|  |  |
| --- | --- |
|  |  |
| Blazor Server | Blazor Webassembly |
| Blazor is executed on the server from within an ASP.NET Core app. UI updates, event handling |  |
| calls are handled over a SignalR connection |  |
| In this configuration there is no need for WebAssembly and Blazor is executed on the ASP.NET Core runtime at the server. All UI updates are sent as diffs, bidirectionally as binary packets over web sockets |  |
| Blazor Server is supported in ASP.NET Core 3.0. | Blazor webassembly is supported in ASP.NET Core 3.1. |
| To the user, the application is indistinguishable from any other web application. | To the user, the application is indistinguishable from any other web application. |
| the app is executed on the server from within an ASP.NET Core app. | The Blazor app, its dependencies, and the .NET runtime are downloaded to the browser. |
|  |  |
| **ASP.NET Core Hosted** | **Client-Side** |
| the application is executed on the server from within an ASP.NET Core app. | The app is executed directly on the browser UI thread. |
| Do not work in offline mode since the signal connection is required for various actions(Event dispatching, DOM updates etc). | Works in offline mode. |
| DOM updates will be handled in the server part which is hosted in ASP.NET Core server. | Since the whole application resides in the browser, DOM updates will be handled in the browser itself. |
| Suitable for database connectivity and operation. You can use Web APIs to consume and do database operations. | Not suitable for database connectivity as it required to send the connection string to the client side which might be a potential security risk. |
| Debugging is not possible for client part as browser DevTools are at the initial stage of using web assemblies. | Debugging is hard as browser DevTools are at the initial stage of using web assemblies. |
| Initial page load might be high based on the application side as all required DLL/Assemblies will be downloaded at the browser. | The initial page load might be high as all required DLL/Assemblies will be downloaded at the browser. |
|  |  |
| **Server-side** |  |
| application will reside in the ASP.NET Core server and requests will be made from browser to server for various actions |  |
| It does not send any form of DLL to the browser. |  |
| DOM updates will be handled in the server part which is hosted in ASP.NET Core server. |  |
| Do not work in offline mode since the signal connection is required for various actions(Event dispatching, DOM updates etc). |  |
| Suitable for database connectivity and operation. You can use Web APIs to consume and do database operations. |  |
| Debugging is good. You can use default visual studio debugging procedure to debug your application. |  |
| The initial page load will be good. |  |
| it does not send WebAssembly to the browsers. |  |
| the app and component models are the same. | the app and component models are the same. |
|  |  |

Webassambly:

Its just a way to run high level programming languages on the browser.

It download all .dlls at the first load of page.

**Blazor WebAssembly**

|  |  |
| --- | --- |
| Voordelen | Nadelen |
| There's no .NET server-side dependency. The app is fully functioning after it's downloaded to the client. | The app is restricted to the capabilities of the browser. |
| Client resources and capabilities are fully leveraged. | Capable client hardware and software (for example, WebAssembly support) is required. |
| Work is offloaded from the server to the client. | Download size is larger, and apps take longer to load. |
| An ASP.NET Core web server isn't required to host the app. Serverless deployment scenarios are possible (for example, serving the app from a CDN). | .NET runtime and tooling support is less mature. For example, limitations exist in [.NET Standard](https://docs.microsoft.com/en-us/dotnet/standard/net-standard) support and debugging. |
| Not need for Javascript |  |

## Blazor Server

|  |  |
| --- | --- |
| Voordelen | Nadelen |
| Download size is significantly smaller than a Blazor WebAssembly app, and the app loads much faster. | Higher latency usually exists. Every user interaction involves a network hop. |
| The app takes full advantage of server capabilities, including use of any .NET Core compatible APIs. | There's no offline support. If the client connection fails, the app stops working. |
| .NET Core on the server is used to run the app, so existing .NET tooling, such as debugging, works as expected. | Scalability is challenging for apps with many users. The server must manage multiple client connections and handle client state. |
| Thin clients are supported. For example, Blazor Server apps work with browsers that don't support WebAssembly and on resource-constrained devices. | An ASP.NET Core server is required to serve the app. Serverless deployment scenarios aren't possible (for example, serving the app from a CDN). |
|  |  |