Microsoft STUDY4 Build School

Blazor Component 開發實戰

講師：Gelis

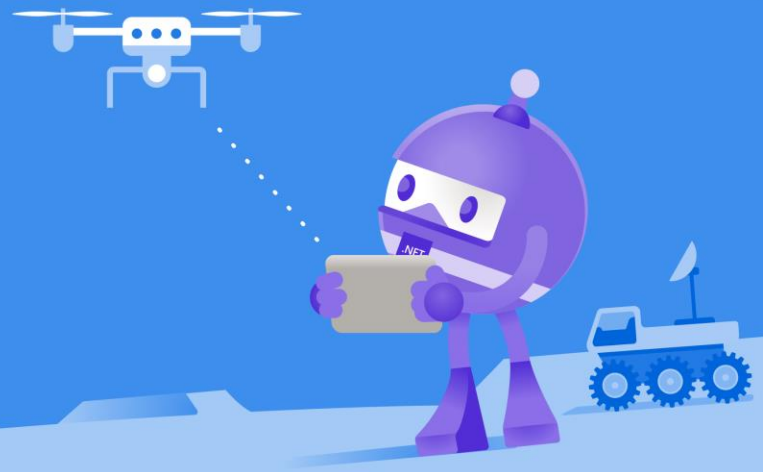
Blazor

不管你用了沒有？但我是用了！

Microsoft STUDY4 Build School

.NET Conf

探索 .NET 新世界



關於我

吳俊毅 Gelis - FB 軟體開發之路-經營者

- 部落格 (Gelis 技術隨筆)
  http://gelis-dotnet.blogspot.tw/
- FB 粉絲團(Gelis 的程式設計訓練營)
  https://www.facebook.com/gelis.dev.learning/?ref=bookmarks
- FB 社團 (軟體開發之路)
  https://www.facebook.com/groups/361804473860062/?ref=ts&fref=ts

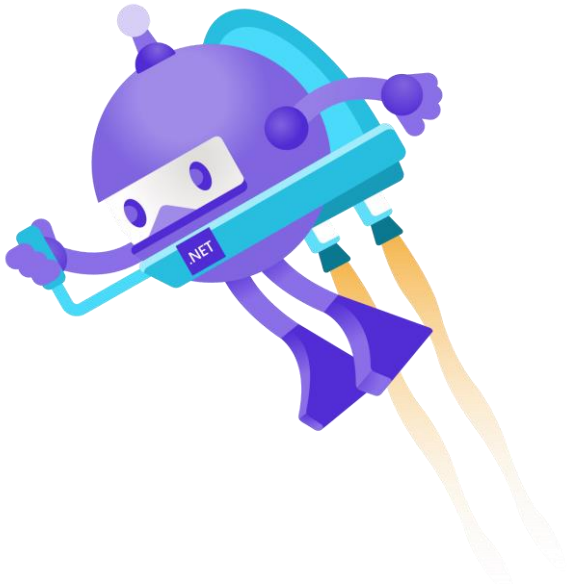
資深.NET技術顧問

Agenda

- Blazor 嶄新的開發模式
- Blazor Component vs. ASP.NET Web Form Web Controls
- Blazor Component 的生命週期
- Blazor Component 的狀態管理
- Blazor Component 開發基本介紹
- Blazor Component 的元件設計概念與實務
- Demo: 開發一個 Blazor Component 的 Button
- Demo: 開發一個 GridView Blazor Component

