# Blazor Render Modes

Tim Purdum

Blazor Day

September 25, 2025
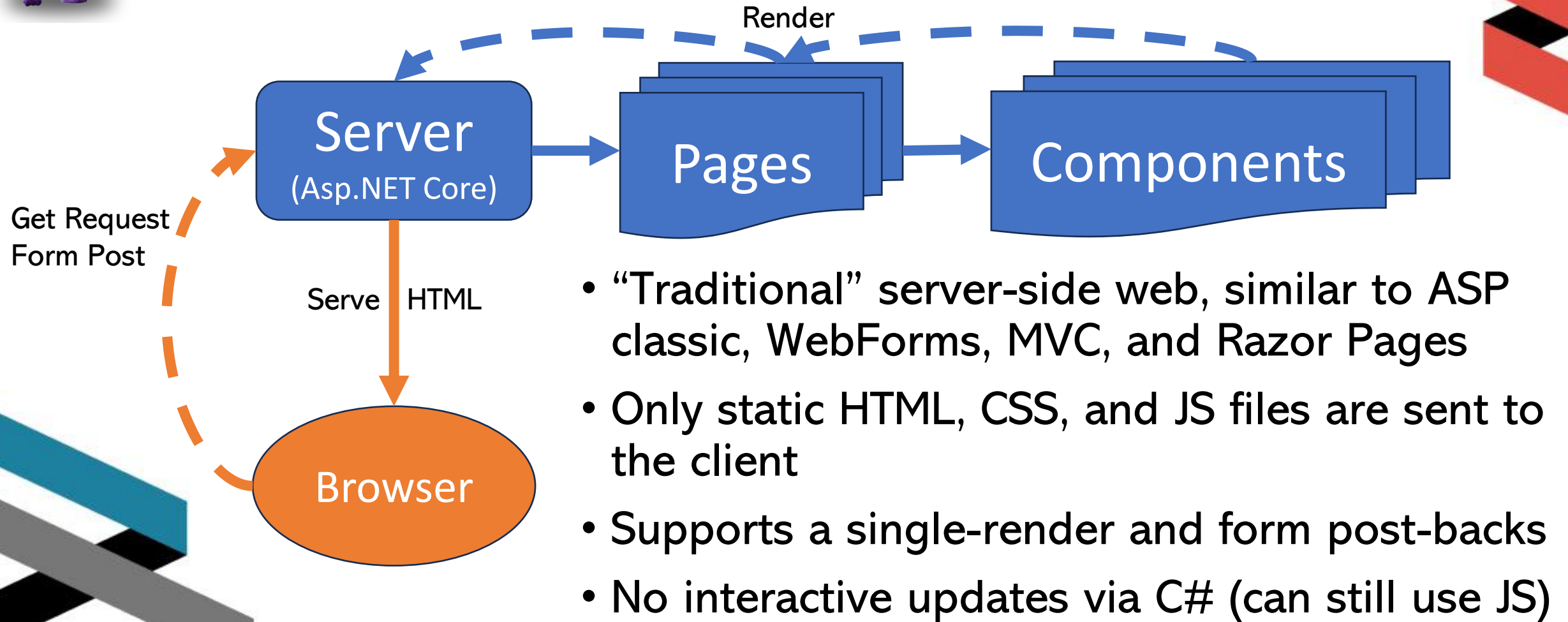
# Blazor Component Render Modes



Blazor Hybrid (MAUI)

