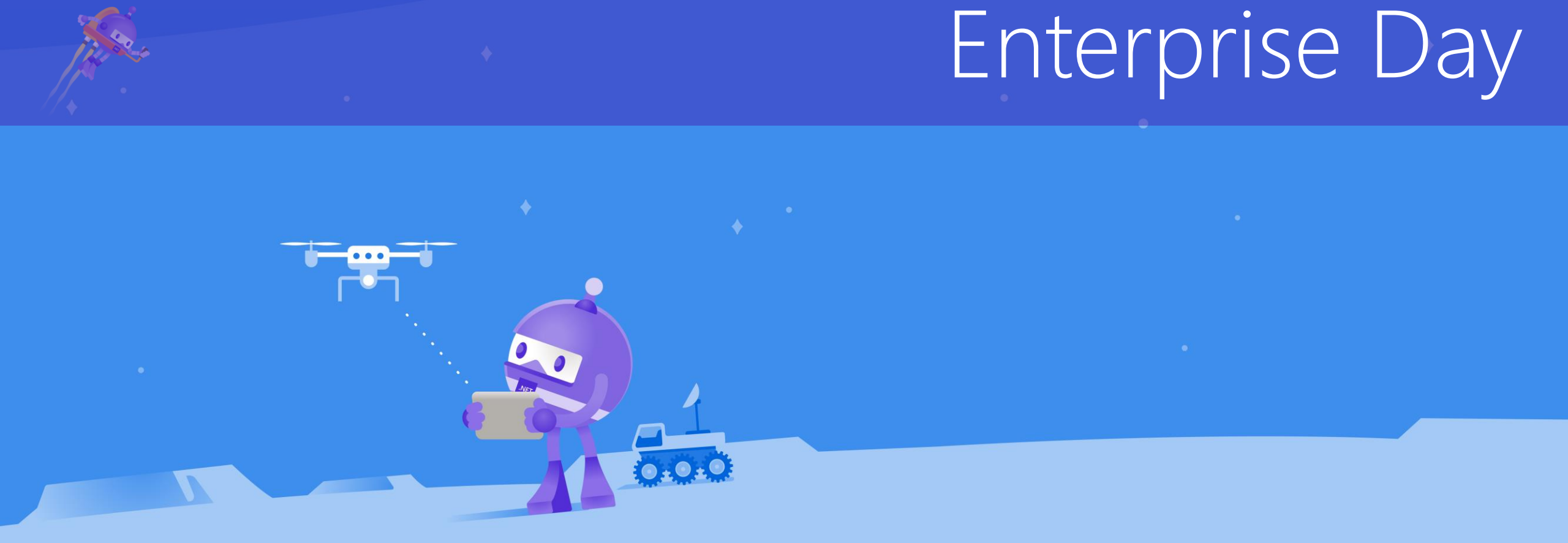
Microsoft STUDY4 Build School

.NET Conf 2020

Enterprise Day



ABOUT ME

- Duran Hsieh
  - Microsoft CE (App/Infra)
  - Other position
    - Senior .NET Developer
    - Taiwan Software Engineering Association Director
    - Study4.TW .NET Community Member
    - Taichung Google Developer Group Co-Organizer
    - 2016-2018 Microsoft MVP (VSDT)

Microsoft STUDY4 Build School

使用 Blazor WebAssembly 和 Visual Studio Code 來建置 Web 應用程式

台灣微軟企業支援服務經理

Duran Hsieh

Microsoft STUDY4 Build School

Overview

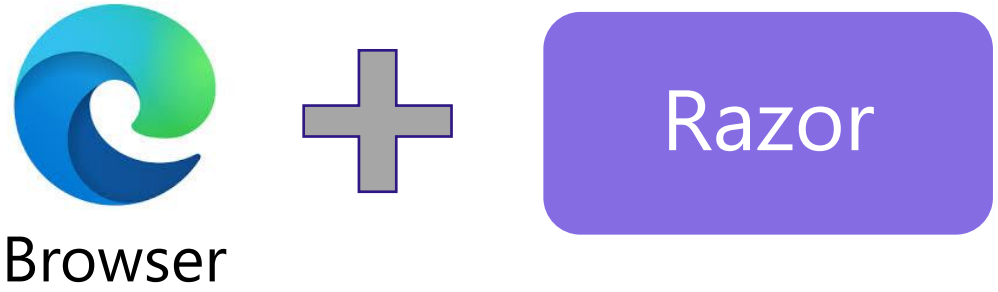
Microsoft STUDY4 Build School

Agenda

- Overview
- Hosting Modules
- Project Structure & Layout
- Blazor Components
- Dependency Injection
- Form in Blazor
- JavaScript Interop

什麼是 Blazor

- Framework
- 透過 C# 與 HTML 組成
- 建立可互動性 Web Uis
- 以 WebAssembly 為基礎
- 不須任何套件，使用 Web 標準
- 可以與 JavaScripts 整合
- 可以使用 Visual Studio 與 .NET 優勢
  - 效能檢測
  - 相關套件

