Blazor Render Modes

Tim Purdum

Blazor Day

September 25, 2025

esri Partner Network Gold

Esri Partner Conference 2023 Award Winner GIS for Good

Federal Small Business Specialty

ArcGIS Hub Specialty

Nonprofit Specialty

State and Local Government Specialty

WOMEN OWNED

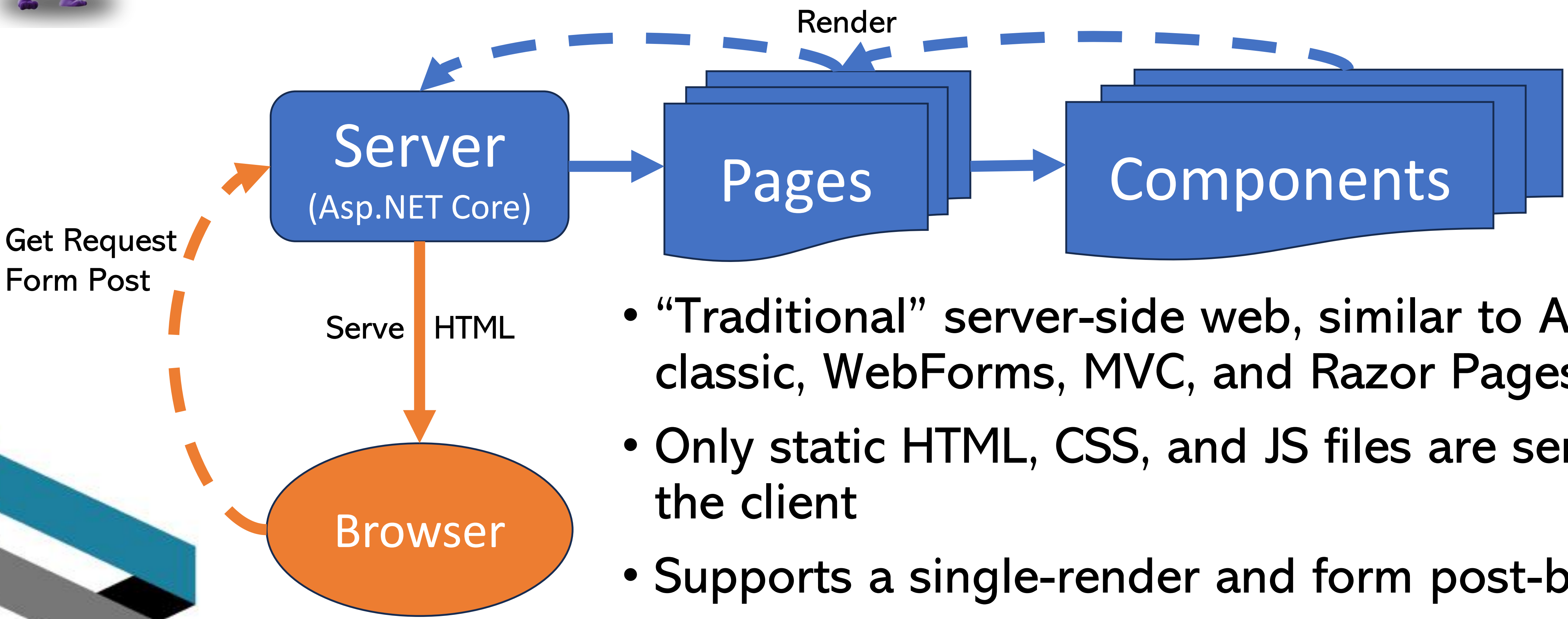
SBA EDWOSB CERTIFIED

Blazor Component Render Modes

- **Static Server Mode**
- **Interactive Server Mode**
- **Interactive WebAssembly Mode**
- **Interactive Auto Mode**
- **Blazor Hybrid ***

