Interactive WebAssembly and **WASM Standalone** in Blazor

---

1. Conceptual Overview

Interactive WebAssembly (Blazor Web App model)

- Introduced with **.NET 8+ Blazor Web Apps**
- WebAssembly runs **in the browser**
- The app is **hosted by ASP.NET Core**
- Uses **render modes** (InteractiveWebAssembly)
- Supports **hybrid rendering** (SSR → WebAssembly)

WASM Standalone (Classic Blazor WASM)

- Introduced in **Blazor WebAssembly (pre-.NET 8)**
- Runs **entirely in the browser**
- No server-side UI hosting
- Pure **Single Page Application (SPA)** model

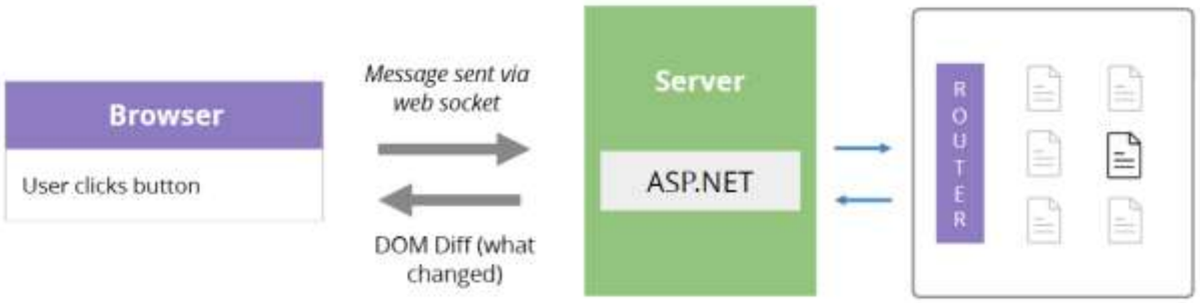
---

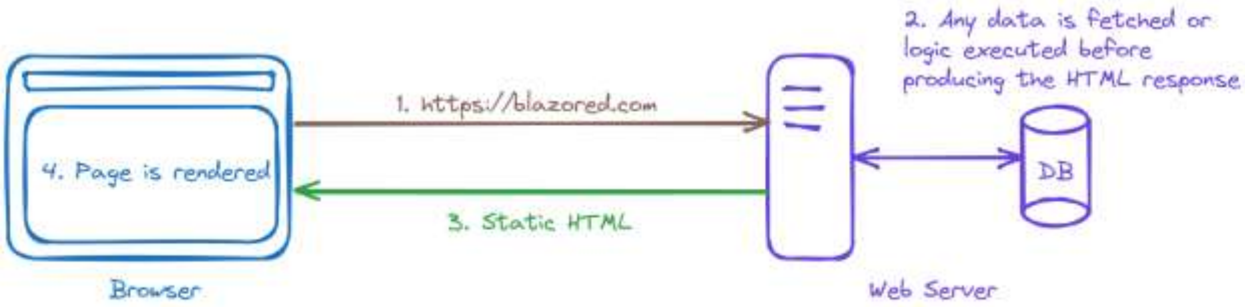
2. Hosting Model

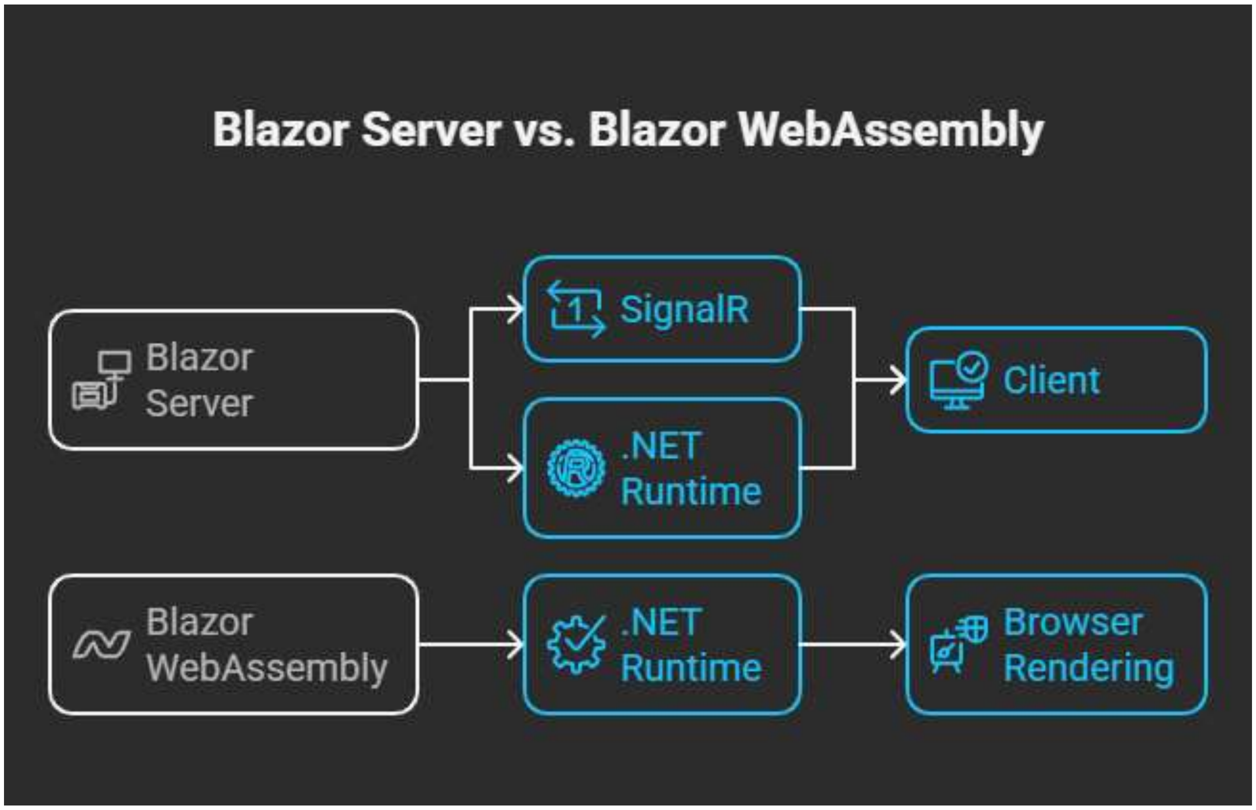
Aspect	**Interactive WebAssembly**	**WASM Standalone**
Server Required	Yes (ASP.NET Core)	No
Hosting Location	Server + Browser	Browser only
Deployment	ASP.NET Core app	Static files (CDN/IIS/S3)

---

3. Rendering Pipeline




1. https://blazored.com
2. Any data is fetched or logic executed before producing the HTML response
DB
3. Static HTML
4. Page is rendered
Browser
Web Server



Interactive WebAssembly

1. Page is **server-side rendered (SSR)**

2. HTML is sent immediately to browser
3. WebAssembly loads in background
4. App becomes **fully interactive**

WASM Standalone

1. Browser downloads:
    - .NET runtime
    - DLLs
    - App assemblies
2. App renders **only after download completes**
3. No SSR

---

4. Performance & User Experience

Feature	**Interactive WebAssembly**	**WASM Standalone**
First Load	Faster (SSR)	Slower
Perceived Speed	High	Medium
SEO	Excellent	Poor
Offline Support	Limited	Excellent
Cold Start	Minimal	High

---

5. Code Structure Differences

Interactive WebAssembly

@rendermode InteractiveWebAssembly