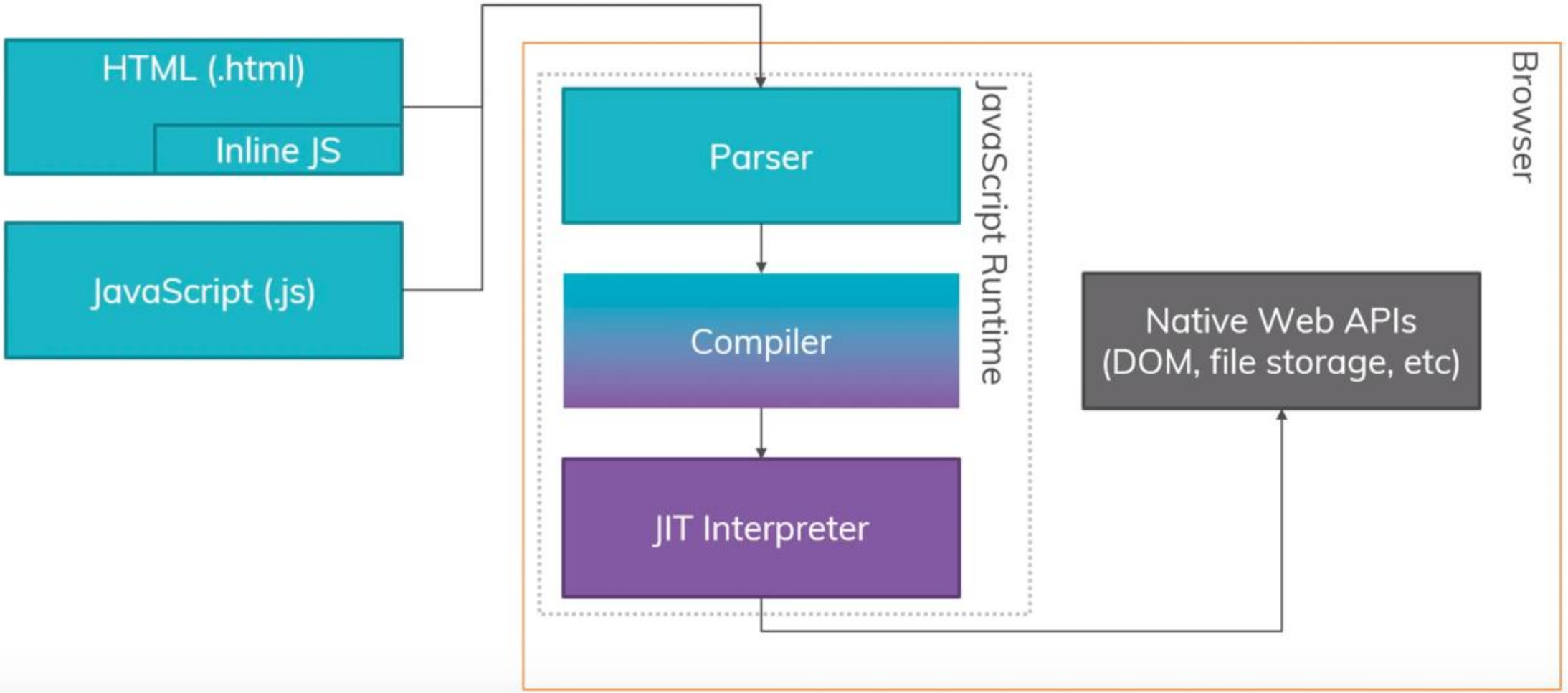


WebAssembly (wasm)

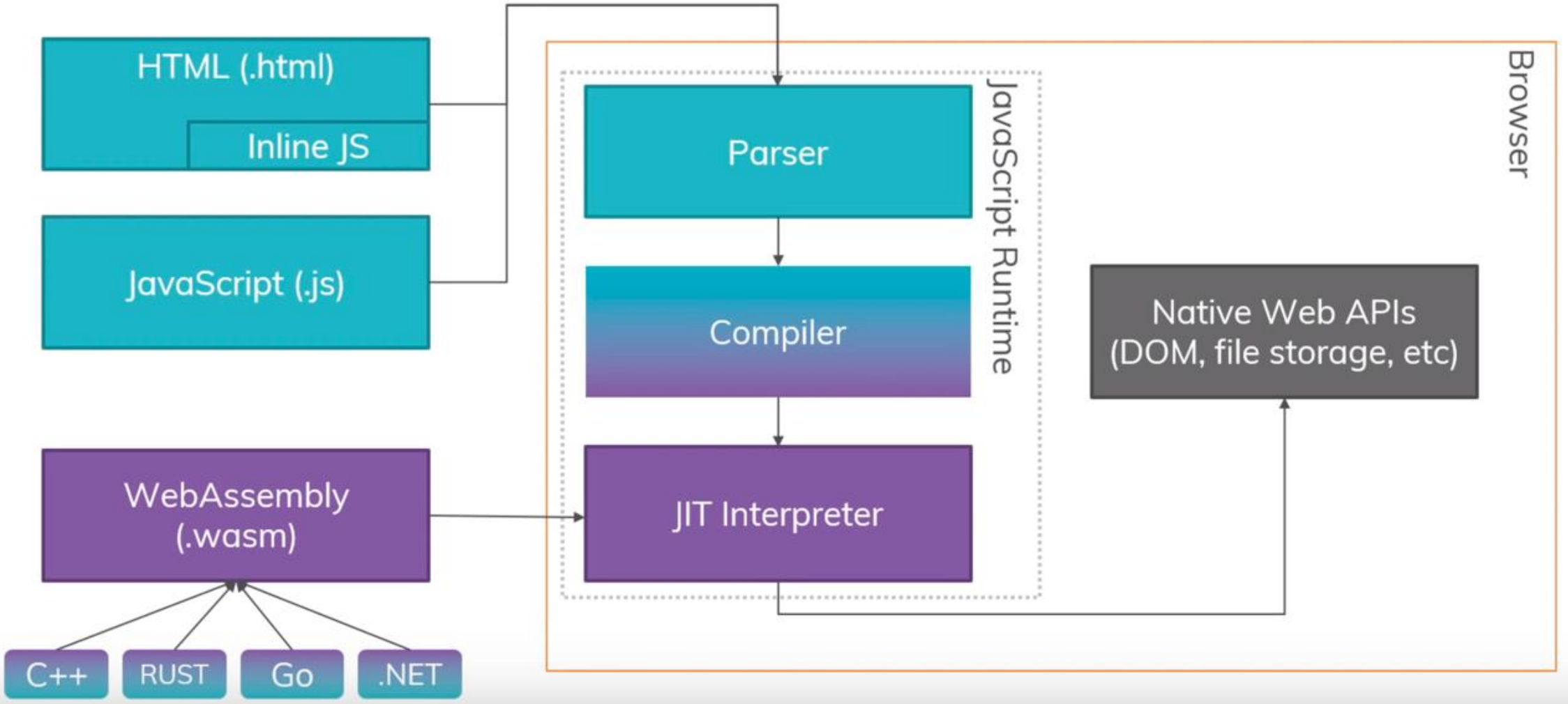
- 一種開放式二進位標準
- 應用於瀏覽器內的客戶端
- 設計提供比 JavaScript 更快速的編譯及執行
- 可讓開發者運用自己熟悉的語言編譯
- 開發團隊分別來自 Mozilla、Google、Microsoft、Apple，代表著四大網路瀏覽器Firefox、Chrome、Microsoft Edge、Safari
- 與 HTML，CSS 和 JavaScript 一起，成為 Web 的第四種語言