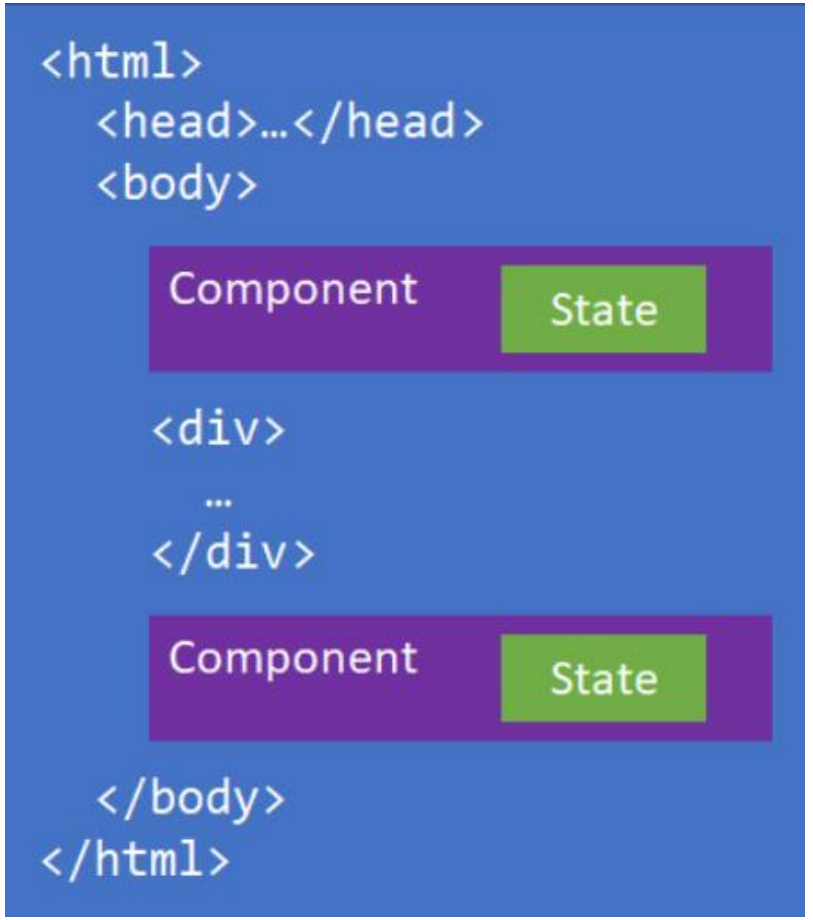
Microsoft STUDY4 Build School

Blazor 嶄新的開發模式



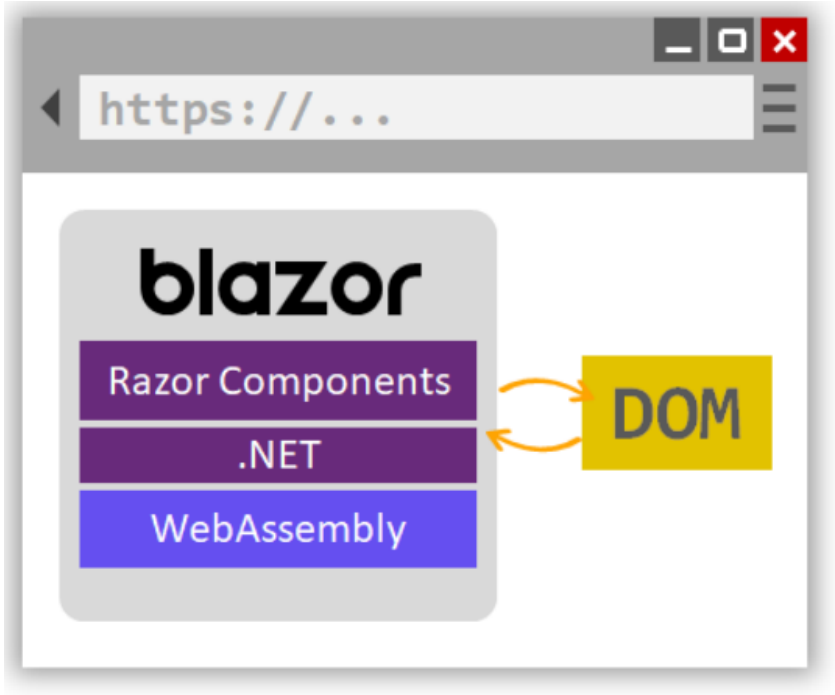
Blazor 嶄新的開發模式

- .NET Core 底下分支
  - Blazor
- View 就是個 Component
- 以 C# 取代 JavaScript
  - *Full-stack web development*
- WebAssembly
- 進化的 Code-Behind
- 兩種啟動模式：
  - Blazor WebAssembly
  - Blazor Server (SignalR)