- Components can:
    - Start as SSR
    - Become interactive later

- Can mix:
    - Static pages
    - Interactive WASM
    - Interactive Server

WASM Standalone

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

- Entire app is client-only
- No render modes
- No SSR

---

6. API & Security Model

Aspect	**Interactive WebAssembly**	**WASM Standalone**
API Calls	Can be server-local	Must be HTTP
Authentication	Server-based auth possible	Token/JWT only
Secrets	Can stay on server	Must be public-safe

---

7. Progressive Web App (PWA)

Feature	**Interactive WASM**	**WASM Standalone**
PWA Support	Partial	Full
Offline Mode	Weak	Strong
Background Sync	Limited	Supported
Push Notifications	Supported	Supported

Note: For **offline-first apps**, WASM Standalone is superior.

8. When to Use Which?

Choose Interactive WebAssembly when:

- You want **fast initial load**
- SEO matters
- You want **server + client hybrid**
- You are building enterprise apps
- You want shared auth, logging, DI

Choose WASM Standalone when:

- You need **offline-first**
- You want **static hosting**
- You are building a **PWA**
- You want zero server dependency
- You want CDN-scale deployment

9. Summary Comparison Table

Dimension	**Interactive WebAssembly**	**WASM Standalone**
Rendering	SSR → WASM	Client-only
Server Dependency	Required	Not required
SEO	Excellent	Poor
Offline Support	Limited	Strong
Hosting	ASP.NET Core	Static files
Complexity	Higher	Lower
Best For	Enterprise, SEO apps	PWAs, offline apps

---

10. One-Line Definition

- **Interactive WebAssembly**:
  A server-hosted Blazor app that becomes interactive using WebAssembly.
- **WASM Standalone**:
  A pure client-side Blazor SPA running entirely in the browser.