參考資料:https://zh.wikipedia.org/wiki/WebAssembly

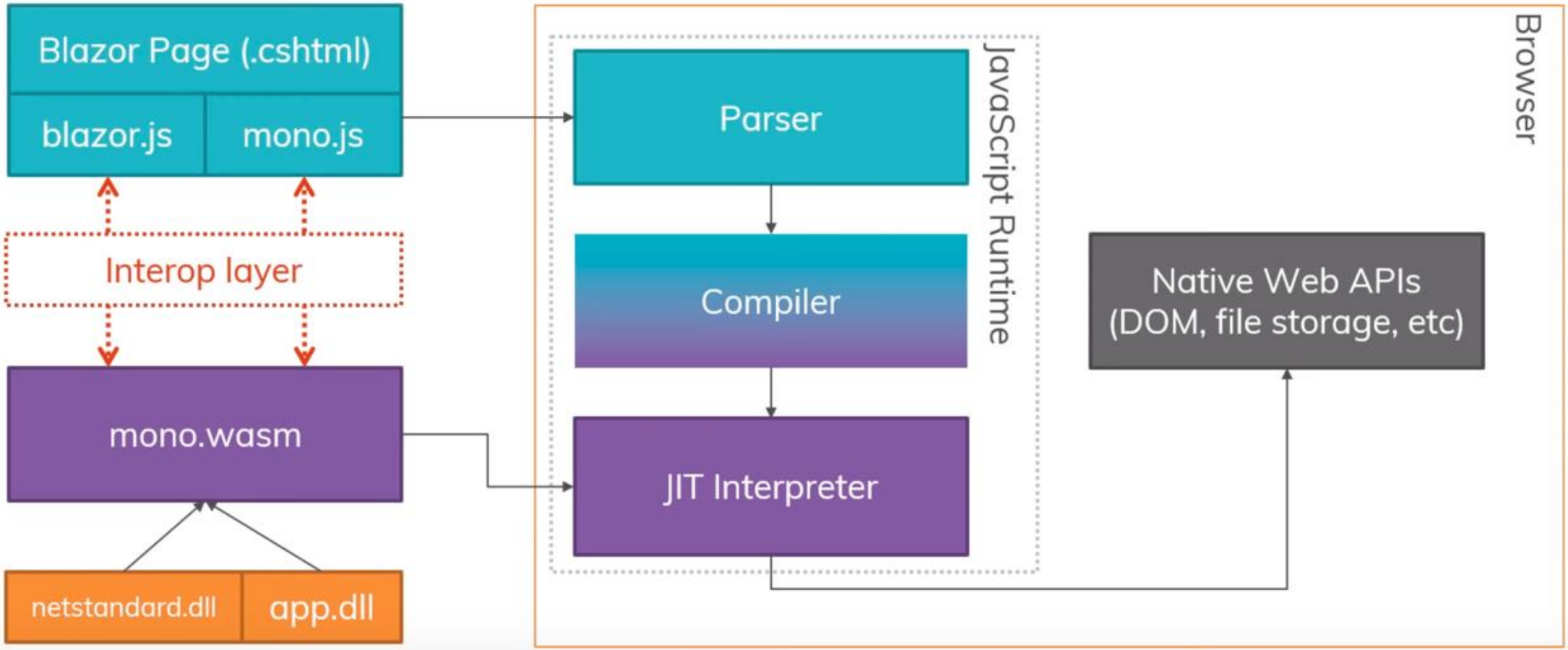
JavaScript


HTML (.html)
Inline JS
JavaScript (.js)
Parser
Compiler
JIT Interpreter
JavaScript Runtime
Native Web APIs
(DOM, file storage, etc)
Browser

WebAssembly



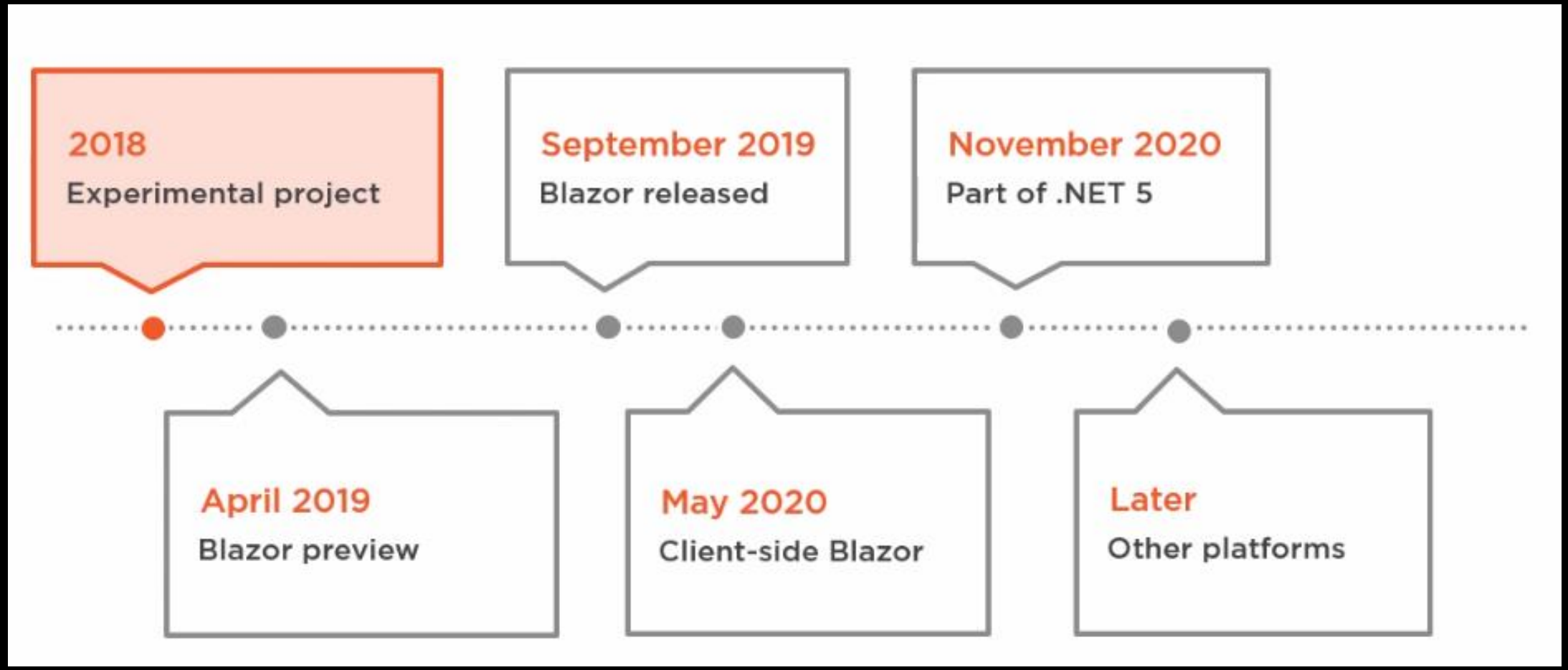
Blazor


Blazor Page (.cshtml)
blazor.js
mono.js
Interop layer
mono.wasm
netstandard.dll
app.dll
Parser
Compiler
JIT Interpreter
JavaScript Runtime
Native Web APIs
(DOM, file storage, etc)
Browser

Blazor 優勢

- 加速應用程式開發
- 降低 Build pipeline 的複雜度
- 簡化維護工作
- 讓開發人員了解並使用用戶端與伺服器端程式碼

Blazor Roadmap



必要基礎知識

- HTML & CSS (Web 應用程式概念的基本知識)
- C# (初學者等級)
- Some Razor (推薦，但非必要)