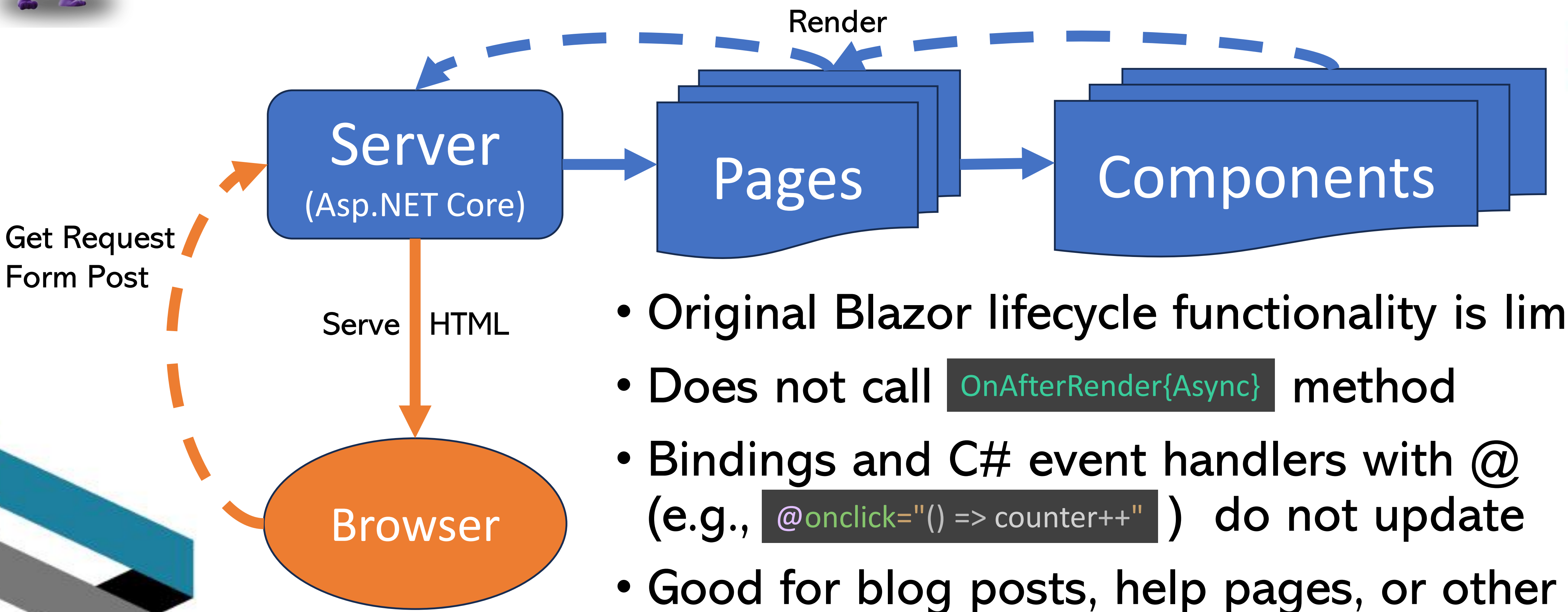
** technically a "Blazor Hosting Model", not a render mode*

