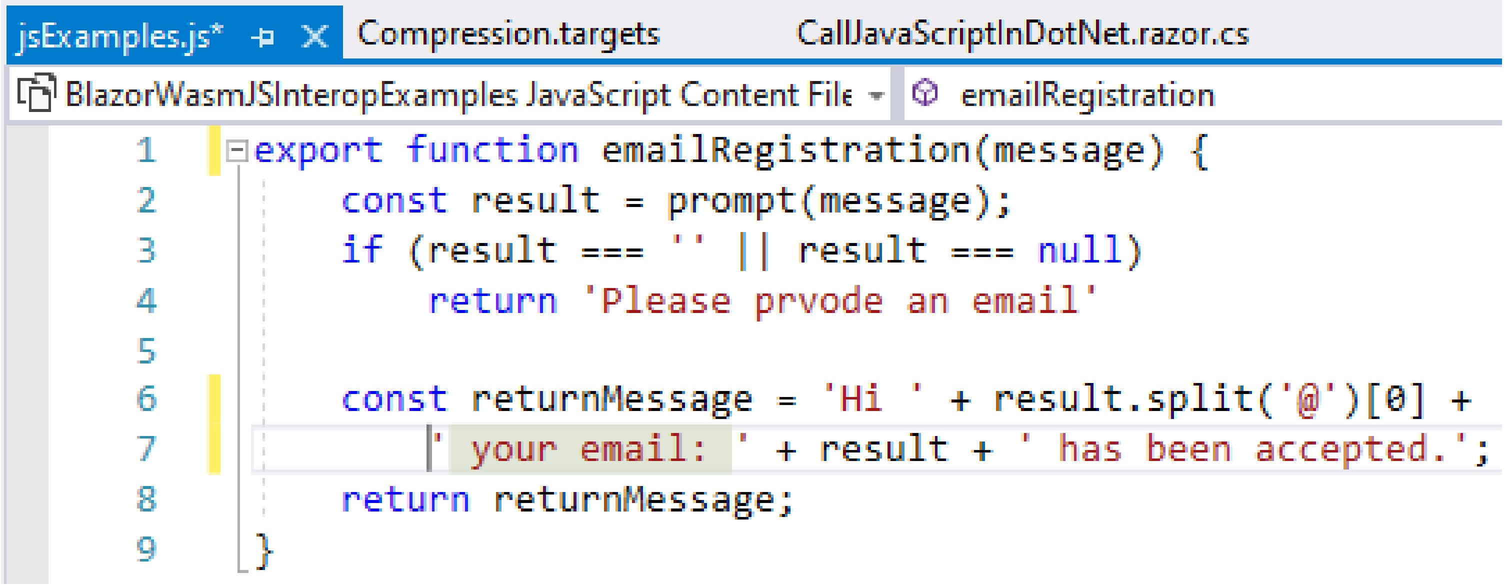
```
export function showAlert(obj) {
    const message = 'Name is ' + obj.name + ' Age is ' + obj.age;
    alert(message);
}
```

2

FPT Education
FPT POLYTECHNIC

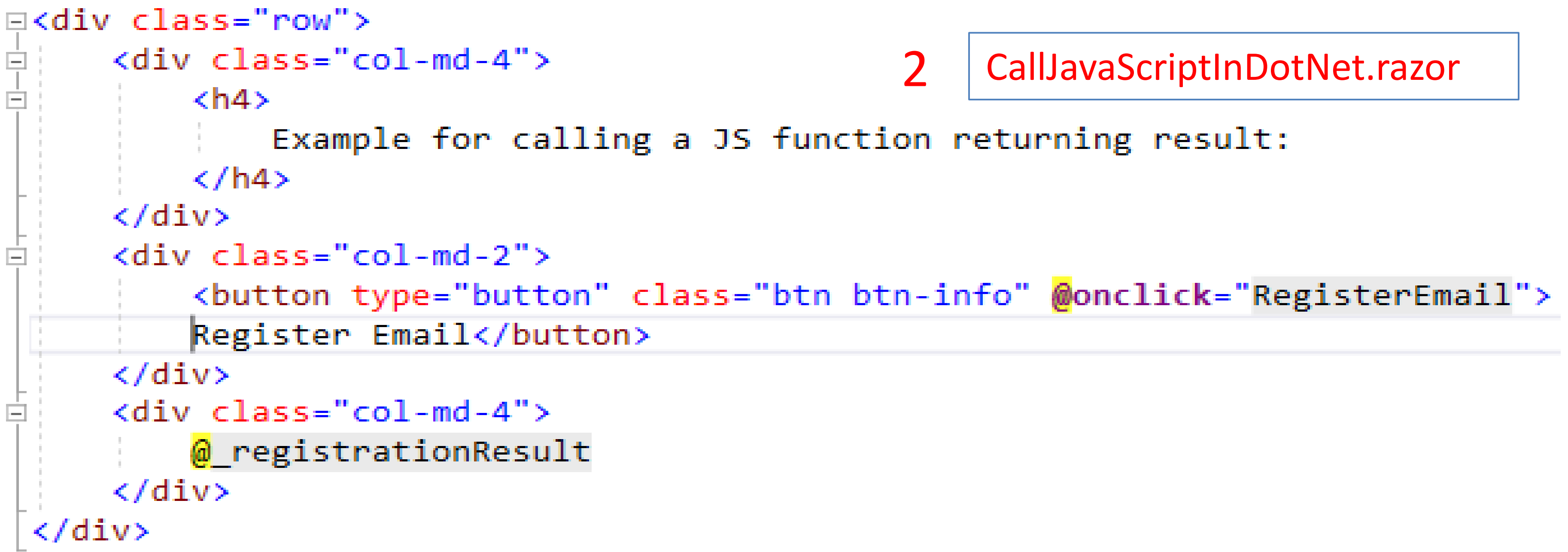
INVOKEASYNC<T>(NAME, ARGS)