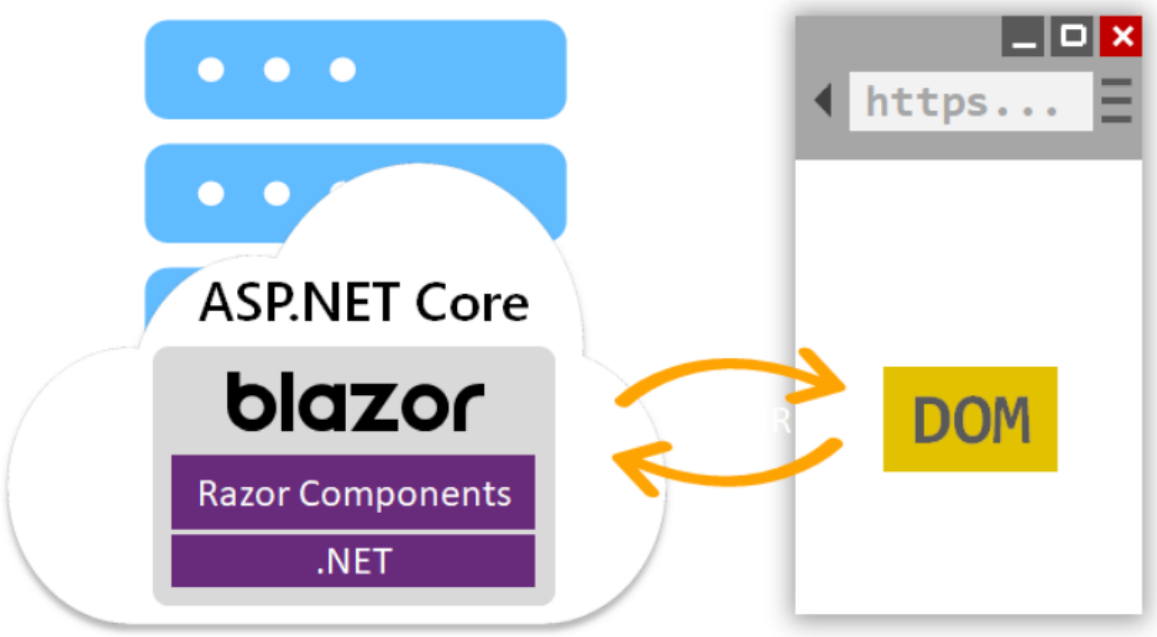


Blazor 兩種啟動模式

Blazor WebAssembly



Blazor Server



Tag Helper vs. Blazor Component

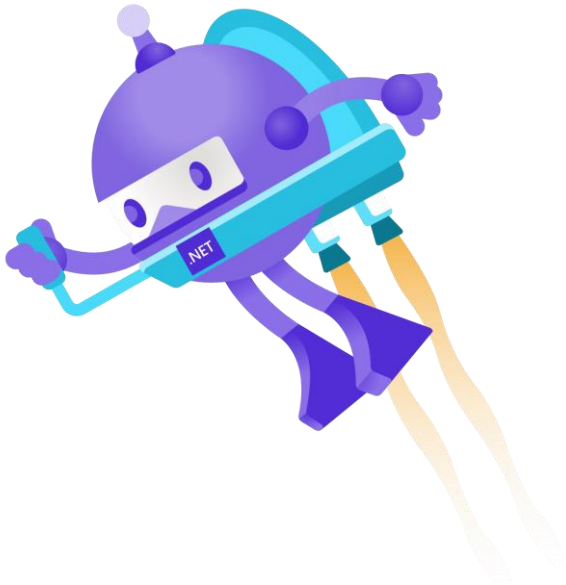
Microsoft.AspNetCore.Mvc.TagHelper

- 副檔名：.cshtml
- HTML 渲染
  - 只能與前端 HTML 互動
  - 沒有伺服器完整生命週期

Blazor Component

- 副檔名：.razor
- *Server Side 渲染*
  - 具備*完整*伺服器生命週期
  - 使用 C# 取代 JavaScript 建立互動式 Web UI

Blazor Component vs. Web Form WebControls



Blazor 的 Code-Behind (In-Line)

```
@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++;
      }
 }
```

Web Form 的 Code-Behind (In-Line)

```
<%@ Page Language="C#" AutoEventWireup="true" %>
 <!DOCTYPE html>
 <html xmlns="http://www.w3.org/1999/xhtml">
 <head runat="server">
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
     <title></title>
 </head>
 <body>
     <form id="form1" runat="server">
         <div>
            <asp:Button ID="Button1" runat="server" Text="Click Me" OnClick="Button1_Click"/>
         </div>
     </form>
 </body>
 </html>
 <script runat="server">
     void Page_Load()
     {
         if(!IsPostBack){  }
     }
     private int _count { get; set; }
     protected void Button1_Click(object sender, EventArgs e)
     {

     }
 </script>
```

Blazor 與 Web Form 的 Code-Behind 比較

Blazor Code-Behind

UI:

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

Code:

```
public partial class Counter: ComponentBase
{
     private int currentCount = 0;
     private void IncrementCount()
     {
         currentCount++;
     }
}
```

Web Form Code-Behind

UI:

```
<%@ Page Language="C#"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
     <form id="form1" runat="server">
         <div>
         </div>
     </form>
 </body>
 </html>
```

Code:

```
public partial class Default : System.Web.UI.Page
{
     protected void Page_Load(object sender, EventArgs e)
     {
     }
 }
```

Blazor Component vs. WebCotrols

Blazor Component

```
public class MyButton2: ComponentBase
{
    [Parameter]
    1 個參考
    public string ButtonText { get; set; }
    12 個參考
    protected override void BuildRenderTree(RenderTreeBuilder builder)
    {
        base.BuildRenderTree(builder);

        builder.AddMarkupContent(0, $"<button class=\"btn btn-primary\">{ButtonText}</button>");
    }

    1 個參考
    protected override Task OnParametersSetAsync()
    {
        return base.OnParametersSetAsync();
    }

    3 個參考
    protected override void OnParametersSet()
    {
        base.OnParametersSet();
    }

    2 個參考
    protected override void OnInitialized()
    {
        base.OnInitialized();
    }
}
```

