Blazor Hosting Models

Blazor Hosting

There are two hosting types for a Blazor App:

1. Blazor WebAssembly (Is in preview for ASP.NET Core 3.1)
2. Blazor Server (Supported in ASP.NET Core 3.0)

Blazor WebAssembly :

- The principle hosting model for Blazor is running client-side in the browser using WebAssembly.
- The app, its dependencies and the .NET runtime are downloaded to the browser
- The app is executed directly on the browser UI thread (Fig1)

Fig 1

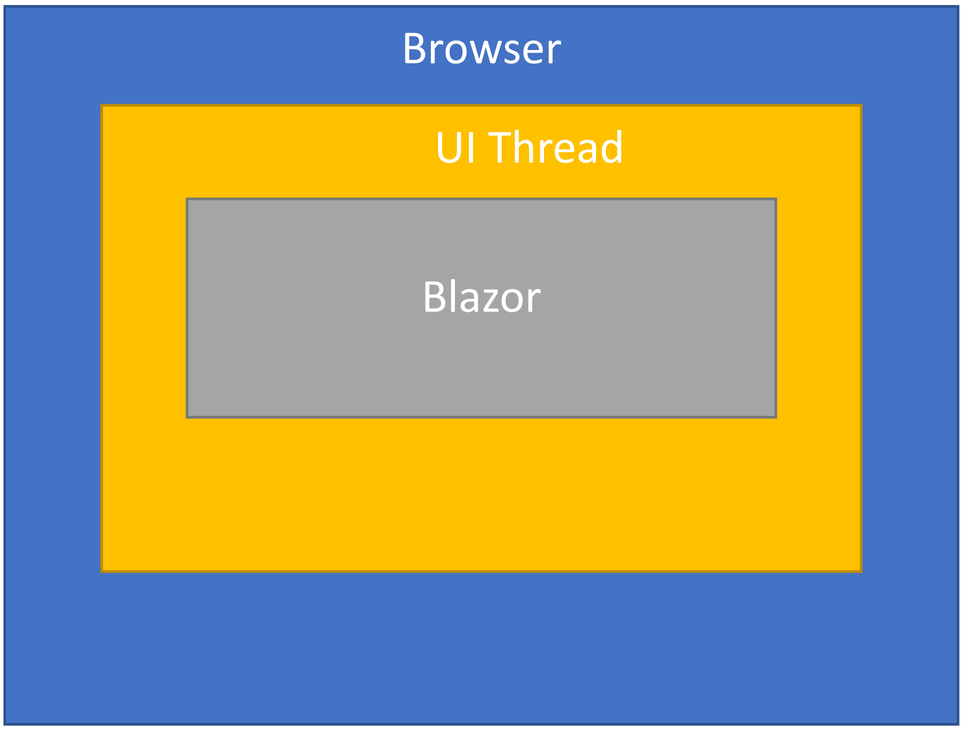
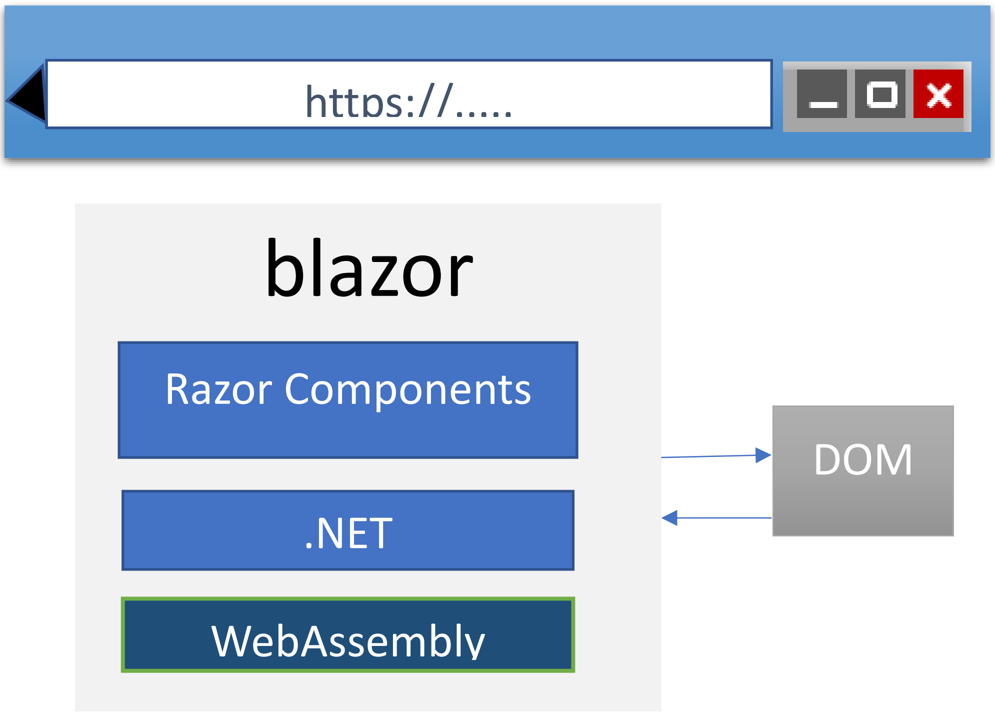