1


jsExamples.js* | Compression.targets | CallJavaScriptInDotNet.razor.cs

BlazorWasmJSInteropExamples JavaScript Content File | emailRegistration

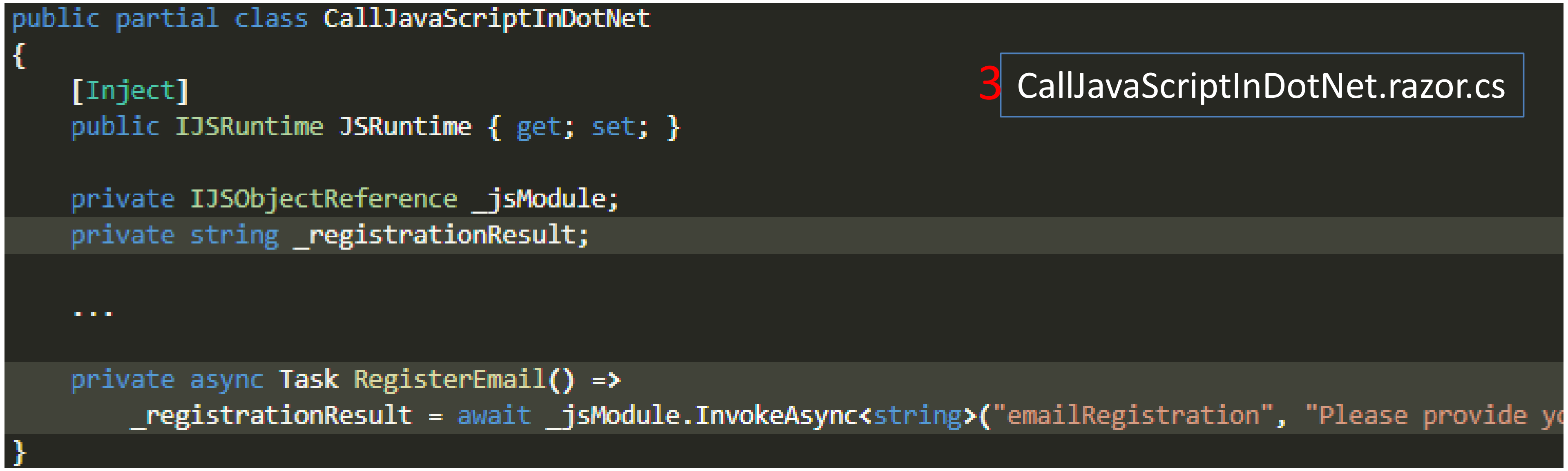
```
export function emailRegistration(message) {
    const result = prompt(message);
    if (result === '' || result === null)
        return 'Please prvode an email'

    const returnMessage = 'Hi ' + result.split('@')[0] +
        ' your email: ' + result + ' has been accepted.';
    return returnMessage;
}
```


```
<div class="row">
    <div class="col-md-4">
        <h4>
            Example for calling a JS function returning result:
        </h4>
    </div>
    <div class="col-md-2">
        <button type="button" class="btn btn-info" @onclick="RegisterEmail">
        Register Email</button>
    </div>
    <div class="col-md-4">
        @_registrationResult
    </div>
</div>
```

2 CallJavaScriptInDotNet.razor

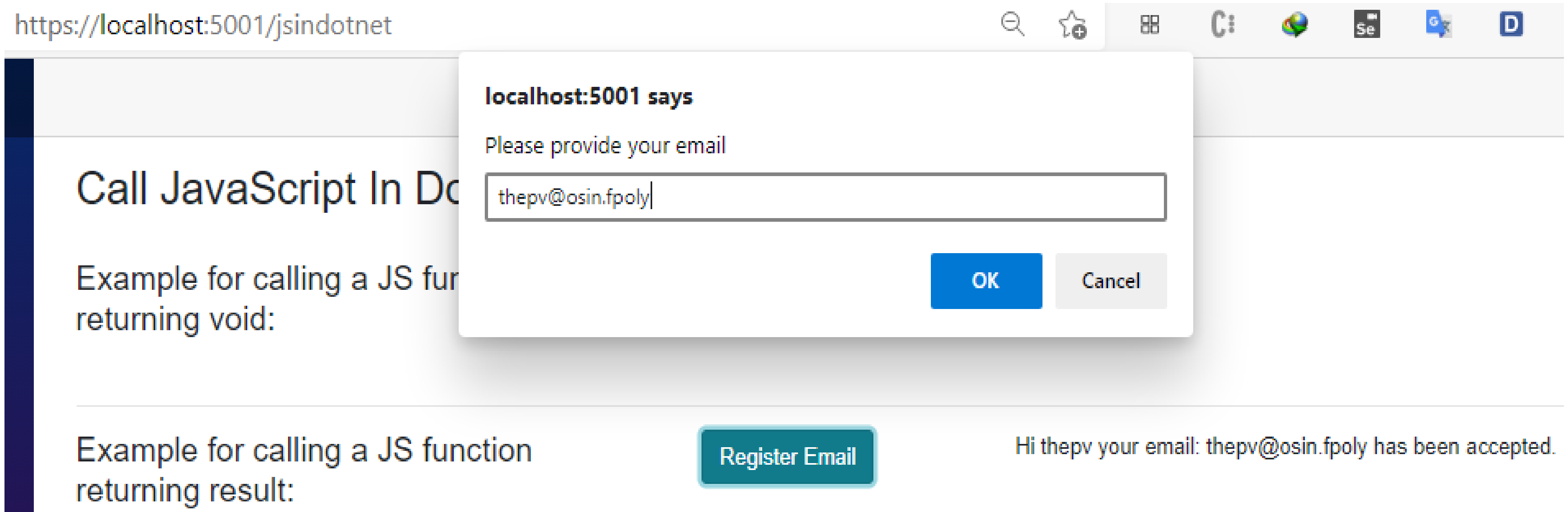
FPT Education
FPT POLYTECHNIC

InvokeAsync<T>(name, args)



3 CallJavaScriptInDotNet.razor.cs

```
public partial class CallJavaScriptInDotNet
{
    [Inject]
    public IJSRuntime JSRuntime { get; set; }

    private IJSObjectReference _jsModule;
    private string _registrationResult;

    ...

    private async Task RegisterEmail() =>
        _registrationResult = await _jsModule.InvokeAsync<string>("emailRegistration", "Please provide yo
}
```



https://localhost:5001/jsindotnet

Call JavaScript In Do

Example for calling a JS fur
returning void:

Example for calling a JS function
returning result:

localhost:5001 says

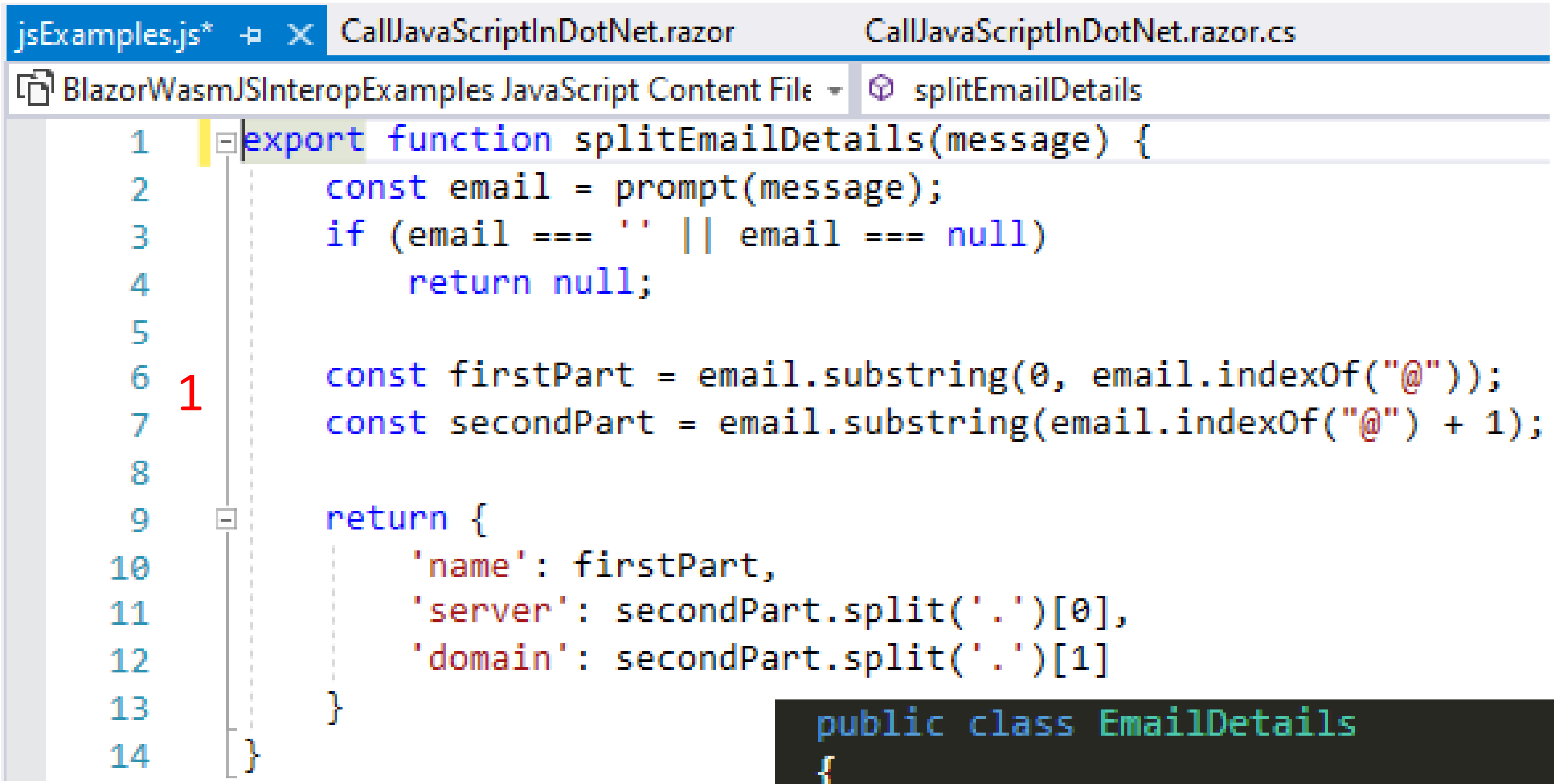
Please provide your email

thepv@osin.fpoly

Hi thepv your email: thepv@osin.fpoly has been accepted.

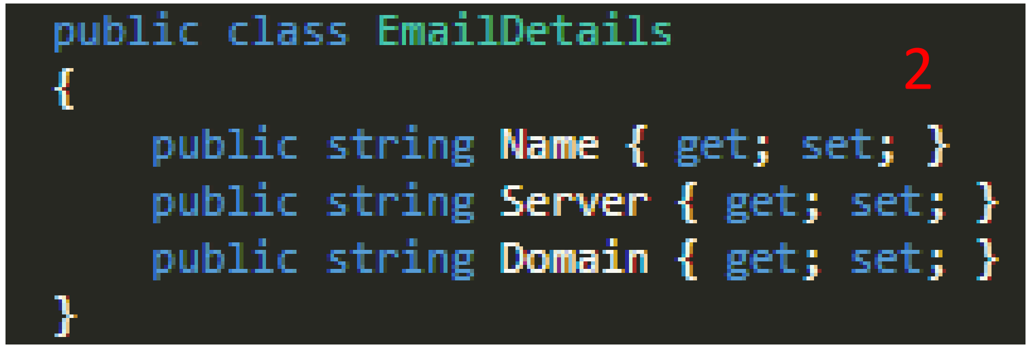
INVOKEASYNC<T>(NAME, ARGS) -OBJECT

returns an object



jsExamples.js*
CallJavaScriptInDotNet.razor
CallJavaScriptInDotNet.razor.cs
BlazorWasmJSInteropExamples JavaScript Content File
splitEmailDetails
1