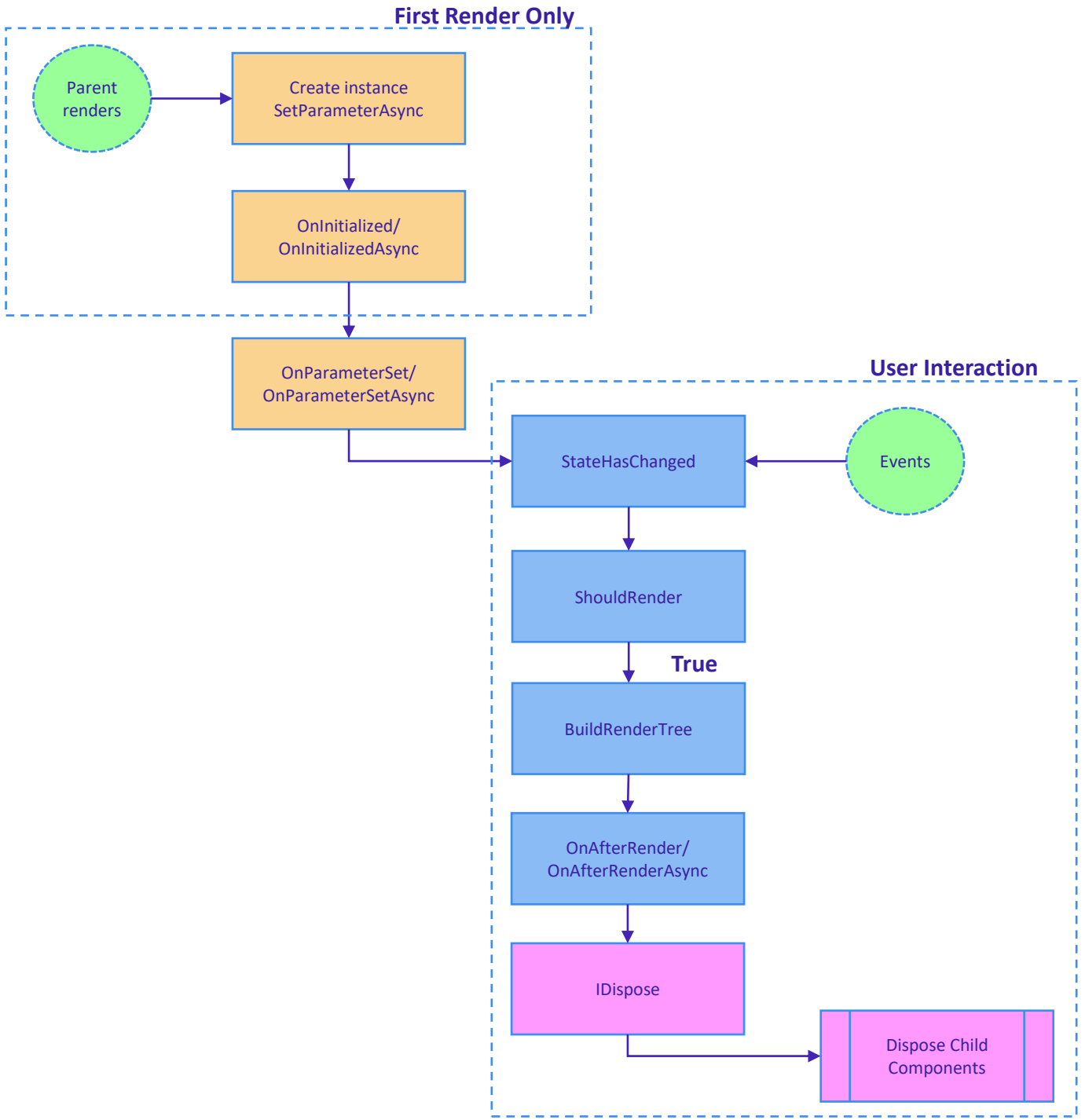
Web Control: DatePicker

```
1 個參考
public class WebDatePicker : System.Web.UI.WebControls.TextBox
{
    private const string W_SCRIPT_INITIAL_CALENDAR = "_ARI_WEBDATAPICKER";

    [Bindable(true)]
    [Browsable(true)]
    [Category("圖片")]
    [DefaultValue("")]
    [Description("觸發按鈕顯示影像的 URL")]
    [Editor("System.Web.UI.Design.ImageUrlEditor, System.Design, , Culture=neutral, PublicKeyToken=b03f5f7f11d
    1 個參考
    public string ButtonImageUrl...

    Others Properties
    0 個參考
    public WebDatePicker()...

    3 個參考
    protected override void OnPreRender(EventArgs e)...
```

```
    3 個參考
    protected override void Render(HtmlTextWriter output)
    {
        object[] item;
        string str = null;
        string str1 = null;
        str = string.Format("<TABLE Border=0 cellpadding=0 cellspacing=0 STYLE=\"POSITION:{0};LEFT:{1};TOP:{2
        output.Write(str);
        base.Render(output);
        str1 = (this.MouseOverImageUrl == null
            || this.MouseOverImageUrl.Length <= 0 ? string.Concat("<img src=\"{0}\" border=\"0\" onclick=\"act
        item = new object[] { this.ButtonImageUrl, this.MouseOverImageUrl };
        str1 = string.Format(str1, item);
        output.Write(str1);
        output.Write("</TD></TR></TABLE>");
    }
}
```

Microsoft STUDY4 Build School

Blazor Component 的生命週期



Blazor Component 的生命週期

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

- 當元件要開始初始化（*Create Instance*）前會先執行『屬性直插入』的事件
- 您也可以在這個事件裡取得所有傳遞至元件內的 Parameters

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

- 元件的初始化方法
- 只有第一次會呼叫
  - @page
- 在 render-mode="ServerPrerendered" 的時候預設會叫用兩次方法
- *會**先**執行同步方法、後執行非同步方法*

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

- 在元件被初始化 Initialized 產生新的實體時執行
- 一樣都是**先**執行同步方法、後執行非同步方法

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

- 元件完成轉譯之後，會呼叫的方法
- 通常可以使用此階段，利用轉譯的內容來執行其他如：
  - 啟用在轉譯的 DOM 專案上操作的協力廠商 JavaScript 程式庫
- *在元件第一次完成轉譯並渲染至前端後永遠為 False*、因此判斷 OnAfterRender(bool:firstRender=false) 可確保一次性的初始化工作，避免無窮迴圈

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

- 避免 UI 重新整理
- 呼叫 BuildRenderTree() 以重新渲染 UI 到前端
- 預設傳回 true，可複寫為 False 以讓不需與 Server 互動的靜態元件再次渲染至前端以增加 UI 效能

Blazor Component 的生命週期

- Lifecycle Methods
  - SetParameterAsync
  - OnInitialized
  - OnInitializedAsync
  - OnParametersSet
  - OnParametersSetAsync
  - OnAfterRender
  - OnAfterRenderAsync
  - ShouldRender
  - IDisposable