Blazor Render Modes: Static Server



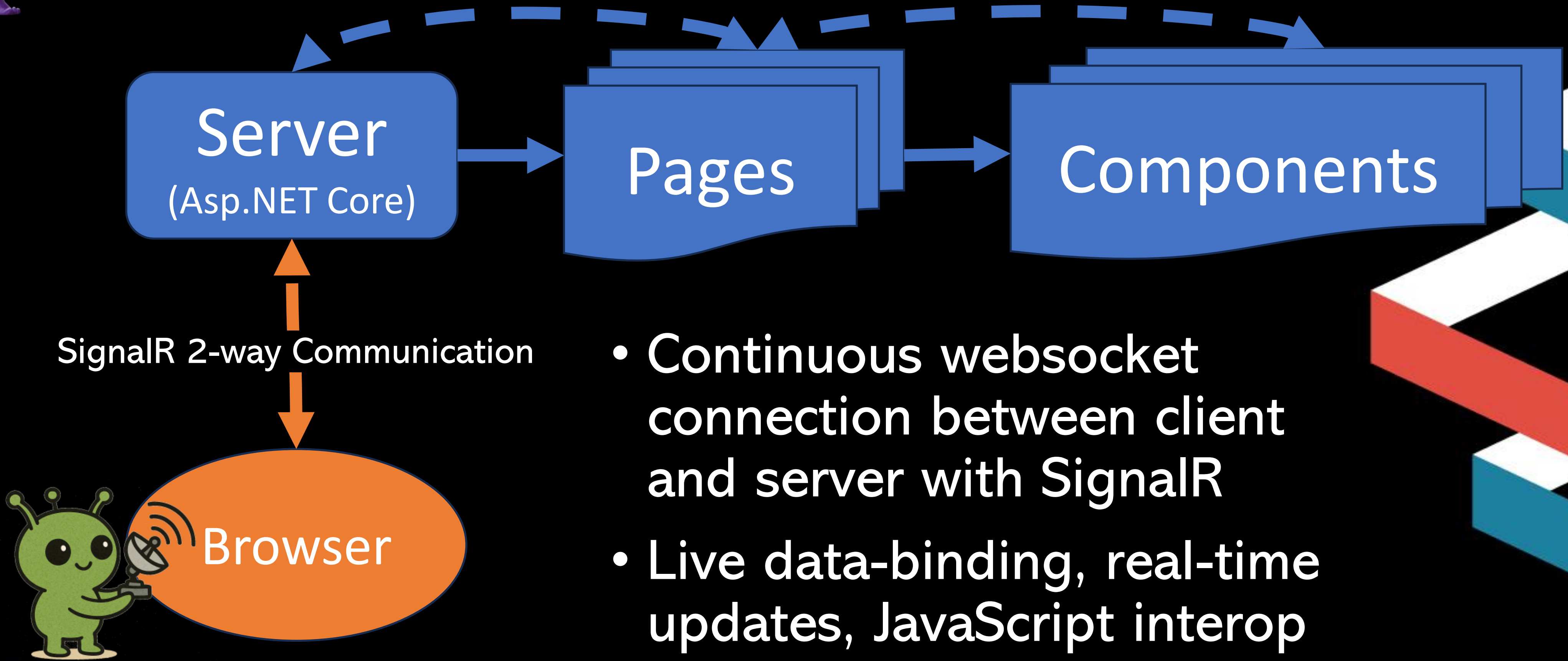
- "Traditional" server-side web, similar to ASP classic, WebForms, MVC, and Razor Pages
- Only static HTML, CSS, and JS files are sent to the client
- Supports a single-render and form post-backs
- No interactive updates via C# (can still use JS)

Blazor Render Modes: Static Server (cont.)