需要環境

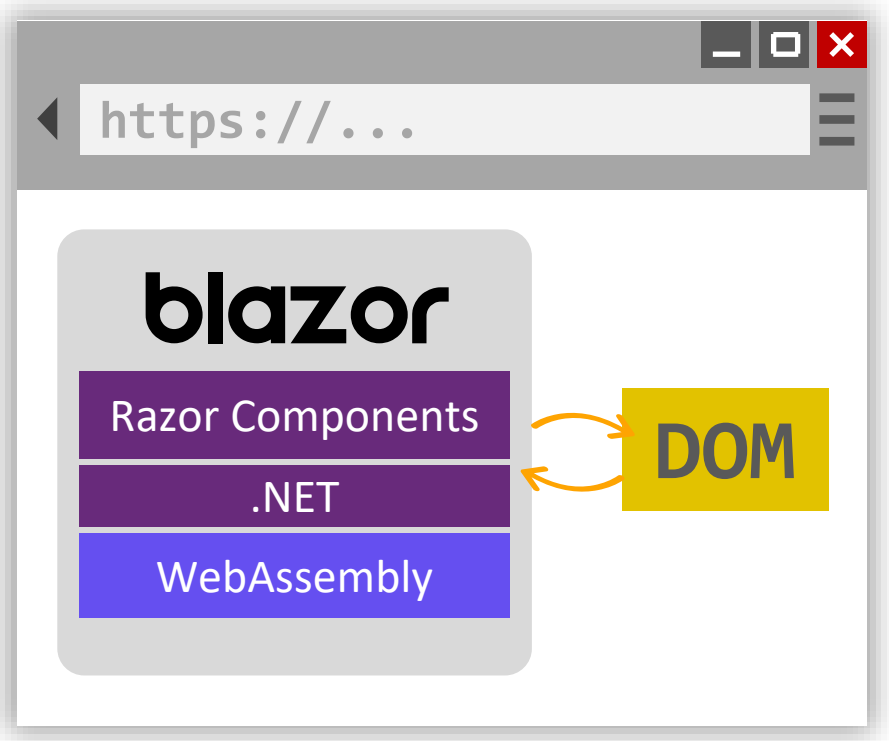
- IDE
  - Visual Studio Code
    - C#
    - C# 擴充套件
    - JavaScript Debugger
  - Visual Studio 2019 (16.6 or High)
- .NET Core 3.1 SDK
- .NET 5 compatible (所有程式碼皆相容)
- 瀏覽器

Microsoft STUDY4 Build School

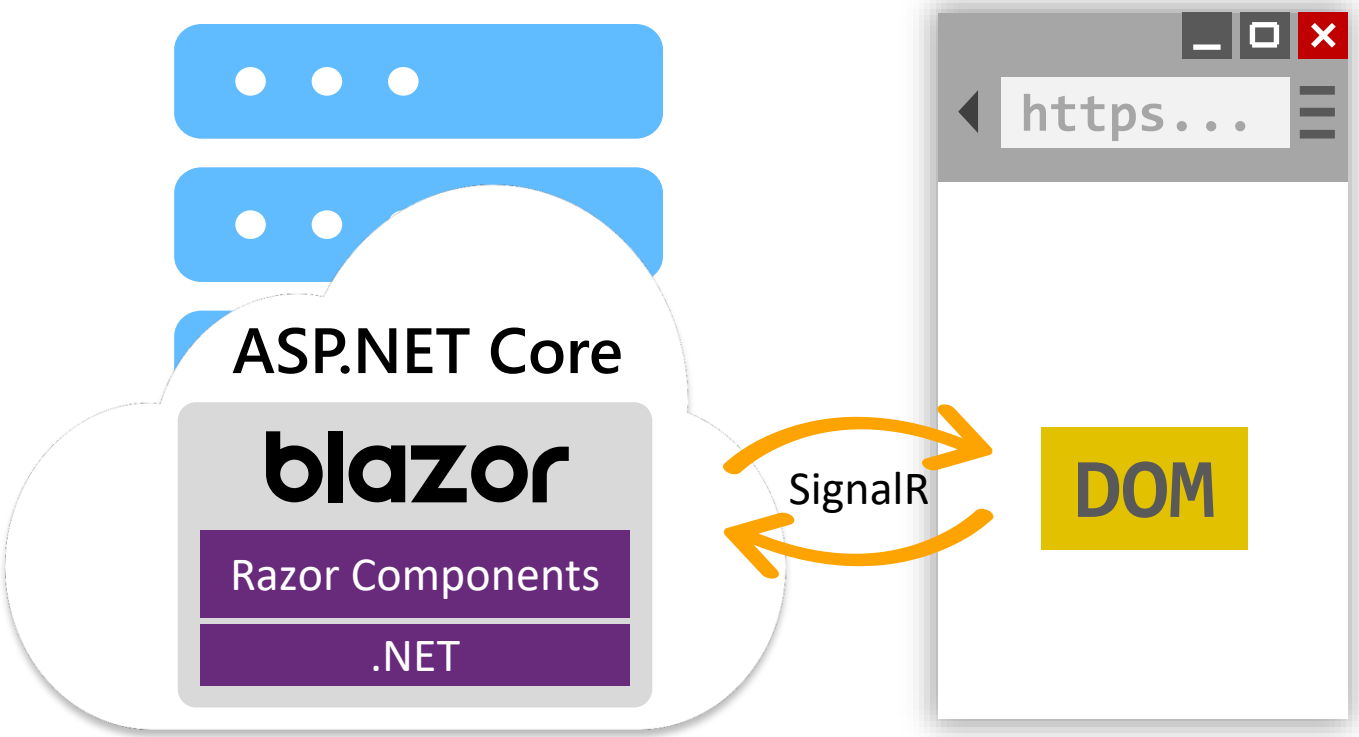
Hosting Modules

Blazor on client or server

Client-side



Server-side



Client-side

- 可執行全部現代化*的瀏覽器
- 不需要安裝 .NET 在伺服器
- SPA 使用者體驗
- 不支援舊型瀏覽器
- 首次下載容量較大 (有快取支援)
- 支援除錯