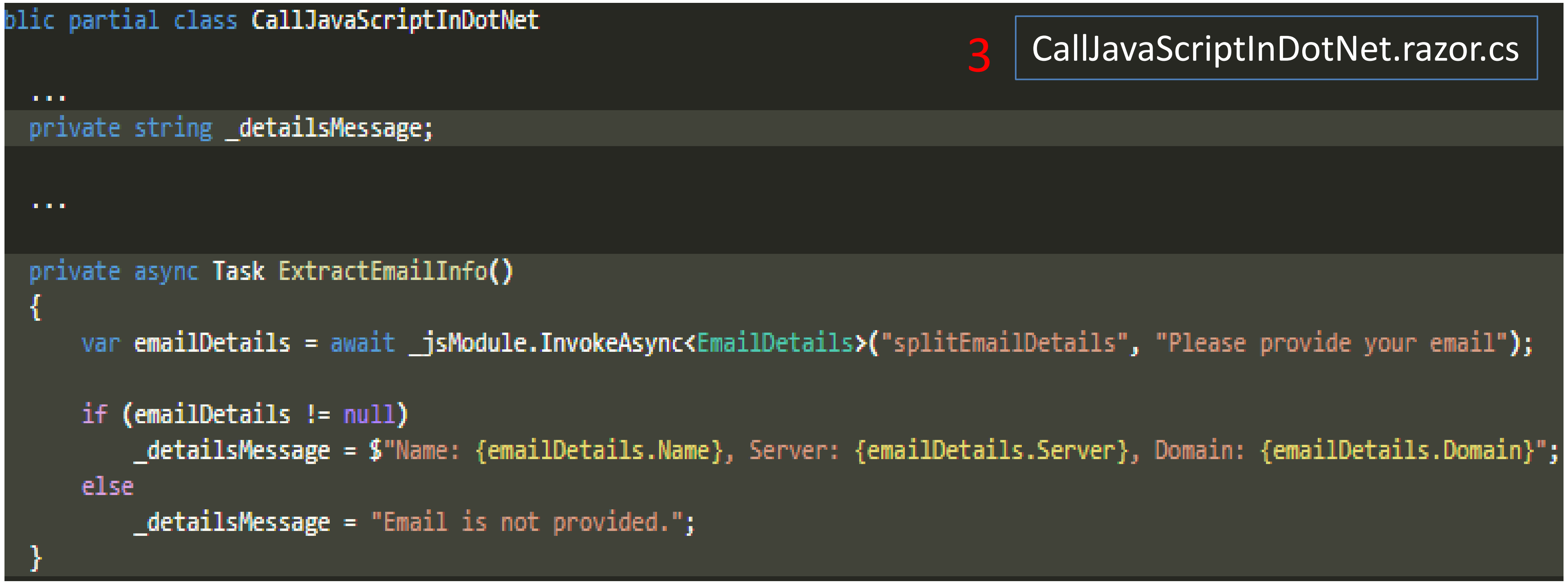
```
export function splitEmailDetails(message) {
    const email = prompt(message);
    if (email === '' || email === null)
        return null;

    const firstPart = email.substring(0, email.indexOf("@"));
    const secondPart = email.substring(email.indexOf("@") + 1);

    return {
        'name': firstPart,
        'server': secondPart.split('.')[0],
        'domain': secondPart.split('.')[1]
    }
}
```



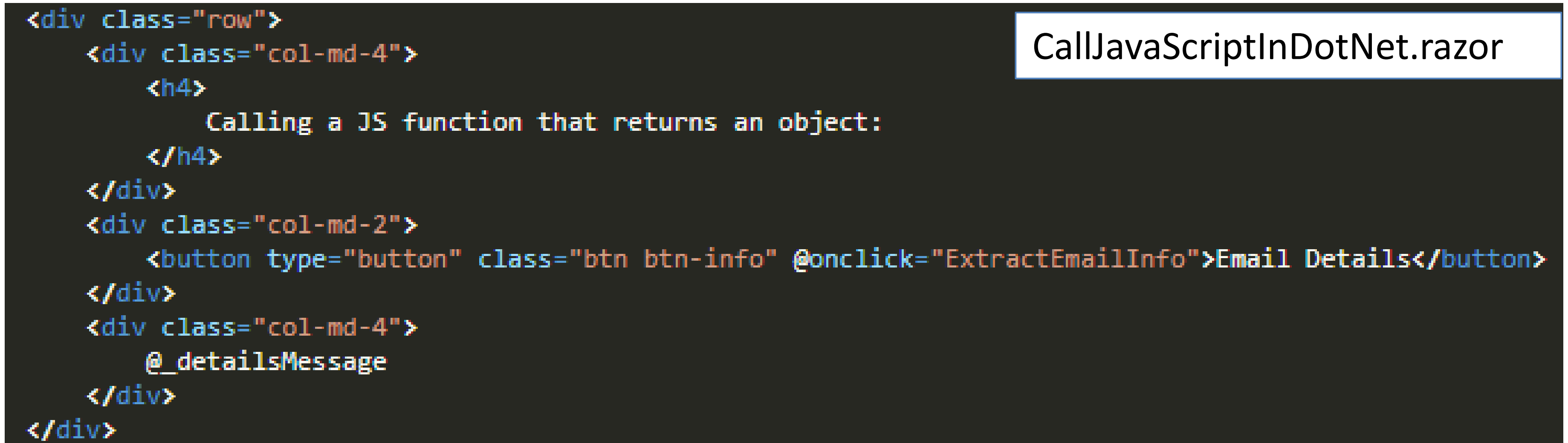
```
public class EmailDetails
{
    public string Name { get; set; }
    public string Server { get; set; }
    public string Domain { get; set; }
}
```