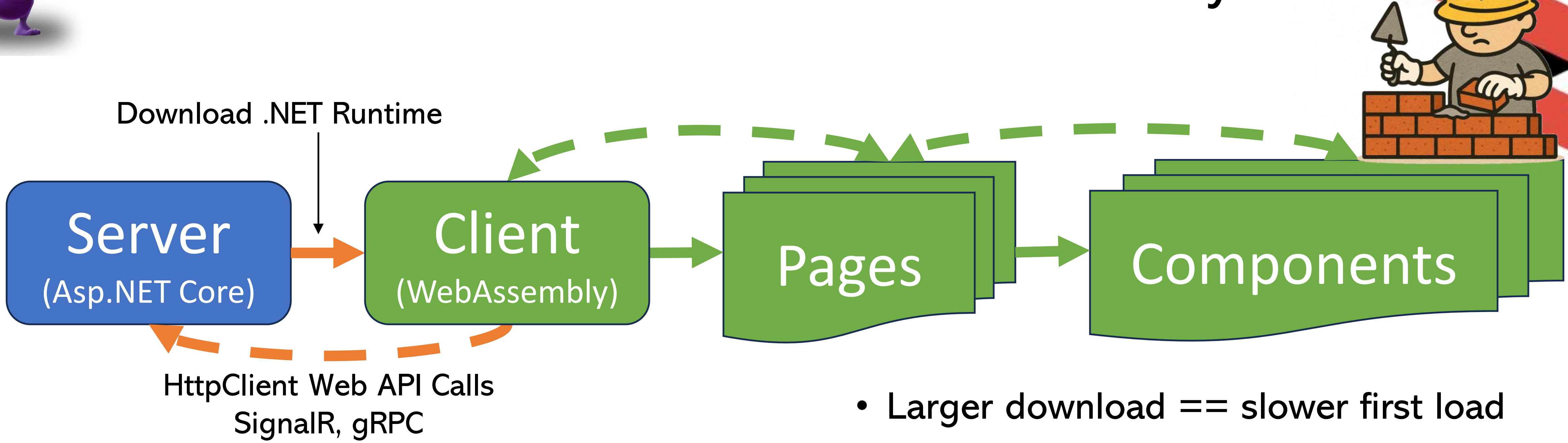
- Original Blazor lifecycle functionality is limited
- Does not call `OnAfterRender{Async}` method
- Bindings and C# event handlers with @ (e.g., `@onclick="() => counter++"`) do not update
- Good for blog posts, help pages, or other read-only content and simple forms

Blazor Render Modes: Interactive Server



- 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



- Runs in the client browser
- Live data-binding, real-time updates, JavaScript interop
- HttpClient calls to communicate with server web API
- Single-threaded

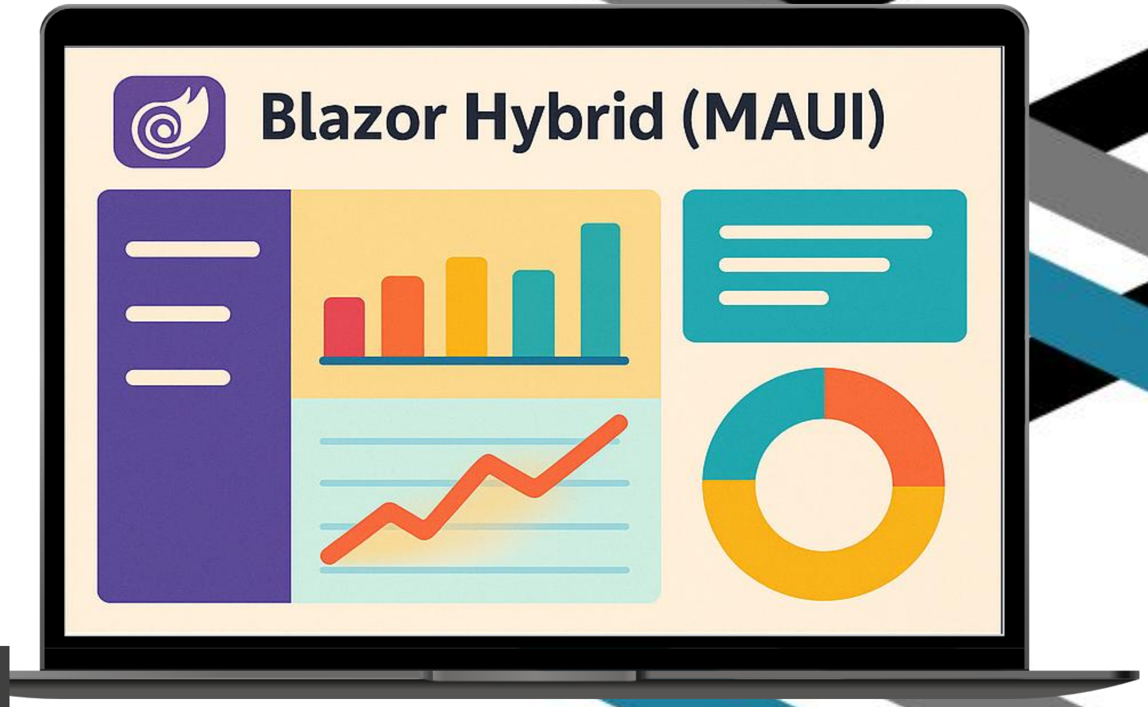
- Larger download == slower first load
- Faster interactions after first load (no network latency on events)
- Closest in approach to most JS SPA frameworks
- Available in the hosted Blazor Web App and standalone WebAssembly projects