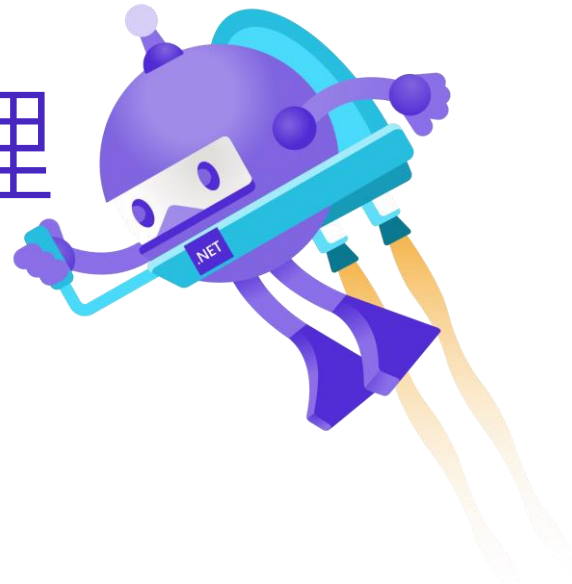
```
@using System
@implements IDisposable
...
@code {
    public void Dispose()
    {
        ...
    }
}
```

◆*頁面上有注入的元件建議實作 Dispose 方法加以釋放*

◆頁面可使用 @implements 關鍵字來實作介面

Microsoft STUDY4 Build School

Blazor Component 的狀態管理



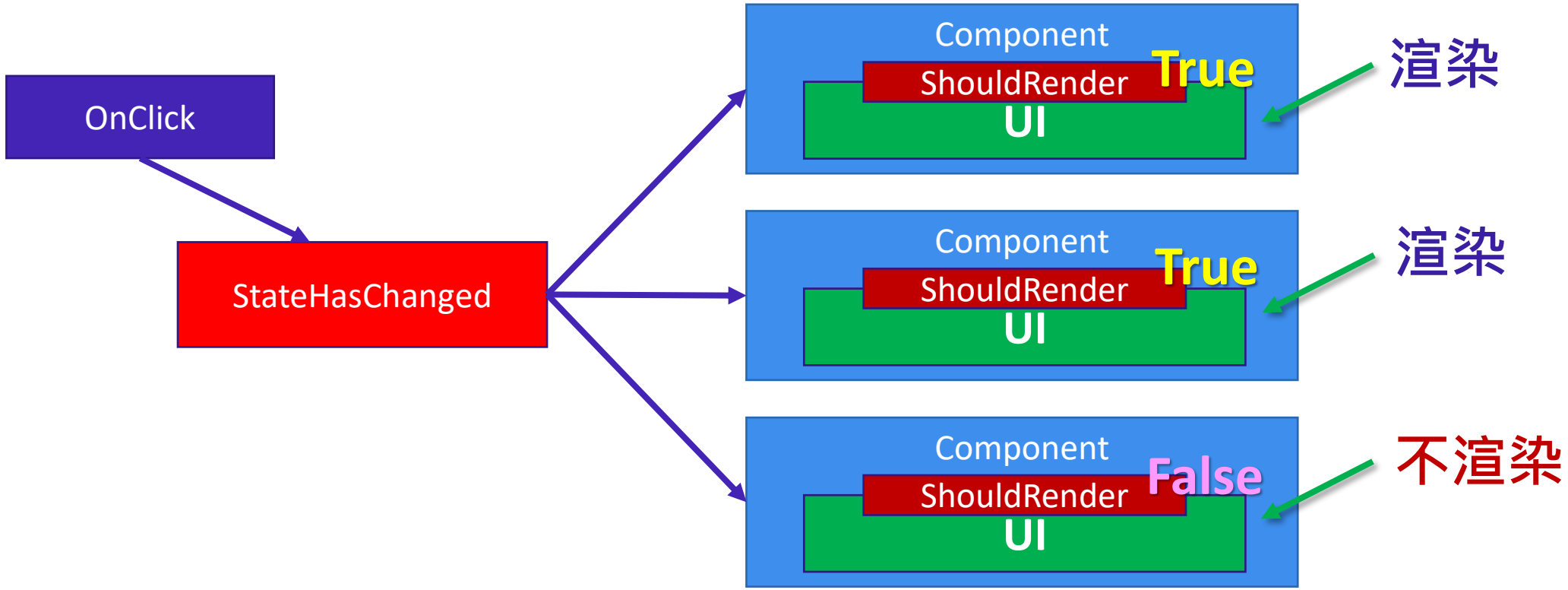
ShouldRender 避免 UI 重新整理

- 每次轉譯元件時就會呼叫
- 預設回傳 true 便自動呼叫 BuildRenderTree() 方法
- 可複寫 override 為回傳 false 避免 Blazor Component 呼叫 BuildRenderTree() 方法*再次*渲染前端而可以增加 UI 的效能

```
protected override bool ShouldRender()
{
        return true;
 }
```

StateHasChanged 狀態變更

- 一般來說，只要在 UI 觸發任一個 OnClick 事件便會自動調用 StateHasChanged() 方法以通知 UI 進行變更



Blazor Component 的狀態保存

- Blazor 的狀態保存：
  - 資料庫
  - 協力廠商支援 Cache（ Redis, Others… ）
  - 保存記憶體內容狀態 (.NET Core)
    - Server Blazor ➔ Startup.cs (Singleton)
    - Client Web Assembly ➔ program.cs
  - Browser Local Storage
  - URL

Blazor Component 的狀態保存

- Blazor 的狀態保存：
  - 資料庫
  - 協力廠商支援 Cache（Redis, Others…）
  - 保存記憶體內容狀態 (.NET Core)
    - Server Blazor ➔ Startup.cs (Singleton)
    - Client Web Assembly ➔ program.cs
  - Browser Local Storage
  - URL

保存記憶體內容狀態 (.NET Core)

1

```
public class CounterModel
{
    2 個參考
    public int Counter { get; set; }
}
```