- Static Server Mode
- Interactive Server Mode
- Interactive WebAssembly Mode
- Interactive Auto Mode
- Blazor Hybrid *
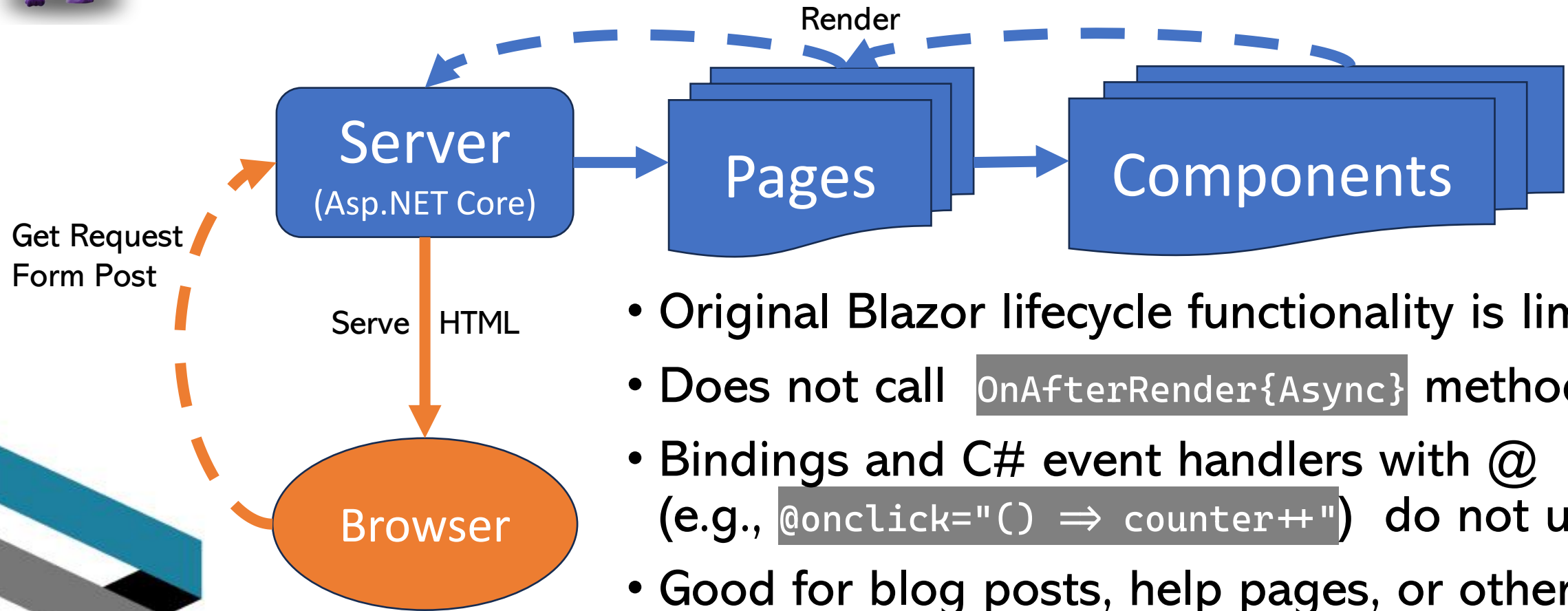
*technically a "Blazor Hosting Model", not a render mode*
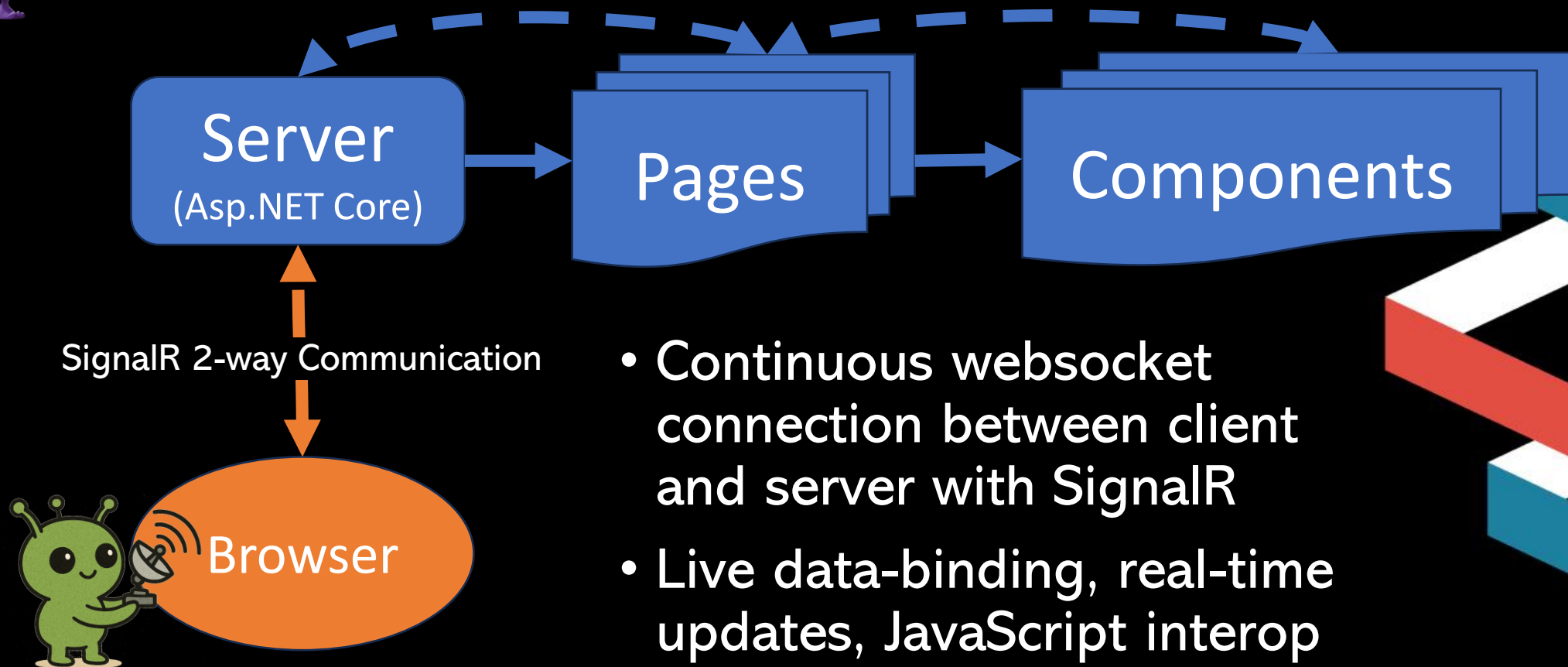
# Blazor Render Modes: Static Server