Fig 2



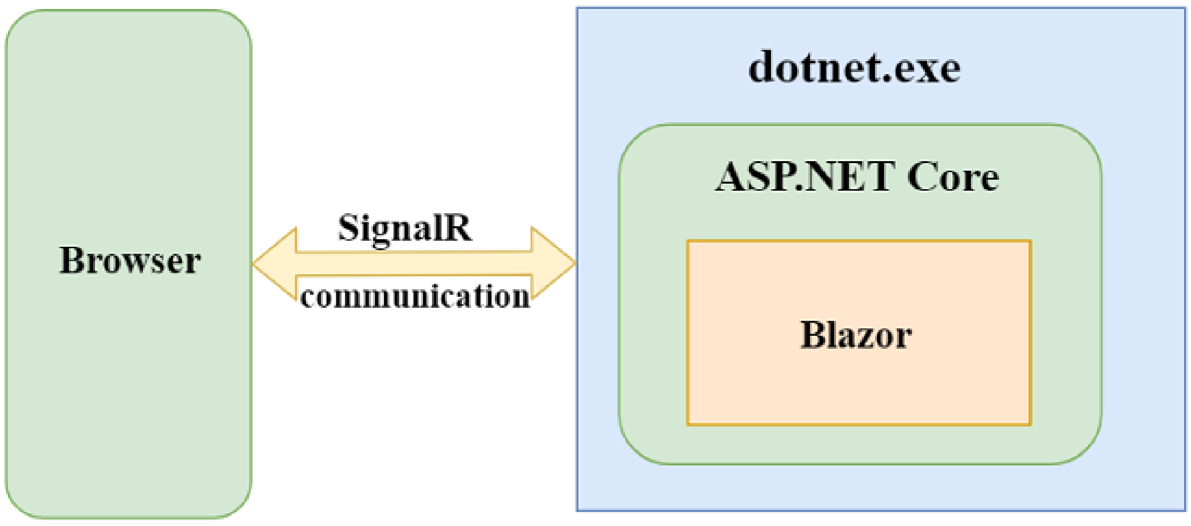
Blazor WebAssembly (Pros) :

- Apps run purely client-side
- Such apps can be deployed to static site hosting solutions like GitHub Pages
- .NET isn’t required on the server at all.
- To get all the benefits of Blazor and full-stack .NET web development, host your Blazor WebAssembly app with ASP.NET Core
- By using .NET on both client and server, we can easily share code

Blazor WebAssembly (Cons) :

- Big download size
- Requires support for WebAssembly
- Less mature runtime
- Debugging functionalities not so great
- Still in preview
- DLLs are also downloaded

Fig 3



Blazor Server (Difference from Client Side) :

- Instead of downloading everything on the client and running on Web Assembly it uses

1. An ASP.NET Core application running on the server
2. Opening a real-time Signal R connection from the browser to the server

Blazor Server (Pros) :

1. No need for Javascript
2. Small size, fast loading
3. Use of .NET Core
4. Debugging works great as expected
5. Runs on any browser as no web assembly needed
6. Secure code as it never leaves the server

Blazor Server (Cons) :

1. Need for a server-side
2. No offline support(without server availability)
3. Higher latency and worse user experience
4. Hard to maintain and scale (as there are millions of Signal R connections to be managed if there are number of browser tabs opened by hundreds of thousands of users)
5. One of the solutions to the maintainability and scaling issues is to use Microsoft’s Azure SignalR service