FPT Education
FPT POLYTECHNIC

INVOKEASYNC<T>(NAME, ARGS) -OBJECT


3 CallJavaScriptInDotNet.razor.cs

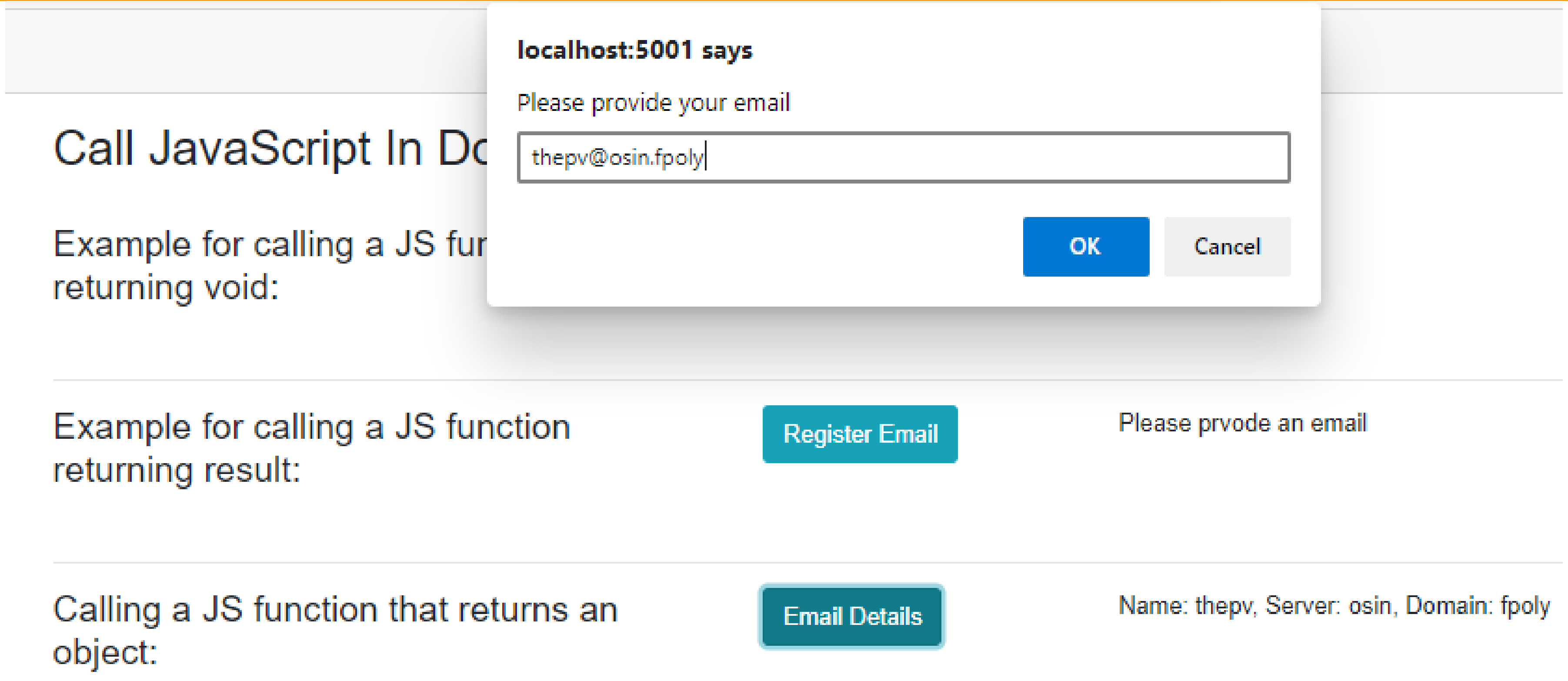
```
blic partial class CallJavaScriptInDotNet

  ...
  private string _detailsMessage;

  ...

  private async Task ExtractEmailInfo()
  {
      var emailDetails = await _jsModule.InvokeAsync<EmailDetails>("splitEmailDetails", "Please provide your email");

      if (emailDetails != null)
          _detailsMessage = $"Name: {emailDetails.Name}, Server: {emailDetails.Server}, Domain: {emailDetails.Domain}";
      else
          _detailsMessage = "Email is not provided.";
  }
```


CallJavaScriptInDotNet.razor

```
<div class="row">
    <div class="col-md-4">
        <h4>
            Calling a JS function that returns an object:
        </h4>
    </div>
    <div class="col-md-2">
        <button type="button" class="btn btn-info" @onclick="ExtractEmailInfo">Email Details</button>
    </div>
    <div class="col-md-4">
        @_detailsMessage
    </div>
</div>
```

FPT Education
FPT POLYTECHNIC

INVOKEASYNC<T>(NAME, ARGS) -OBJECT


localhost:5001 says

Please provide your email

thepv@osin.fpoly

Call JavaScript In Dc

Example for calling a JS fur
returning void:

Example for calling a JS function returning result:

Please prvode an email

Calling a JS function that returns an object:

Name: thepv, Server: osin, Domain: fpoly

DEMO 3.2

Blazor WebAssembly Gọi JavaScript

FPT Education

FPT POLYTECHNIC

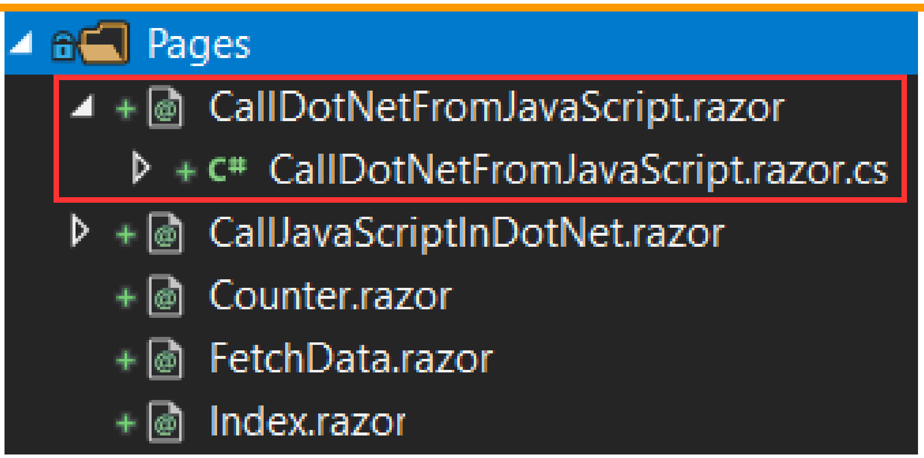
LẬP TRÌNH C# 6

BÀI 3: BLAZOR WEBASSEMBLY

P3.3

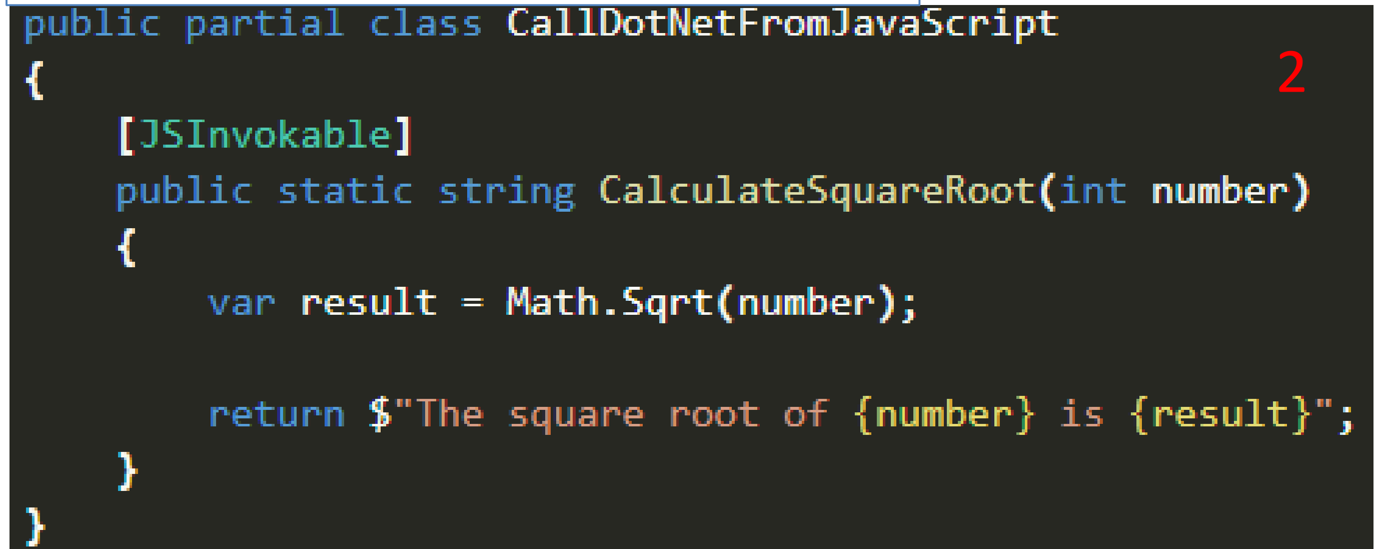
FPT Education
FPT POLYTECHNIC

JAVASCRIPT GỌI C# METHOD


1

```
Pages
  CallDotNetFromJavaScript.razor
    CallDotNetFromJavaScript.razor.cs
  CallJavaScriptInDotNet.razor
  Counter.razor
  FetchData.razor
  Index.razor
```

CallDotNetFromJavaScript.razor.cs

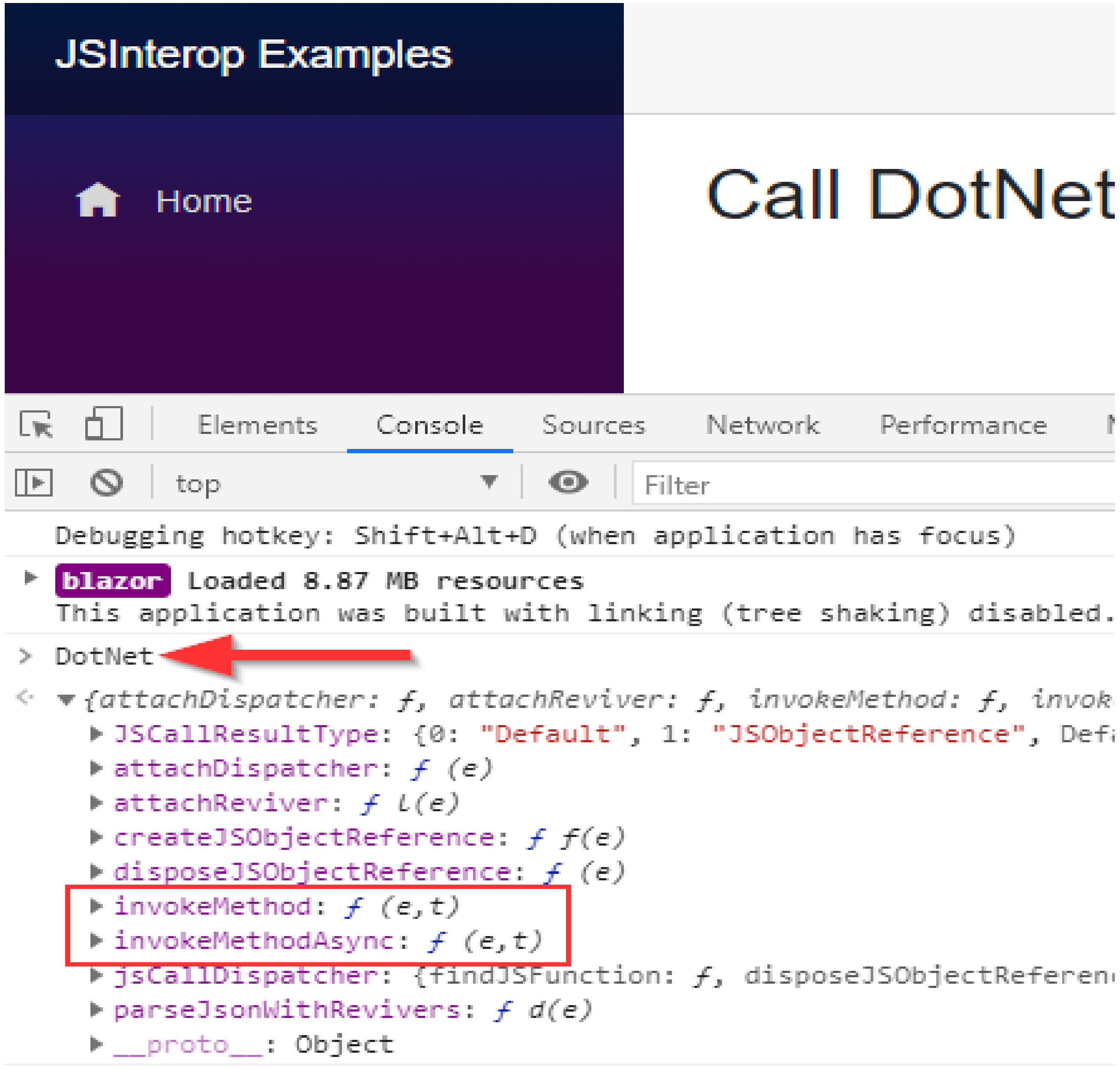

2

```csharp
public partial class CallDotNetFromJavaScript
{
    [JSInvokable]
    public static string CalculateSquareRoot(int number)
    {
        var result = Math.Sqrt(number);

        return $"The square root of {number} is {result}";
    }
}
```