Render

Server
(Asp.NET Core)

Pages

Components

Get Request
Form Post

Serve | HTML

Browser

- "Traditional" server-side web, similar to ASP classic, WebForms, MVC, and Razor Pages
- Only static HTML, CSS, and JS files are sent to the client
- Supports a single-render and form post-backs
- No interactive updates via C# (can still use JS)

# Blazor Render Modes: Static Server (cont.)

Render

Server (Asp.NET Core) → Pages → Components

Get Request
Form Post

Serve HTML

Browser

- Original Blazor lifecycle functionality is limited
- Does not call `OnAfterRender{Async}` method
- Bindings and C# event handlers with @ (e.g., `@onclick="() ⇒ counter++"`) do not update
- Good for blog posts, help pages, or other read-only content and simple forms

# Blazor Render Modes: Interactive Server

**Server** (Asp.NET Core)

**Pages**

**Components**

SignalR 2-way Communication

**Browser**

- Continuous websocket connection between client and server with SignalR

- Live data-binding, real-time updates, JavaScript interop

- Direct access to server data store

- Fast on first load

# Blazor Render Modes: Interactive WebAssembly

Download .NET Runtime

**Server** (Asp.NET Core)

**Client** (WebAssembly)

**Pages**

**Components**

HttpClient Web API Calls
SignalR, gRPC

- Runs in the client browser

- Live data-binding, real-time updates, JavaScript interop

- HttpClient calls to communicate with server web API

- Single-threaded

- Larger download == slower first load

- Faster interactions after first load (no network latency on events)

- Closest in approach to most JS SPA frameworks

- Available in the hosted Blazor Web App and standalone WebAssembly projects
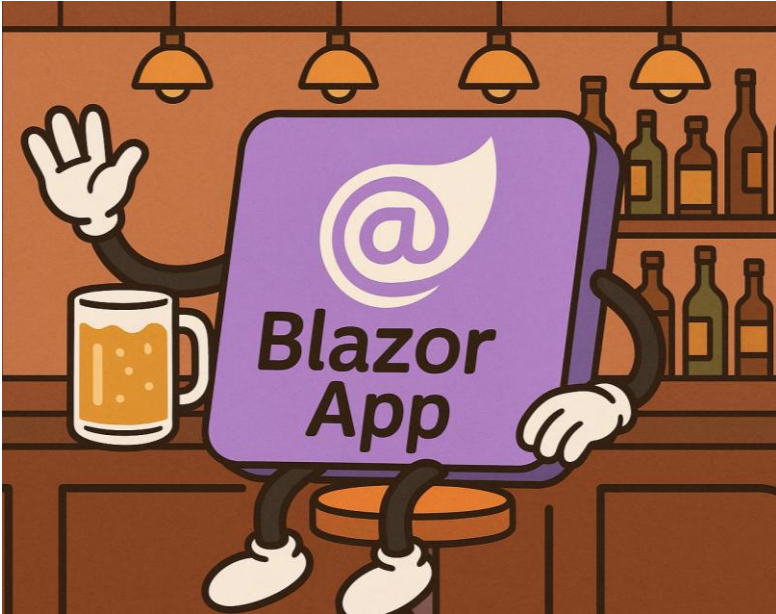
# Blazor Render Modes: Interactive Auto