Blazor Render Modes: Interactive Auto

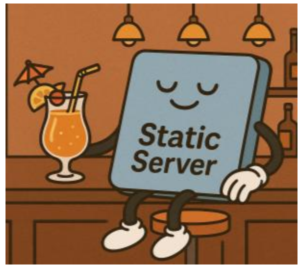
- On first load, runs from server, creating SignalR connection
- In the background, downloads .NET runtime and client code
- On next load, switches to running from WebAssembly
- “Best of both worlds”
  - Fast start on first load (server)
  - More responsive and robust interactions (client)
- Requires flexible data handling/abstraction to handle both client and server modes

Blazor Hybrid

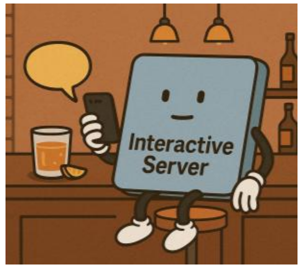
- Runs in a WebView in .NET MAUI (iOS, Android, Mac, Windows), WPF, or Windows Forms
- Native .NET multi-threaded code execution (not WebAssembly)
- Access to device APIs (GPS, Bluetooth, photos, etc.)
- Can reuse components or entire UI applications between web, desktop, and mobile
- Always interactive, fires `OnAfterRender{Async}`
- Does not require defining `@rendermode`



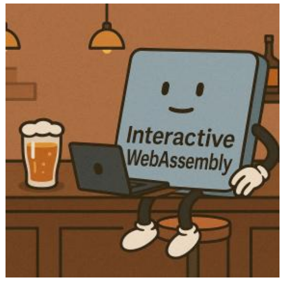
The Blazor Render Modes enter a bar…



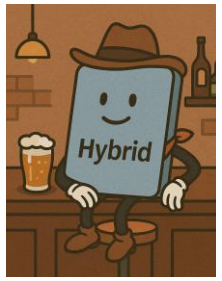
- ***Static Server*** orders a fancy drink, but once it arrives, they never touch it.