Server-side

- 首次下載容量較小
- 與 Server-side API 介接
- 完整除錯支援
- 沒有離線支援
- 網路延遲問題
- 擴展性問題

Microsoft STUDY4 Build School

Project Structure & Layout

開啟 Blazor 專案 – Visual Studio

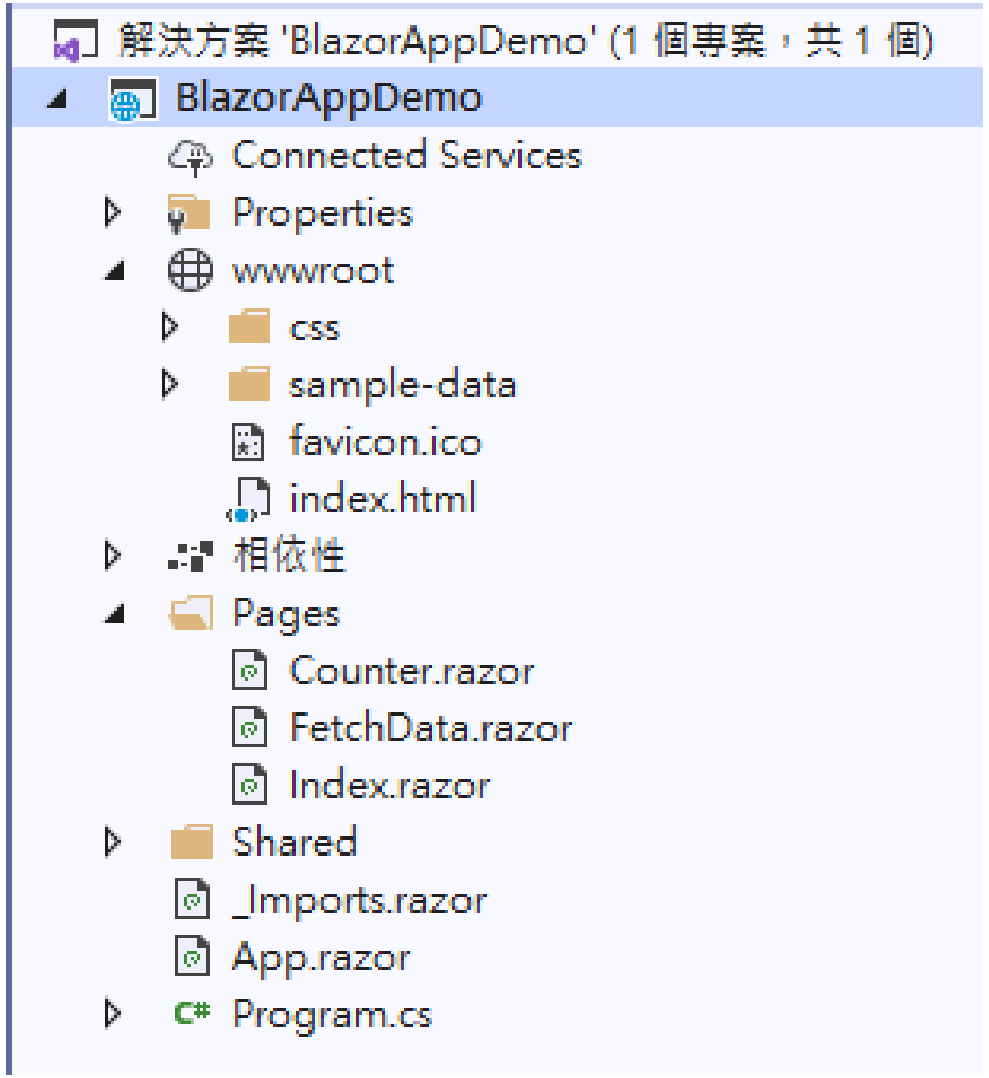
開啟 Blazor 專案 – Visual Studio

開啟 Blazor 專案 – Visual Studio Code

- 在本機開發電腦上，開啟終端機或命令提示字元視窗
- 在命令提示字元中，輸入 dotnet new blazorwasm -o [專案名稱]
- 移至 [專案名稱] 子資料夾 (cd [專案名稱])

註: Blazor Server 範本名稱為 blazorserver

開啟 Blazor 專案 – Visual Studio



- C# 與 Razor 檔案
- 類似 ASP.NET Core 專案架構
  - program.cs
  - wwwroot
  - *.razor (元件皆為區塊)
- 命名 - uppercase
- index.htm
  - Hosting page
  - HTML
  - 讀取 Blazor app
    - Blazor.webassembly.js

First component

Counter.razor

```razor
@page "/counter"

<h1>Counter</h1>

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

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

@code {
    private int currentCount = 0;

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

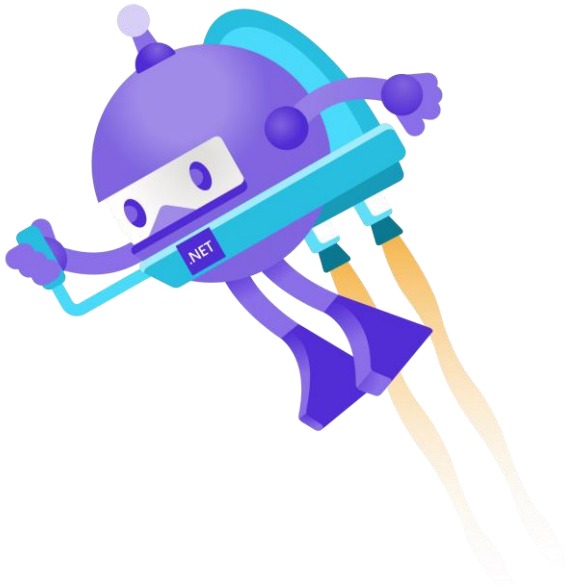
Using a component

```
@page "/"

<h1>Hello, world!</h1>
Welcome to your new app

<Counter />
```

Microsoft STUDY4 Build School

Demo
Project Structure

Duran Hsieh



Layout

BlazorApp 程式順序

1. Program.cs 內 builder.RootComponents.Add<App>("app");
2. App.razor (指向 MainLayout)
3. MainLayout (@body)
4. Index.razor (@page “/”)

Layout

BlazorServer 程式順序

1. Startup.cs 內 endpoints.MapFallbackToPage(“/_Host”);
2. _Host.cshtml
3. App.razor (指向 MainLayout)
4. MainLayout (@body)
5. Index.razor (@page “/”)

Layout

Startup.cs

```
BlazorDemo.Startup

0 references
public void Configure(IApplicationBuilder ap
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        // The default HSTS value is 30 days
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.UseRouting();

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

_Host.cshtml

```
@page "/"
@namespace BlazorDemo.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.Tag
@{
    Layout = null;
}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=devi
    <title>BlazorDemo</title>
    <base href="~/" />
    <link rel="stylesheet" href="css/bootstra
    <link href="css/site.css" rel="stylesheet
</head>
<body>
    <app>
        <component type="typeof(App)" render-mode="ServerPrerendered" />
    </app>
```

MainLayout.razor

```
@inherits LayoutComponentBase

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

<div class="main">
    <div class="top-row px-4">
        <a href="http://blazor.net" target="_blank" class="ml-md-auto">About</a>
    </div>

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

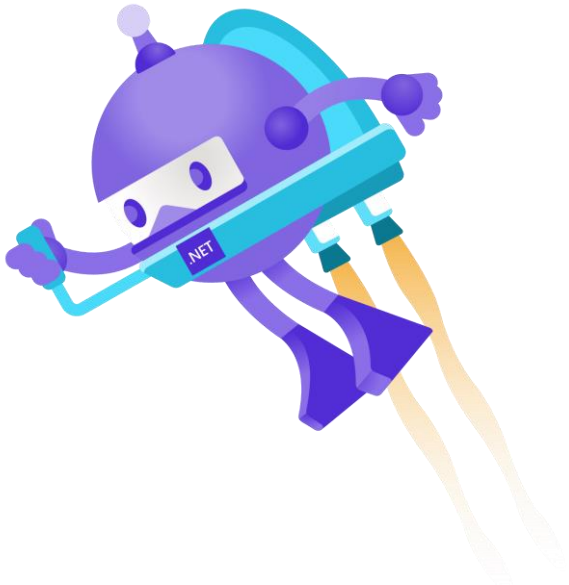
App.razor