JAVASCRIPT GỌI C# METHOD

- ❑ DotNet object
- ❑ invokeMethod
- ❑ invokeMethodAsync

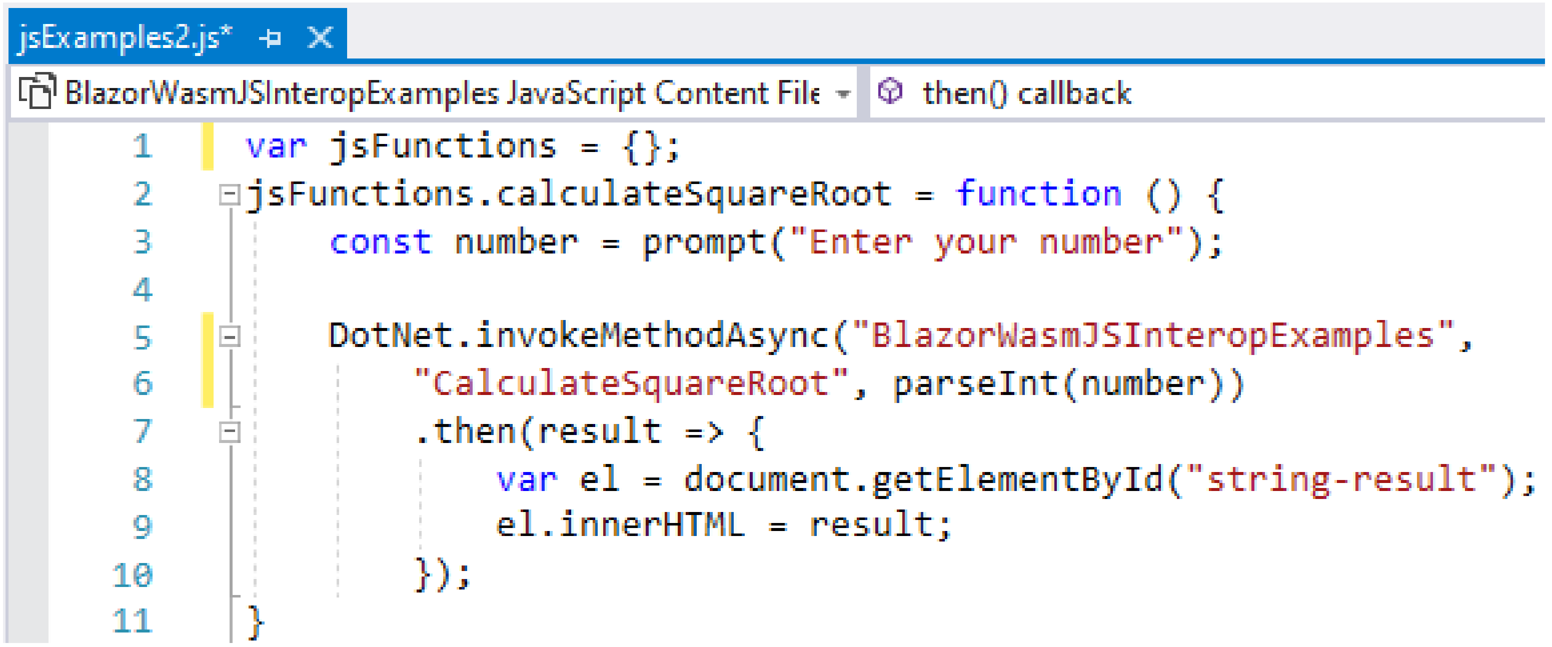
3



```
Debugging hotkey: Shift+Alt+D (when application has focus)
▶ blazor Loaded 8.87 MB resources
  This application was built with linking (tree shaking) disabled.
> DotNet
<· ▼{attachDispatcher: ƒ, attachReviver: ƒ, invokeMethod: ƒ, invok
    ▶ JSCallResultType: {0: "Default", 1: "JSObjectReference", Def
    ▶ attachDispatcher: ƒ (e)
    ▶ attachReviver: ƒ L(e)
    ▶ createJSObjectReference: ƒ f(e)
    ▶ disposeJSObjectReference: ƒ (e)
    ▶ invokeMethod: ƒ (e,t)
    ▶ invokeMethodAsync: ƒ (e,t)
    ▶ jsCallDispatcher: {findJSFunction: ƒ, disposeJSObjectReferen
    ▶ parseJsonWithRevivers: ƒ d(e)
    ▶ __proto__: Object
```

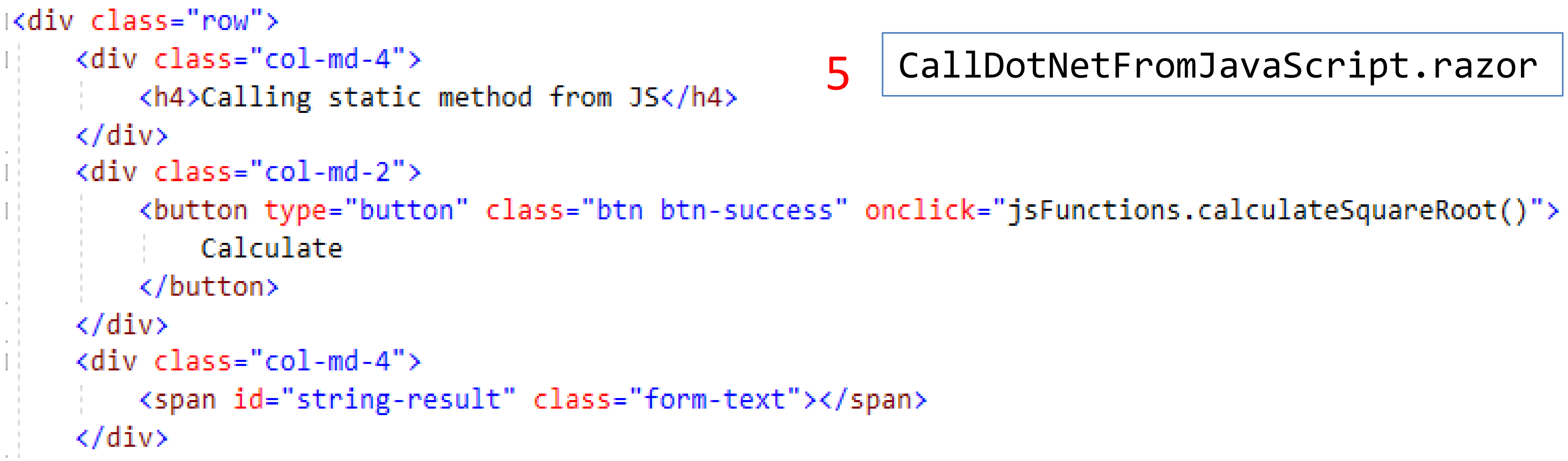
FPT Education
FPT POLYTECHNIC

JAVASCRIPT GỌI C# METHOD


jsExamples2.js*
BlazorWasmJSInteropExamples JavaScript Content File | then() callback

```
var jsFunctions = {};
jsFunctions.calculateSquareRoot = function () {
    const number = prompt("Enter your number");

    DotNet.invokeMethodAsync("BlazorWasmJSInteropExamples",
        "CalculateSquareRoot", parseInt(number))
        .then(result => {
            var el = document.getElementById("string-result");
            el.innerHTML = result;
        });
}
```