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

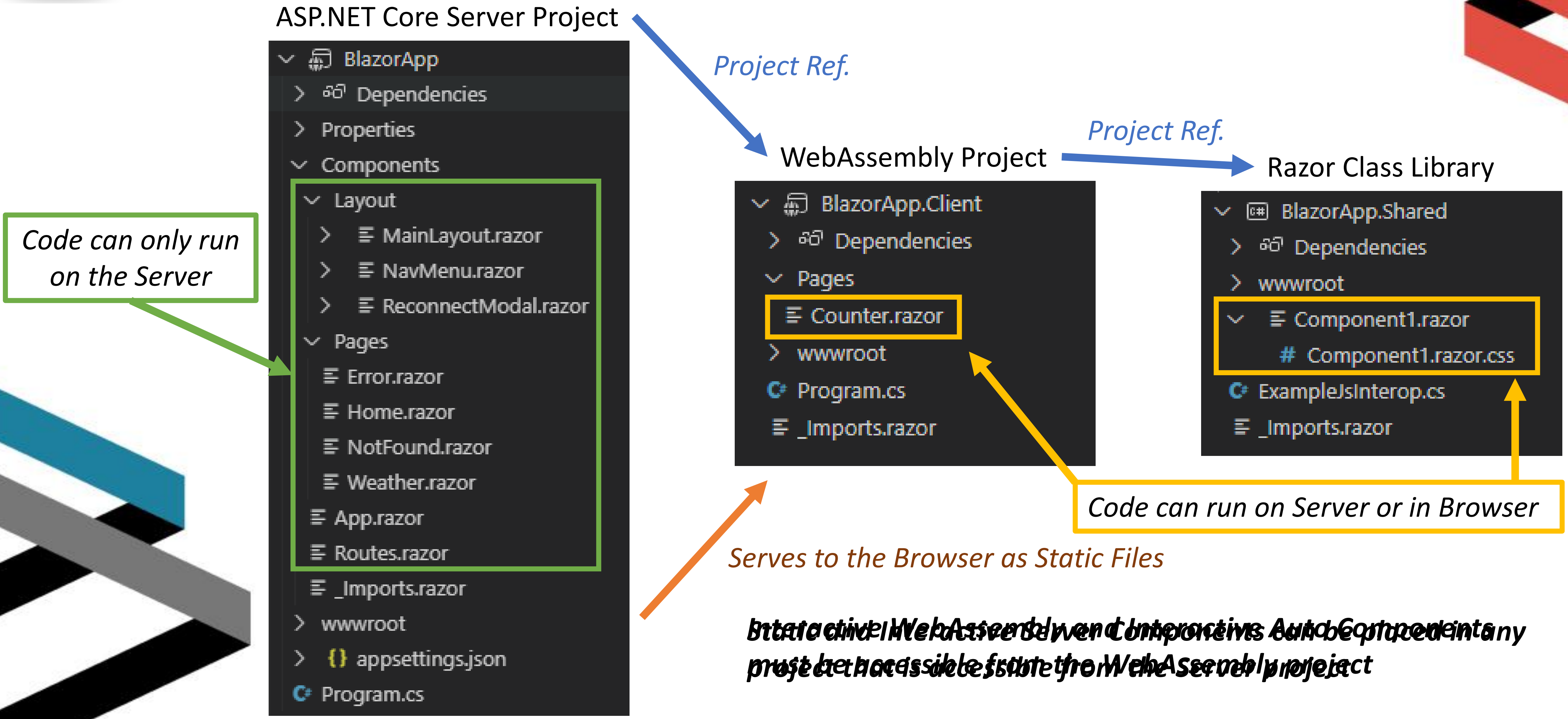


- ***Interactive WebAssembly*** brings their laptop with them and boots it up before ordering.



- ***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

```
BlazorApp
  Dependencies
  Properties
  Components
    Layout
      MainLayout.razor
      NavMenu.razor
      ReconnectModal.razor
    Pages
      Error.razor
      Home.razor
      NotFound.razor
      Weather.razor
    App.razor
    Routes.razor
    _Imports.razor
  wwwroot
  appsettings.json
  Program.cs
```

Code can only run on the Server

Project Ref.

WebAssembly Project

```
BlazorApp.Client
  Dependencies
  Pages
    Counter.razor
  wwwroot
  Program.cs
  _Imports.razor
```

Project Ref.

Razor Class Library

```
BlazorApp.Shared
  Dependencies
  wwwroot
  Component1.razor
    Component1.razor.css
  ExampleJsInterop.cs
  _Imports.razor
```