```
<Router AppAssembly="@typeof(Program).Assembly">
    <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>
```

Microsoft STUDY4 Build School

Demo Layout

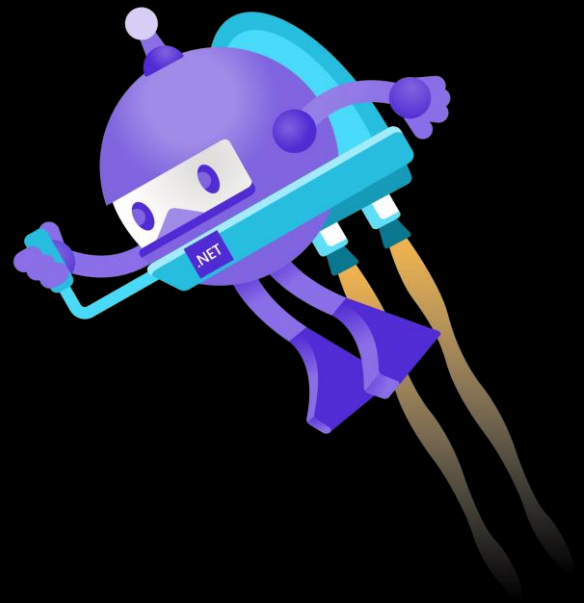
Duran Hsieh



Microsoft STUDY4 Build School

Lab 1. 相關安裝

Duran Hsieh



Microsoft STUDY4 Build School

Blazor Components

程式碼撰寫方式

- 混合模式 (Mixed approach)
  - 使用 @code
  - 舊版使用 @function
  - 建議使用於簡易邏輯或呈現邏輯

- Code Behind
  - 使用 partial 方式
  - 檔案名稱相同即可 (如 counter.razor 與 counter.cs)
  - 方便管理且簡顯易懂

Mixed approach

```
@page "/counter"

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
```

```
@code {
    private int currentCount = 0;

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

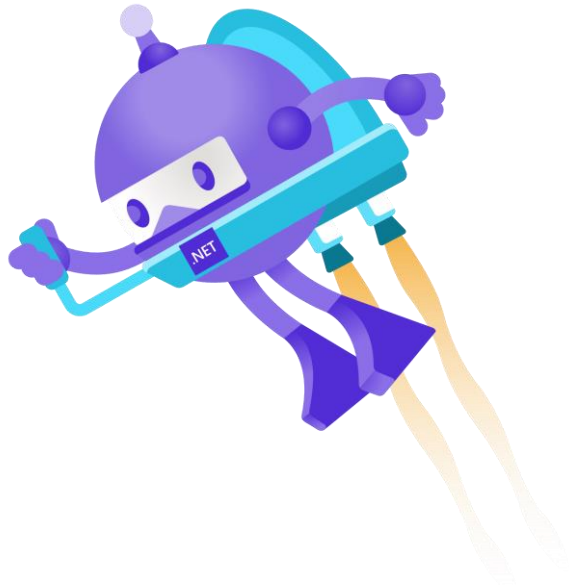
Using Partial Classes

```
public partial class Counter
{


}
```

Microsoft STUDY4 Build School

Demo
Using Partial Classes

Duran Hsieh

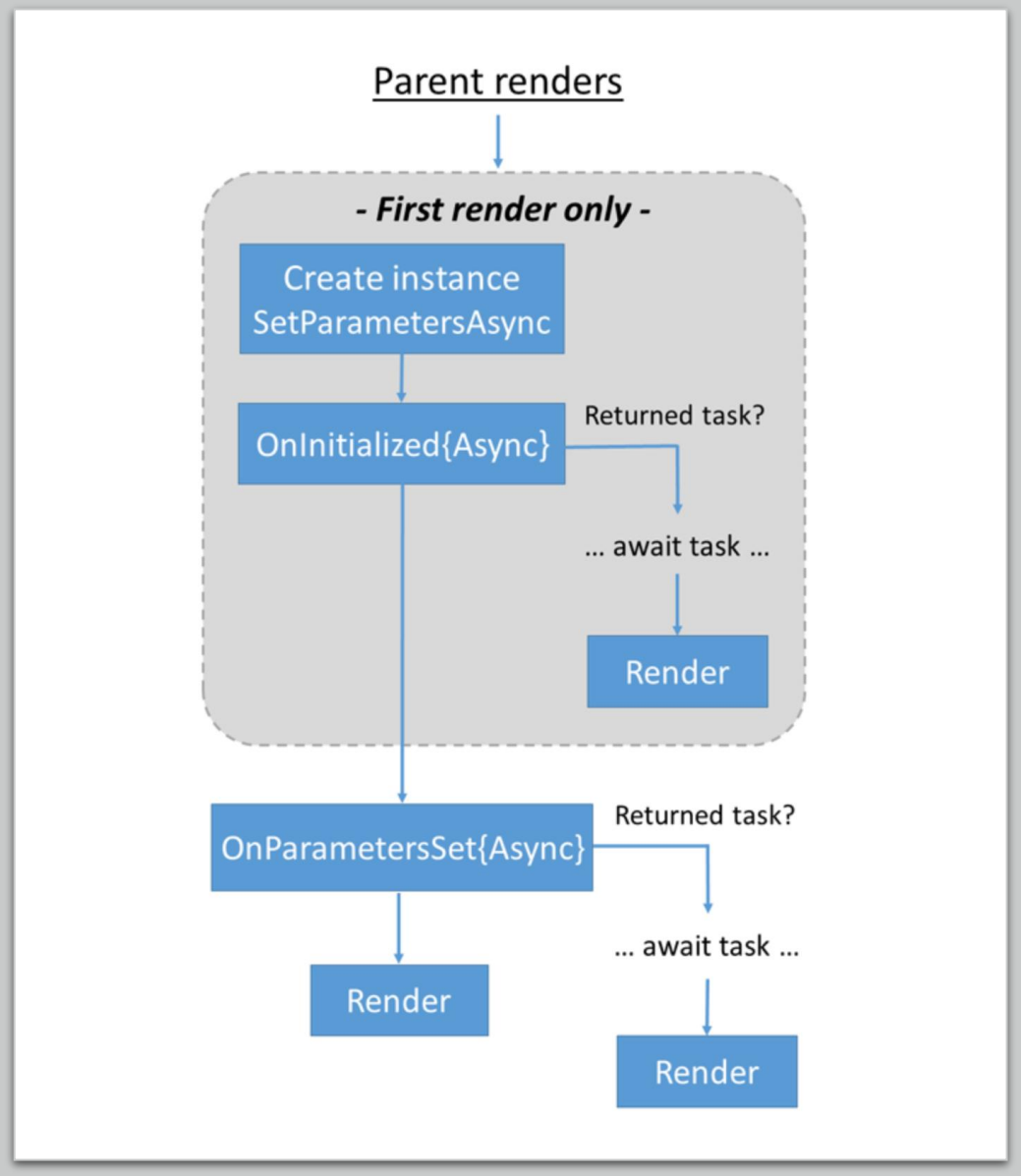


Component Lifecycle

OnInitialized
OnInitializedAsync

OnParametersSet
OnParametersSetAsync

OnAfterRender
OnAfterRenderAsync




```
protected override void OnInitialized()
{
    ...
}
```

```
protected override async Task OnInitializedAsync()
{
    await ...
}
```