2

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

3

Counter.razor

```
@page "/counter"
@using BlazorGridTestServerApp1.Models
@using System.Diagnostics
@inject CounterModel counter

<h1>Counter</h1>

<p>Current count: @counter.Counter</p>

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

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        counter.Counter++;
        //currentCount++;
    }
```

Blazor Component 的狀態保存

- Blazor 的狀態保存：
  - 資料庫
  - 協力廠商支援 Cache（ Redis, Others… ）
  - 保存記憶體內容狀態 (.NET Core)
    - Server Blazor ➔ Startup.cs (Singleton)
    - Client Web Assembly ➔ program.cs
  - Browser Local Storage
  - URL

Blazor Component 的狀態保存

- Browser Local Storage

```
window.stateMeneger = {
        save: function (key, str) {
                localStorage[key] = str;
        },
        load: function (key) {
                return localStorage[key];
        }
}
```

Microsoft STUDY4 Build School

Blazor Component 開發基本介紹

Blazor 開發基本介紹

```
@page "/"
@using BlazorGridTestServerApp1.Pages

<h1>Hello, world!</h1>

Welcome to your new app.

<SurveyPrompt Title="How is Blazor working for you?" />

<GridTest></GridTest>

<Counter></Counter>
```

Blazor 開發基本介紹

Blazor 的 Route 設定 (App.razor)

App.razor | _ViewStart.cshtml | _Layout.cshtml | MainLayout.razor | _Host.cshtml | Startu

```
1   <Router AppAssembly="@typeof(Program).Assembly">
2       <Found Context="routeData">
3           <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
4       </Found>
5       <NotFound>
6           <LayoutView Layout="@typeof(MainLayout)">
7               <p>Sorry, there's nothing at this address.</p>
8           </LayoutView>
9       </NotFound>
10  </Router>
11
```

版面設置（_Host.cshtml & App.razor & MainLayout.razor）

```
8     <!DOCTYPE html>
9     <html lang="en">
10    <head>
11        <meta charset="utf-8" />
12        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
13        <title>BlazorGridTestServerApp1</title>
14        <base href="~/" />
15        <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
16        <link href="css/site.css" rel="stylesheet" />
17        <link href="BlazorGridTestServerApp1.styles.css" rel="stylesheet" />
18    </head>
19    <body>
20        <component type="typeof(App)" render-mode="ServerPrerendered" />
21
22        <div id="blazor-error-ui">
23            <environment include="Stag
24                An error has occurred. Th
25            </environment>
26            <environment include="Developmen
27                An unhandled exception has occu
```

MainLayout.razor | _ViewStart.cshtml | _Layout.cshtml | _Host.cshtml | Startup

```
@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>
```

App.razor | _ViewStart.cshtml | _Layout.cshtml | MainLayout.razor | _H...cshtml | Startu

```
1   <Router AppAssembly="@typeof(Program).Assembly">
2       <Found Context="routeData">
3           <RouteView RouteData="@routeData" DefaultLayout="@typeof(MainLayout)" />
4       </Found>
5       <NotFound>
6           <LayoutView Layout="@typeof(MainLayout)">
7               <p>Sorry, there's nothing at this address.</p>
8           </LayoutView>
9       </NotFound>
10  </Router>
11
```

Blazor Component 與 TagHelper 曖昧關係

- 提供互動功能的 TagHelper ➔ ComponentTagHelper
- Server-Side 完整生命週期
- Signal-R 協定

```
 8    <!DOCTYPE html>
 9    <html lang="en">
10    <head>
11        <meta charset="utf-8" />
12        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
13        <title>BlazorGridTestServerApp1</title>
14        <base href="~/" />
15        <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
16        <link href="css/site.css" rel="stylesheet" />
17        <link href="BlazorGridTestServerApp1.styles.css" rel="stylesheet" />
18    </head>
19    <body>
20        <component type="typeof(App)" render-mode="ServerPrerendered" />
21
22        <div id="blazor-error-ui">
23            <environment include="Staging,Production">
24                An error has occurred. This application may no longer respond until reloaded.
25            </environment>
26            <environment include="Development">
27                An unhandled exception has occurred. See browser dev tools for details.
```

在 Razor View (cshtml) 中顯示 Blazor 元件

1. 加入 _Layout.cshtml
2. 加入 _ViewImport.cshtml
3. 加入 _ViewStart.cshtml

```
@using BlazorGridTestServerApp1
@namespace BlazorGridTestServerApp1.Pages
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

在 Razor View (cshtml) 中顯示 Blazor 元件

1. 加入 _Layout.cshtml
2. 加入 _ViewImport.cshtml
3. 加入 _ViewStart.cshtml