Code can run on Server or in Browser

Serves to the Browser as Static Files

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" />
```

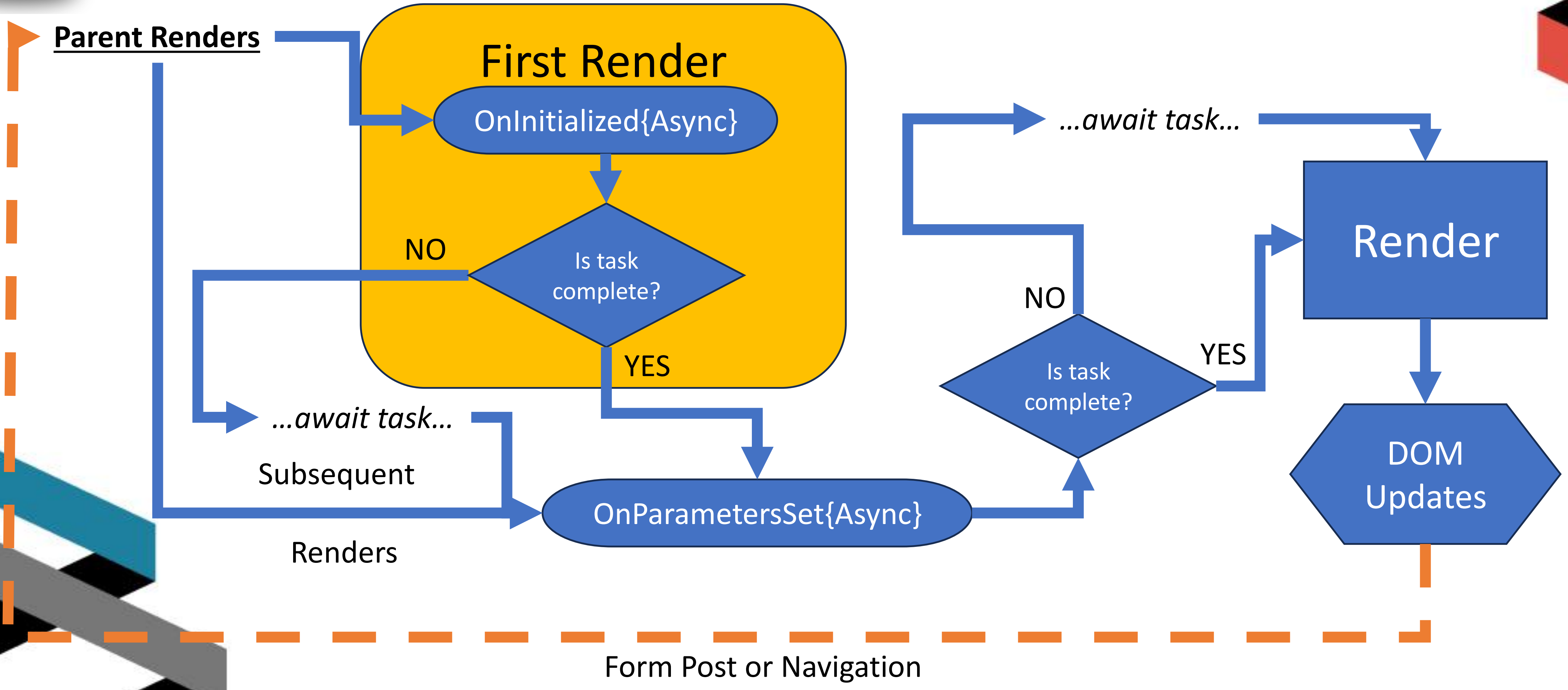
- Declare for the entire site

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

Defining the Render Mode

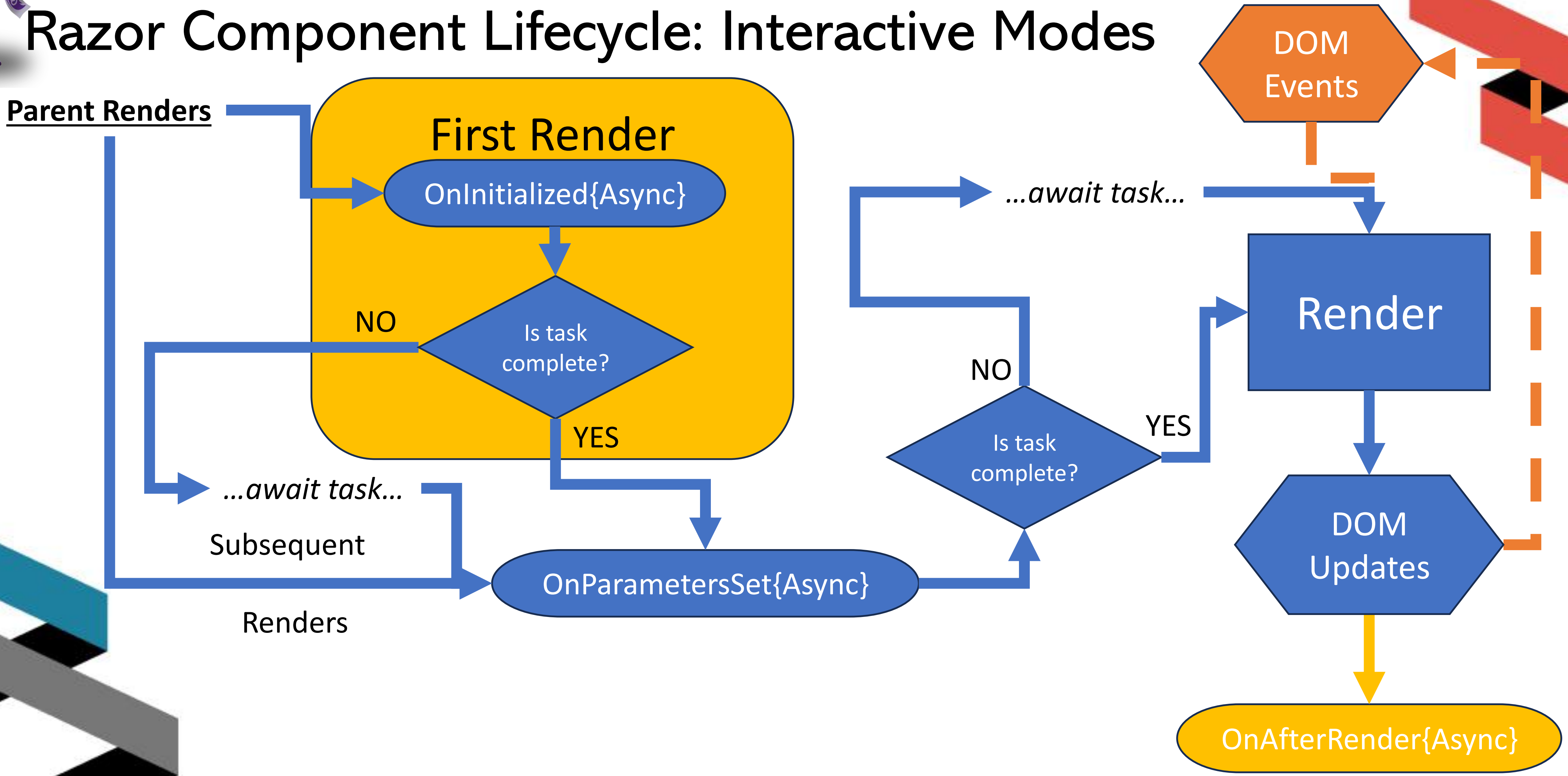
- The top level in a Blazor Web App is always Static Server Mode
- Once you define an Interactive Mode, all child components will inherit that mode
- i.e., you cannot place a WebAssembly component inside an Interactive Server component or vice versa
- You can *read* the current render mode with `@RendererInfo.Name` in any component

Razor Component Lifecycle: Static Server Mode



State set in OnInitialized and OnParametersSet should be Idempotent

Razor Component Lifecycle: Interactive Modes



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

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 `OnInitialized` - 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

Resources

- ASP.NET Core Blazor render modes | Microsoft Learn
  - Official Documentation
- Blazor Basics: Blazor Render Modes in .NET 8 | Telerik Blog
  - Good Overview
- AlexNek/BlazorNet8PlusExamples | GitHub
  - Cool interactive sample
- dymaptic/GeoBlazor.RenderModes | GitHub
  - The GeoBlazor render modes sample I shared
- BlazorDay 2025 | TimPurdum.Dev
  - Full list of these links and demo materials

Thank You!

dymaptic

Notes & Links @
https://timpurdum.dev

GeoBlazor