Data Binding

- One-way
- Two-way
- Component parameter

One-way binding

```
<h1 class="page-title">
    @Employee.FirstName @Employee.LastName
</h1>
```

```
Public Employee Employee { get; set; }
```

Two-way binding

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

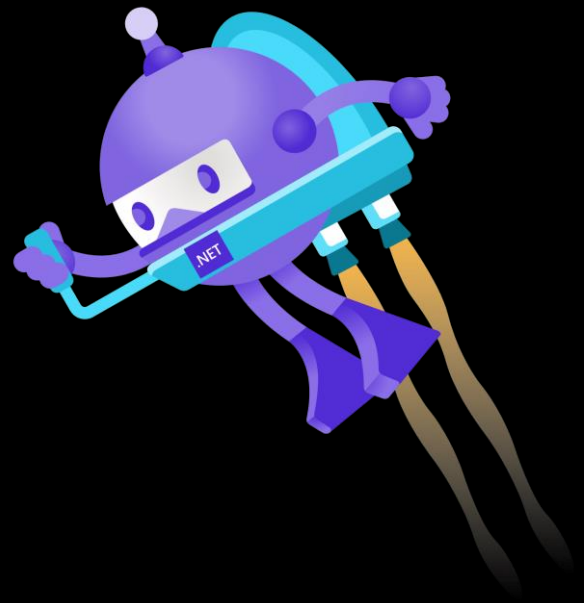
Two-way binding on different event

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

Microsoft STUDY4 Build School

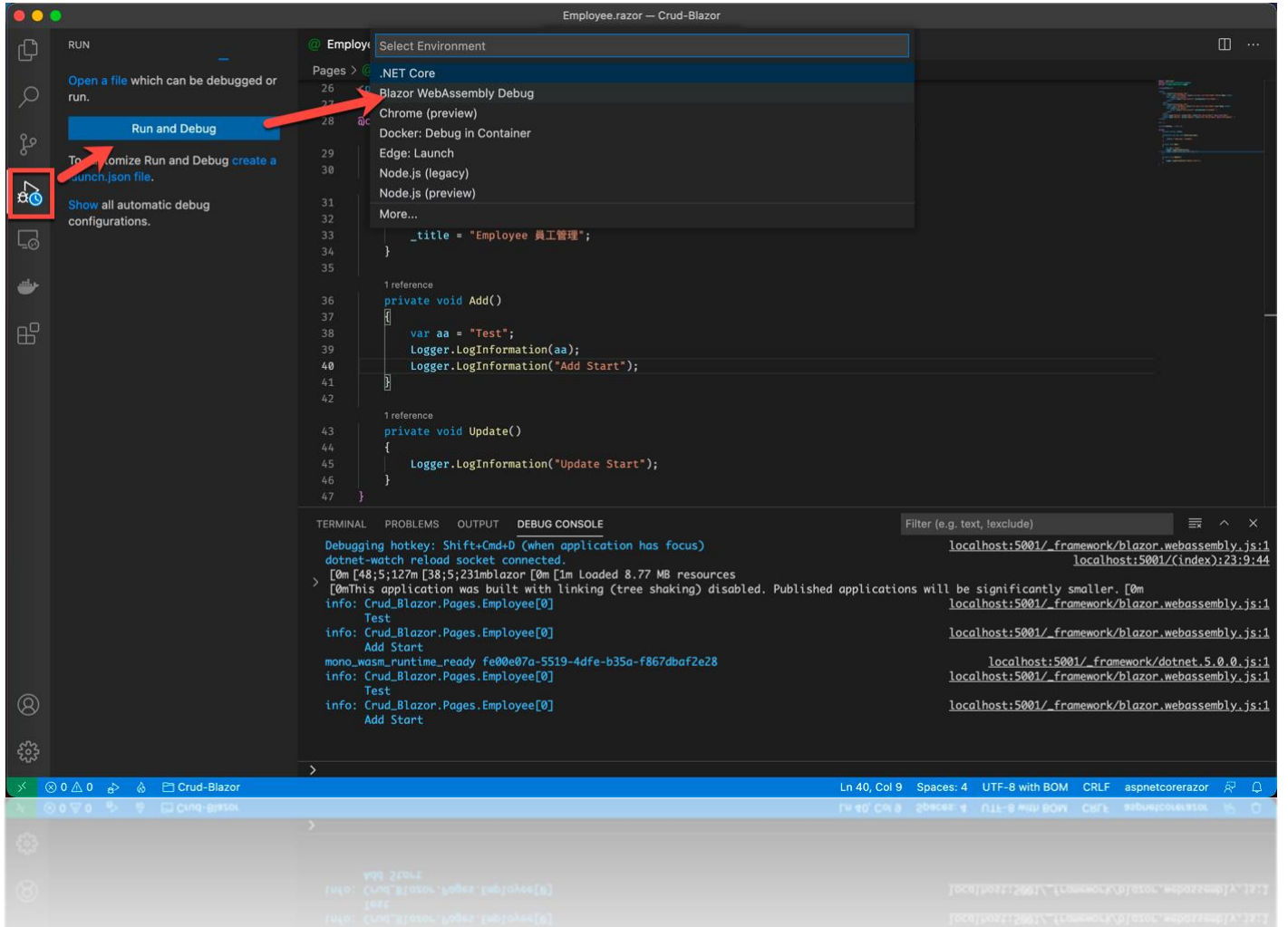
Lab 2. Event Binding

Duran Hsieh



Debug 方式

- 支援 Edge / Chrome
- 直接下中斷點
- 產生 launch.json 並調整



Microsoft STUDY4 Build School

Dependency Injection

Microsoft STUDY4 Build School

Using the HttpClient Service

```
Builder.Services.AddTransient(sp =>
    new HttpClient
    {
        BaseAddress = new Uri("http://api-endpoint")
    }
);
```

Microsoft STUDY4 Build School

Accessing the HttpClient in a component

```
[Inject]
Public HttpClient HttpClient { get; set; }
```

Microsoft STUDY4 Build School

JSON Helper Method

```
Protected override async Task OnInitializedAsync()
{
    Employee = await
        HttpClient.GetFromJsonAsnc<Employee[]>("api/employee");

}
```

Microsoft STUDY4 Build School

Using the HttpClientFactory

```
Builder.Services.AddHttpClient
    <IEmployeeDataService, EmployeeDataService>
    (
        client => client.BaseAddress = new Uri("http://localhost:55550/")
    );
```