- On first load, runs from server, creating SignalR connection

- In the background, downloads .NET runtime and client code

- On next load, switches to running from WebAssembly

- "Best of both worlds"
  - Fast start on first load (server)
  - More responsive and robust interactions (client)

- Requires flexible data handling/abstraction to handle both client and server modes
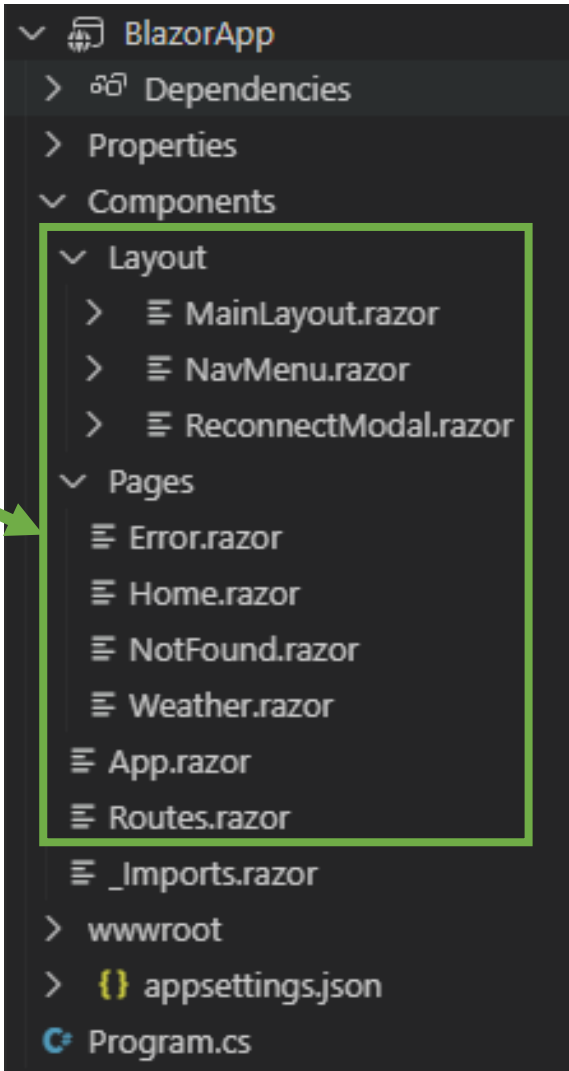
# *The Blazor Render Modes enter a bar…*



- *Static Server* orders a fancy drink, but once it arrives, they never touch it.

- *Interactive Server* walks in, sets their cell phone on the bar, leaves, and then begins ordering drinks over the phone.

- *Interactive WebAssembly* brings their laptop with them and boots it up before ordering.

- *Hybrid* always comes dressed up to look like a local, no matter where the bar is.
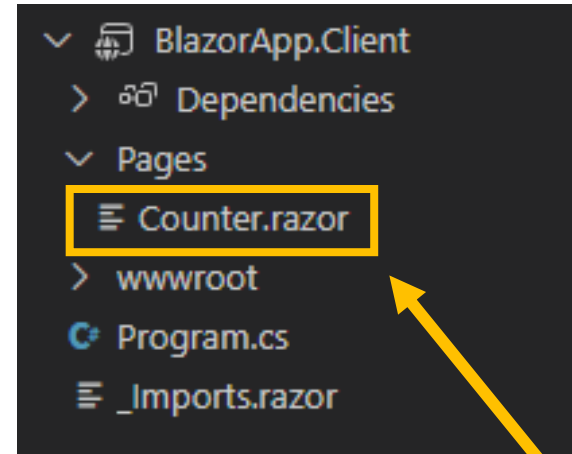
# Blazor Web App Solution Architecture