```
<script src="~/_framework/blazor.server.js" autostart="false"></script>
 <script>
     Blazor.start({
         configureSignalR: function (builder) {
             builder.withUrl('/_blazor' );
         }
     });
 </script>
```

在 Razor View (cshtml) 中顯示 Blazor 元件

1. 加入 _Layout.cshtml
2. 加入 _ViewImport.cshtml
3. 加入 _ViewStart.cshtml
4. 加入一個 View 檢視 .cshtml 頁面

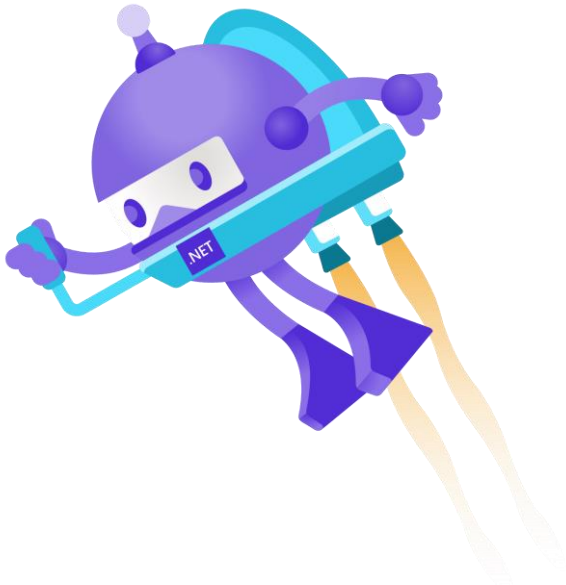
```
1     @page
2     @using BlazorGridTestServerApp1.Pages
3     @using BlazorGridTestServerApp1.Shared
4     @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
5
6     @model BlazorGridTestServerApp1.Pages.MyRenderComtModel
7     @{
8     }
9
10    <h2>My Test Render Component</h2>
11
12    <component>@(await Html.RenderComponentAsync<Counter>(RenderMode.Server))</component>
13
```

Demo : <component/>

Layout 的（階層 / 順序）

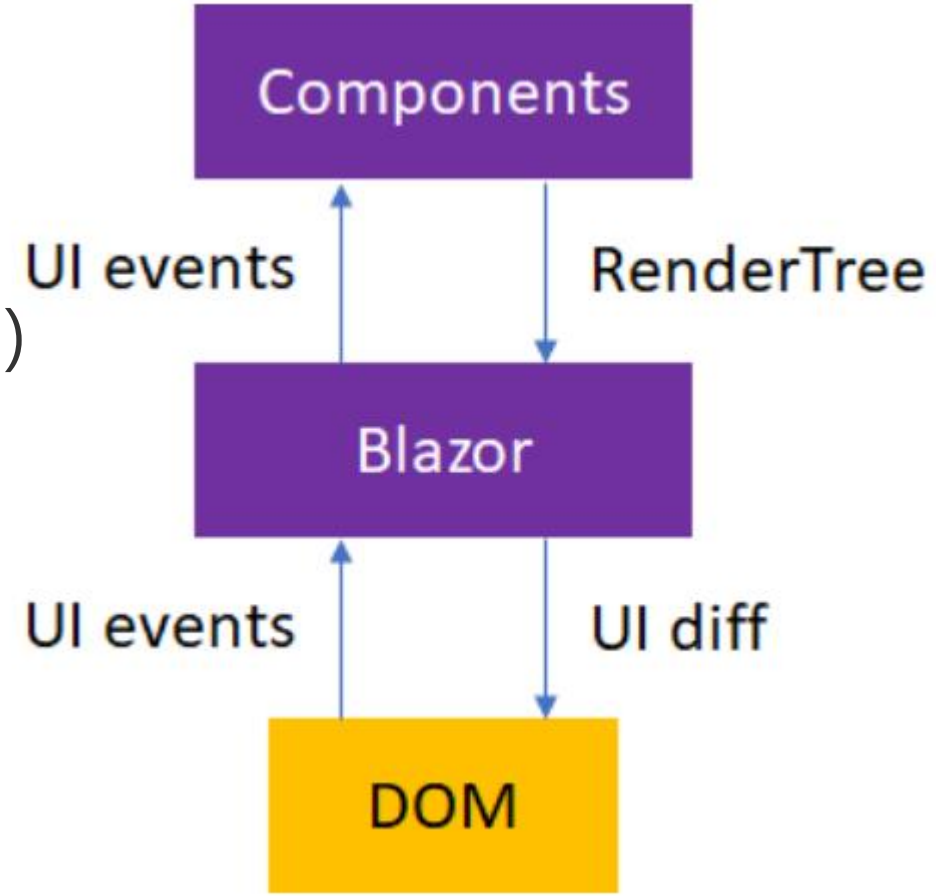
Microsoft STUDY4 Build School

Blazor Component 的元件設計概念與實務



Blazor Server (.NET 5)

- 全端開發
  - Reuse CSS/Bootstrap/Others…
  - 可與原有 Core MVC 共存
  - Reuse DataAnnotations (Validation)
  - *支援雙向繫結 (Two-way Binding)*
  - 完整 Server Side 生命週期
  - 部分概念沿用 WebForm 架構



Blazor Component Parameter

- 傳遞資訊到 Component 的方法
- 套用 [Parameter] 標籤的公開 Property 才可以透過 Parent Component 傳遞資訊到子 Component 中

```
<strong>@Title</strong>

@code {
    // Demonstrates how a parent component can supply parameters
    [Parameter]
    public string Title { get; set; }
}
```

Blazor Component Event Handler

- Component 的 Event 事件的傳遞
  - 父元件可透過 EventCallback 將事件傳遞給子元件

子元件

```
[Parameter]
public EventCallback<MouseEventArgs> OnClickCallback { get; set; }
```

父元件