Microsoft STUDY4 Build School

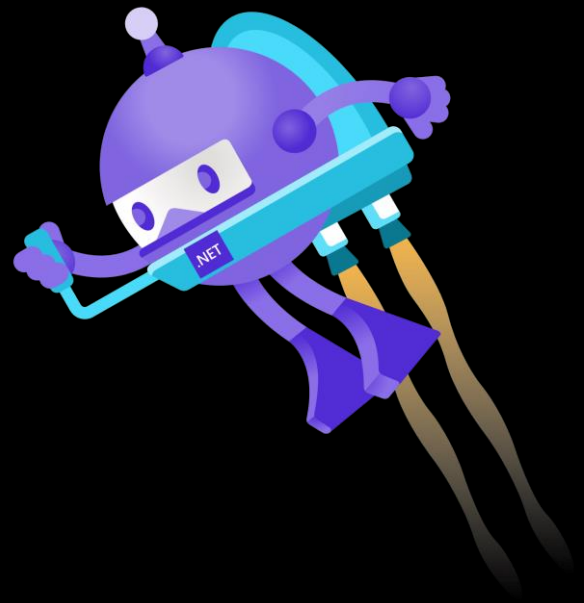
Constructor Injection in services

```
public class EmployeeDataService : IEmployeeDataService
{
    private readonly HttpClient _httpClient
    public EmployeeDataService(HttpClient httpClient)
    {
        _httpClient = httpClient;
    }
}
```

Microsoft STUDY4 Build School

Lab 3. Inject Service

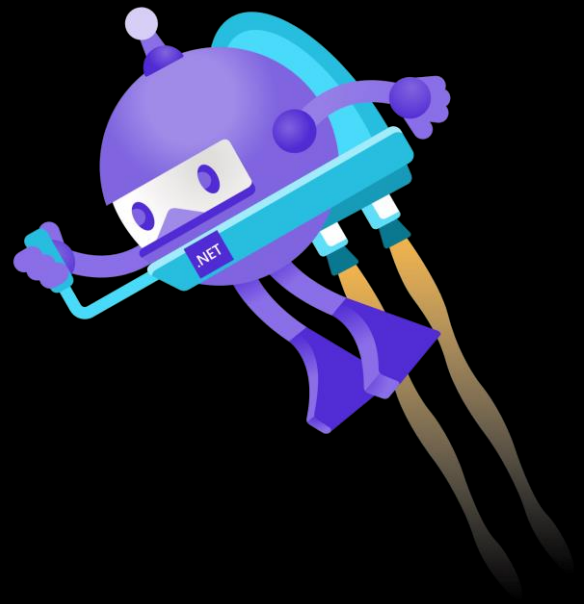
Duran Hsieh



Microsoft STUDY4 Build School

Lab 4. CRUD

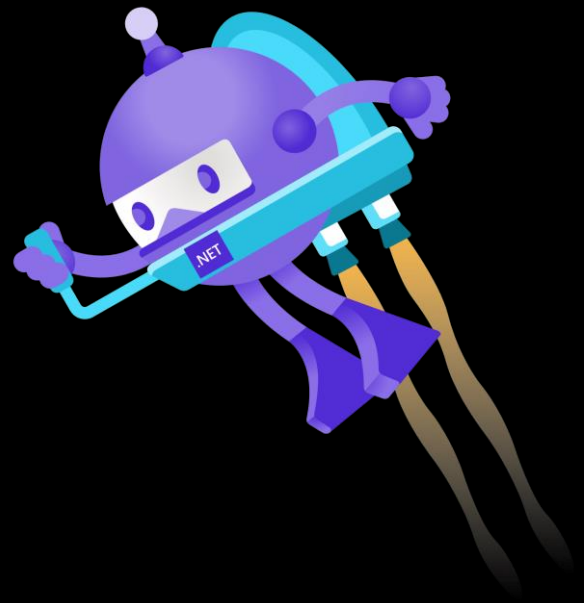
Duran Hsieh



Microsoft STUDY4 Build School

Lab 5. Components

Duran Hsieh



Microsoft STUDY4 Build School

Form in Blazor

EditForm

- Input components
  - InputText
  - InputTextArea
  - InputNumber
  - InputSelect
  - InputData
  - InputCheckbox
- Data binding
- Validation

EditForm

```
<EditForm Model="Employee"
    OnValidSubmit="@HandleValidSubmt"
    OnInvalidSubmit="@HandleInvalidSubmit">

    <InputText id="lastName"
        @bind-Value="@Employee.LastName"
        placeholder="Enter last time"
    </InputText >

</EditForm>
```

Validation

- 與 ASP.NET Core 驗證方式相同
- Data annotations
- DataAnnotationsValidator
- ValidationSummary

Validation

```
using System.ComponentModel.DataAnnotations;

public class ExampleModel
{
    [Required]
    [StringLength(10, ErrorMessage = "Name is too long.")]
    public string Name { get; set; }
}
```

Validation

```
<EditForm Model="@exampleModel" OnValidSubmit="@HandleValidSubmit">
    <DataAnnotationsValidator />
    <ValidationSummary />

    <InputText id="name" @bind-Value="@exampleModel.Name" />

    <button type="submit">Submit</button>
</EditForm>

@code {
    private ExampleModel exampleModel = new ExampleModel();

    private void HandleValidSubmit()
    {
        Console.WriteLine("OnValidSubmit");
    }
}
```

Microsoft STUDY4 Build School

JavaScript Interop

JavaScript Interop

- JavaScripts
  - 發展時間長久
  - 非所有事情皆能只透過 .NET 完成

- JavaScripts interop
  - 從 Blazor Code 呼叫 JavaScripts
  - Run on the Client
    - .NET Code 呼叫 JS
    - JS 呼叫 Code
    - 可以包進 library

JavaScript Interop

- 定義好 JavaScript Function

```
<script>
Windows.FunctionName = () => {
      // do something
}
</sctript>
```

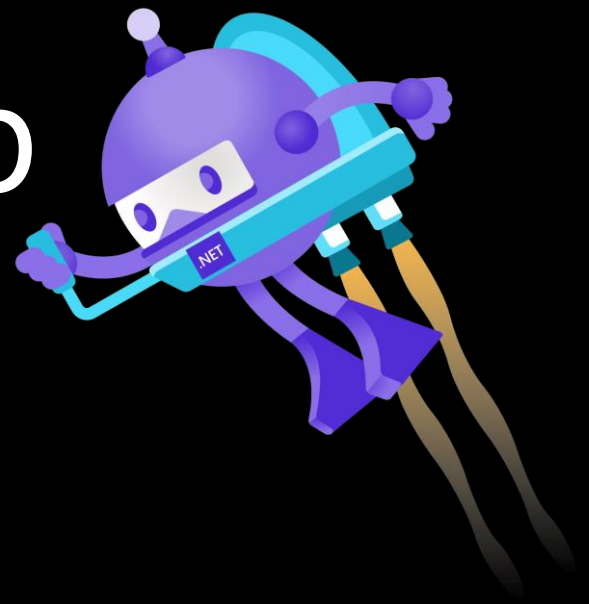
- 透過 JSRunTime 呼叫使用

```
await JSRuntime.InvokeAsync<object>
{
      “FunctionName”,
    “parameter”
}
```

Microsoft STUDY4 Build School

Lab 6. JavaScript Interop

Duran Hsieh



Microsoft STUDY4 Build School

Thanks for joining!

Ask questions on Twitter using #dotNETConf

.NET Conf 2020

特別感謝

91APP Technical Network

Red Hat

KKTIX

JET BRAINS

HackMD

MVP Microsoft® Most Valuable Professional

Microsoft

Build School

STUDY4 為學習而生

以及各位參與活動的你們