ASP.NET Core Server Project

*Project Ref.*

*Project Ref.*

WebAssembly Project

Razor Class Library

*Code can only run on the Server*

*Serves to the Browser as Static Files*

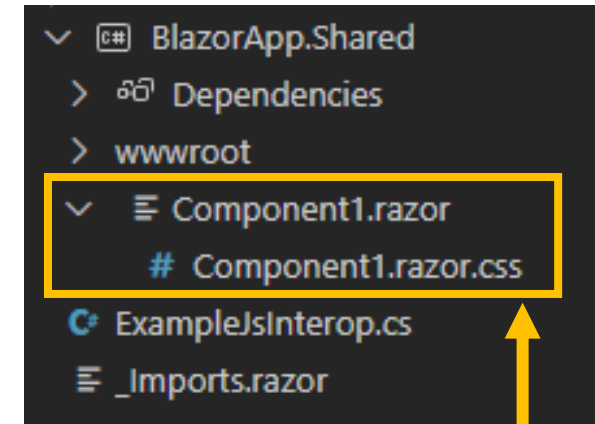*Code can run on Server or in Browser*

Interactive WebAssembly and Interactive Auto Components
must be accessible from the WebAssembly project

Static and Interactive Server Components can be placed in any
project that is accessible from the server project

# Defining the Render Mode

- ## At the top of the component

```
@page "/auto"
@rendermode InteractiveAuto


<PageTitle>Interactive Auto</PageTitle>
```

- ## When declaring a component

```
<SketchPad @rendermode="InteractiveServer" />
```
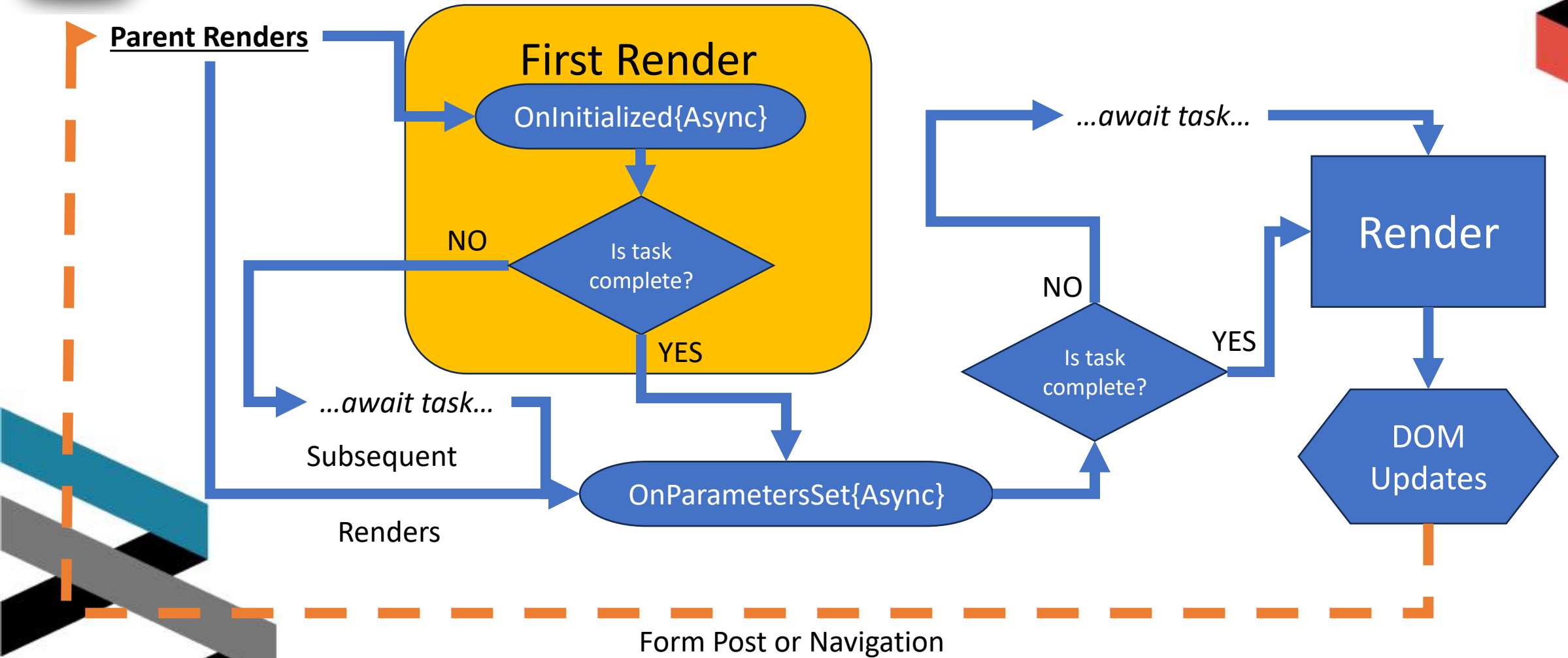
- ## Declare for the entire site

```
<Routes @rendermode="InteractiveServer" />
```