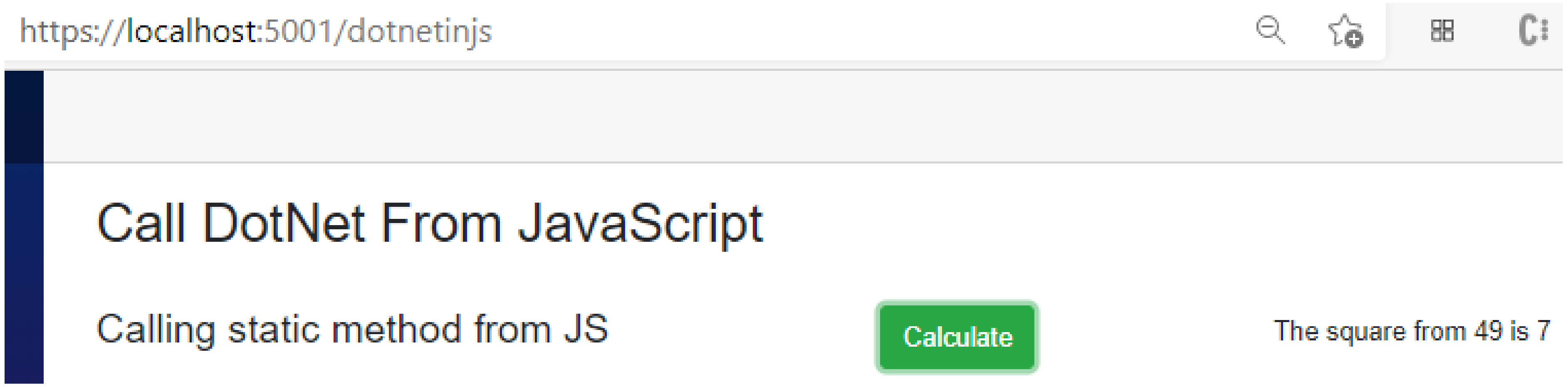
4


5

CallDotNetFromJavaScript.razor

```
<div class="row">
    <div class="col-md-4">
        <h4>Calling static method from JS</h4>
    </div>
    <div class="col-md-2">
        <button type="button" class="btn btn-success" onclick="jsFunctions.calculateSquareRoot()">
            Calculate
        </button>
    </div>
    <div class="col-md-4">
        <span id="string-result" class="form-text"></span>
    </div>
```

JAVASCRIPT GỌI C# METHOD



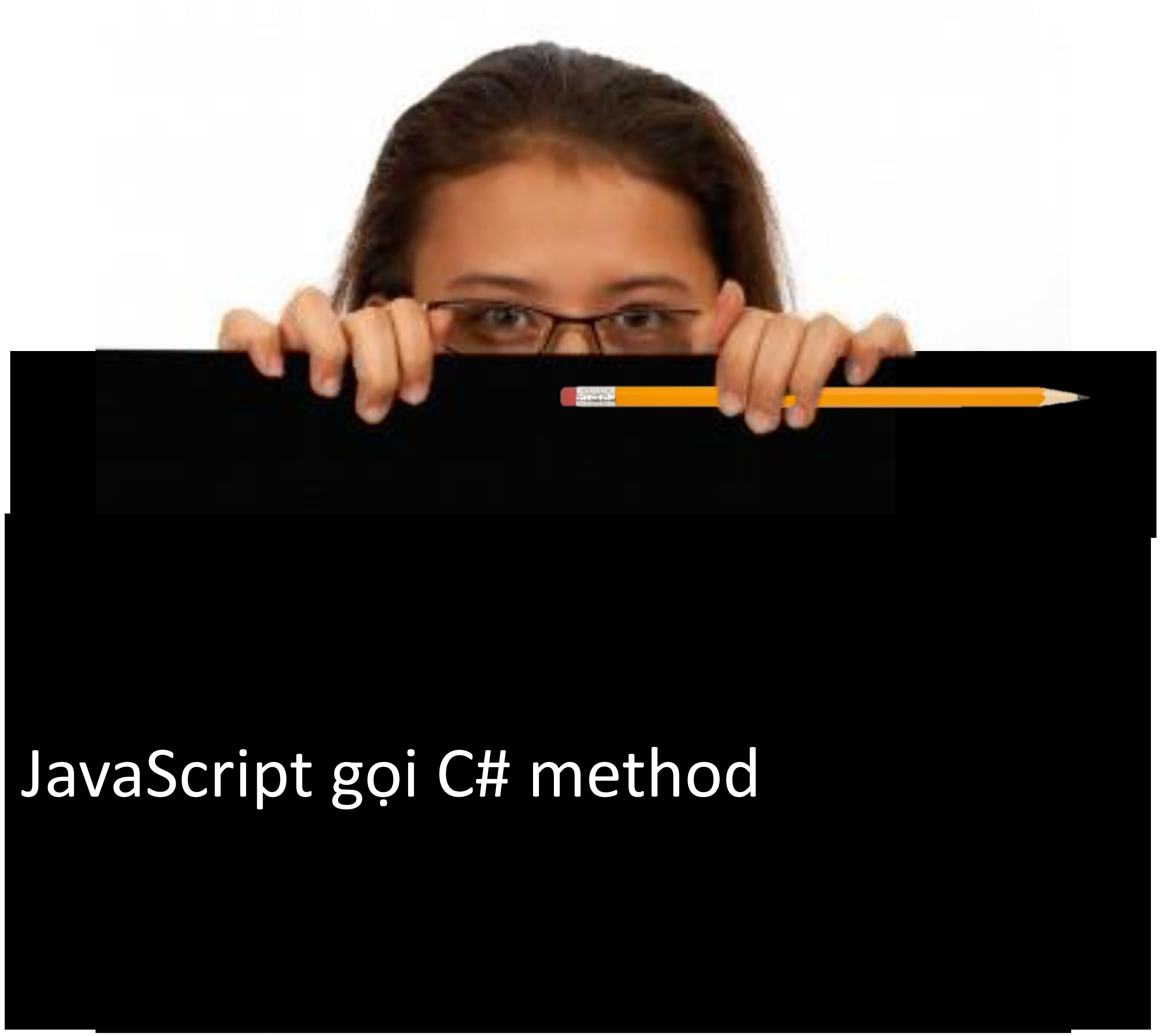
https://localhost:5001/dotnetinjs

Call DotNet From JavaScript

Calling static method from JS

The square from 49 is 7

DEMO 3.3



JavaScript gọi C# method

Tổng kết bài học

- ⦿Blazor WebAssembly
- ⦿Blazor WebAssembly Gọi JavaScript
- ⦿JavaScript gọi C# method

KẾT THÚC