

1. What Is a Blazor Web App?

A **Blazor Web App** is a **single project** that can run:

- As **Blazor Server**
- As **Blazor WebAssembly**
- Or **Auto mode** (initial Server render, then WebAssembly)

This replaces the older *separate* Server and WASM templates.

2. Render Modes in Blazor Web App

Additional information

Blazor Web App C# Linux macOS Windows Blazor Cloud Web

Framework ⓘ
.NET 8.0 (Long Term Support) ▼

Authentication type ⓘ
None ▼

☒ Configure for HTTPS ⓘ

Interactive render mode ⓘ
Auto (Server and WebAssembly) ▼

Interactivity location ⓘ
Per page/component ▼

☒ Include sample pages ⓘ

☐ Do not use top-level statements ⓘ

☐ Enlist in Aspire orchestration ⓘ

1. What Are Render Modes in Blazor Web App?

A render mode defines where a Blazor component is rendered and where its logic executes.

In a Blazor Web App, each component can independently choose:

- Where it runs
- How it becomes interactive
- When WebAssembly is downloaded

This is the major architectural change introduced in .NET 8.

2. Why Render Modes Were Introduced

Before .NET 8:

- You had to choose Blazor Server *or* Blazor WebAssembly at project creation time.

With .NET 8+:

- A single project can mix Server + WebAssembly
 - Interactivity can be progressive
 - Performance, scalability, and UX can be optimized per component
-

3. The Four Render Modes

1. Static (Default)

- Server-side rendering only
- No interactivity
- No SignalR
- No WebAssembly

`<h3>Hello World</h3>`

Use cases:

- Landing pages
- SEO-focused content

- Read-only pages
-

2. InteractiveServer

- Component executes on the server
- UI updates via SignalR
- Thin client

@rendermode InteractiveServer

Characteristics:

- Real-time UI updates
- Low client CPU usage
- Requires persistent connection

Best for:

- Dashboards
 - Admin portals
 - Intranet applications
-

3. InteractiveWebAssembly

- Component executes in the browser
- Downloads .NET runtime + app DLLs
- No server connection after load

@rendermode InteractiveWebAssembly

Characteristics:

- Client-side execution
- Offline capability
- Higher initial load time

Best for:

- Public-facing apps
- Offline-first applications
- High-scale systems

4. InteractiveAuto

- Starts as Server
- Seamlessly switches to WebAssembly

@rendermode InteractiveAuto

Characteristics:

- Fast initial render
- Becomes client-side after load

This is the recommended default for most apps.

4. Render Mode Comparison Table

Feature	Static	Server	WebAssembly	Auto
Interactive	✗	✓	✓	✓
SignalR	✗	✓	✗	Initial only
Offline	✗	✗	✓	✓ (after WASM)
SEO	✓	✓	⚠	✓
Initial Load	Fast	Fast	Slow	Fast
Scalability	High	Limited	Very High	High

5. Where Render Modes Are Applied

Component-Level (Recommended)

@rendermode InteractiveAuto

Page-Level

@page "/products"

@rendermode InteractiveWebAssembly

App-Level (Not Recommended)

Applied globally — reduces flexibility.

6. Program.cs Configuration (Mandatory)

To enable render modes:

builder.Services.AddRazorComponents()

.AddInteractiveServerComponents()

.AddInteractiveWebAssemblyComponents();

Without this, interactive modes will not work.