- *Nested components cannot change interactive modes or become non-interactive. The top level is always Static Server Mode, and each tree branch can only have one interactive mode.*
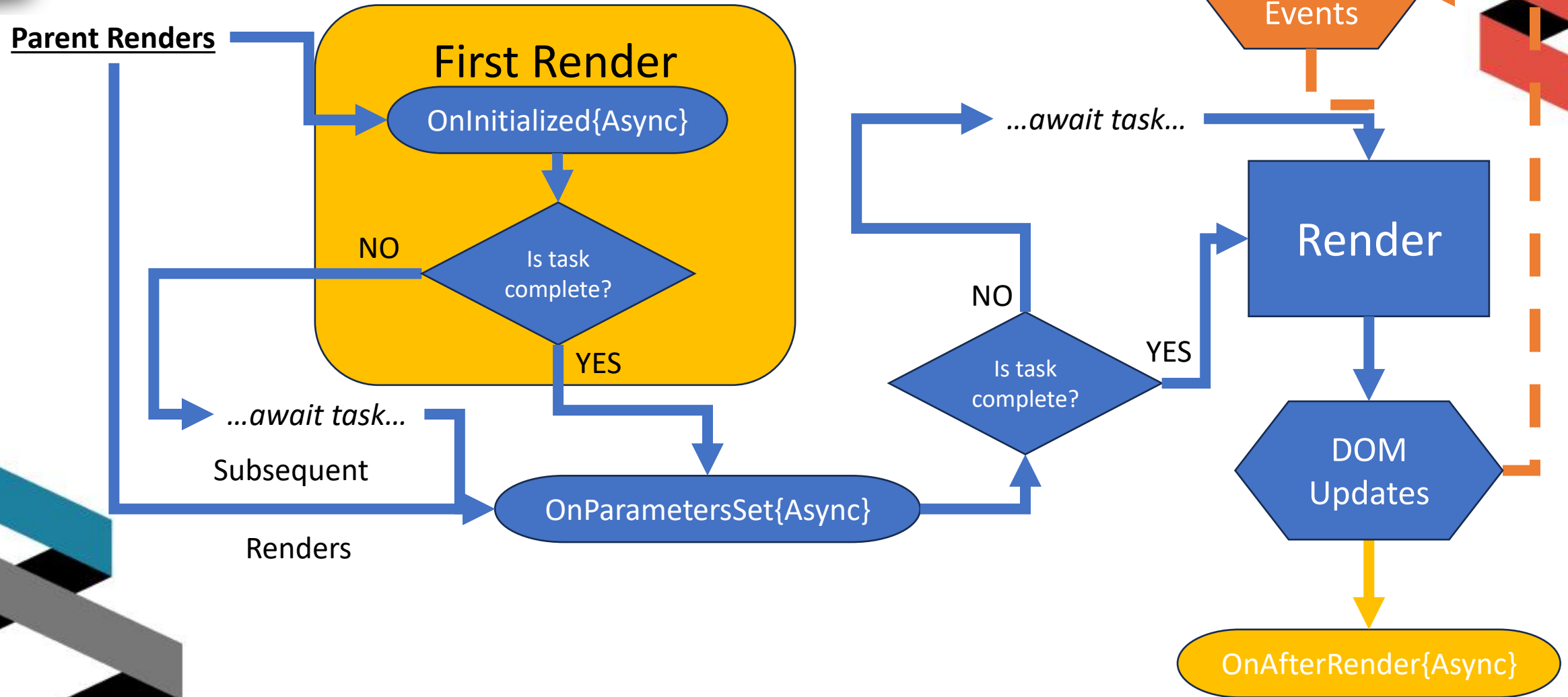
# Razor Component Lifecycle: Static Server Mode



**Parent Renders**

## First Render

OnInitialized{Async}

Is task complete?