```
<MyButton ButtonText=“我是按鈕" OnClickCallback="@Button_Click"></MyButton>

@code {
  protected void Button_Click(MouseEventArgs e)
  {
  }
}
```

Blazor Component (Class Control)

- None Code-Behind
  - 開發方式與用途：
    - *繼承 ComponentBase 容易設計為底層元件設計 Button, TextBox, GridView...*
    - 不透過 .Razor 渲染、透過實作 BuildRenderTree() 方法渲染、單一檔案 .cs 即可實作

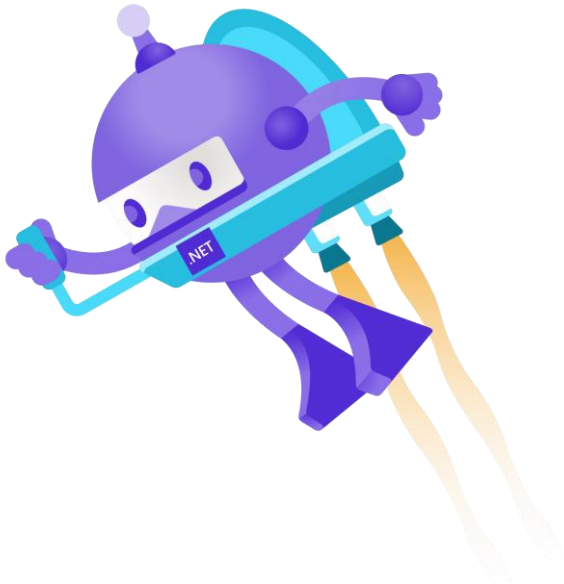
```
public class MyButton: ComponentBase
{
     [Parameter]
     public string ButtonText { get; set; }
     [Parameter]
     public EventCallback<MouseEventArgs> OnClickCallback { get; set; }
     protected override void BuildRenderTree(RenderTreeBuilder builder)
     {
         base.BuildRenderTree(builder);
         builder.OpenElement(5, "button");
         builder.AddAttribute(6, "class", "btn btn-primary");
         builder.AddAttribute(7, "onclick", EventCallback.Factory.Create<MouseEventArgs>(this, Button_Click));
         builder.AddContent(8, ButtonText);
         builder.CloseElement();
     }
}
```

Blazor Component Parameter (雙向繫結)

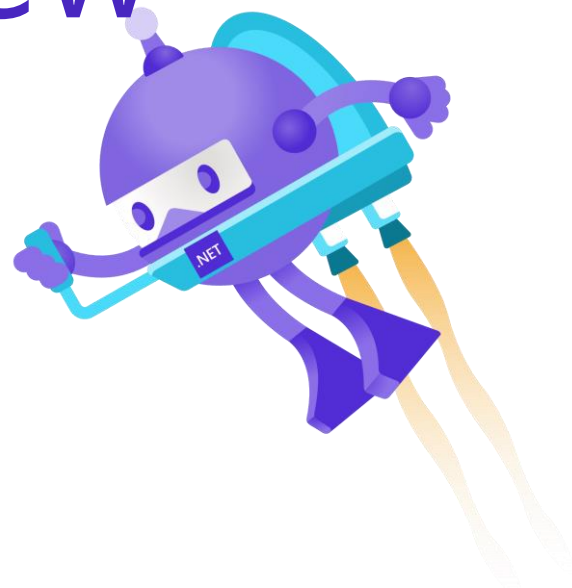
- Blazor Component 透過 @bind 提供（欄位、屬性或運算式值命名的 HTML 元素屬性）的資料雙向繫結功能

```
<h3>TwowayBinding</h3>
 <p>
     <input @bind="CurrentValue" /> Current value: @CurrentValue
 </p>

 @code {
     private string CurrentValue { get; set; }
 }
```

Demo: 開發一個 Blazor Component 的 Button



Demo: 開發一個 GridView Blazor Component



總結：

- 重複使用原本 C# 與 MVC 的 Skill
- 開發可重複使用的 Components 資源
- 考量企業端環境抉擇使用 Server 或 Client WebAssembly 來開發
- 廣大的 .NET Core 生態支援
- 還要想什麼？就開始用就對了！

Microsoft STUDY4 Build School

感謝聆聽!

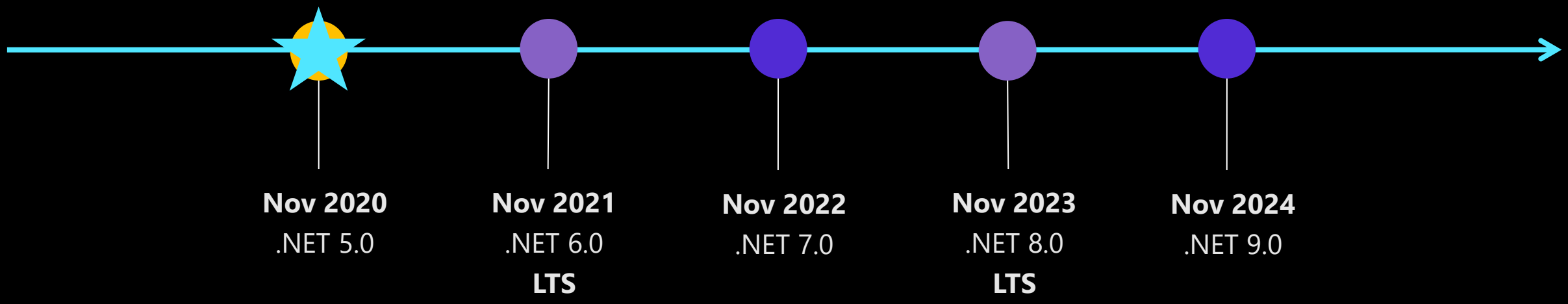
- 關於我：
- FB 社團 (軟體開發之路)

  https://www.facebook.com/groups/361804473860062/?ref=ts&fref=ts

Microsoft STUDY4 Build School

Q & A

.NET Schedule



- .NET 5.0 released today!
- Major releases every year in November
- LTS for even numbered releases
- Predictable schedule, minor releases as needed

.NET Conf 2020

特別感謝

91APP Technical Network

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