NO

YES

...await task...

Subsequent

Renders

OnParametersSet{Async}

...await task...

Is task complete?

NO

YES

Render

DOM Updates

Form Post or Navigation

*State set in OnInitialized and OnParametersSet should be Idempotent*

# Razor Component Lifecycle: Interactive Modes

**Parent Renders**

**DOM Events**

## First Render

OnInitialized{Async}

Is task complete?

NO

YES

*...await task...*

Subsequent

OnParametersSet{Async}

Renders

Is task complete?

NO

YES

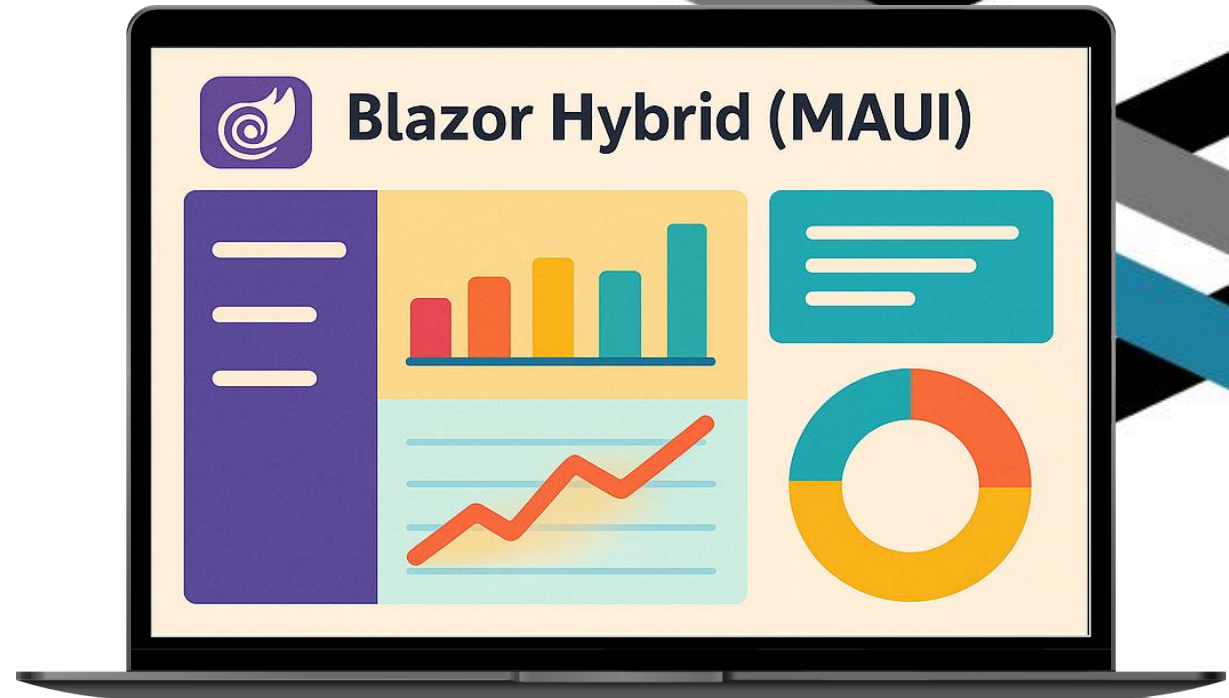*...await task...*

Render

DOM Updates

OnAfterRender{Async}

*Be careful of updating bound values in OnAfterRender, which could cause cycles*

# Blazor Hybrid

- Runs in a WebView in .NET MAUI (iOS, Android, Mac, Windows), WPF, or Windows Forms

- Native .NET multi-threaded code execution (not WebAssembly)

- Access to device APIs (GPS, Bluetooth, photos, etc.)

- Can reuse components or entire UI applications between web, desktop, and mobile

- Always interactive, fires `OnAfterRender`

- Does not require defining `@rendermode`

# Additional Rendering Patterns and Techniques

- Prerendering
  - Enabled by default for all interactive components
  - Improves first-load experience
  - Often the cause of unexpected duplicated logic from `OnInitialize` – avoid updating state in this method in a way that can't be repeated
  - Can define custom render mode to disable:
    `new InteractiveServerRenderMode(prerender: false)`
- Streaming rendering
  - Can use with prerendering or Static Server Mode
  - Improves the experience for components that load large data sets