Microsoft

BLAZOR

Cheat Sheet

ezzylearning.net

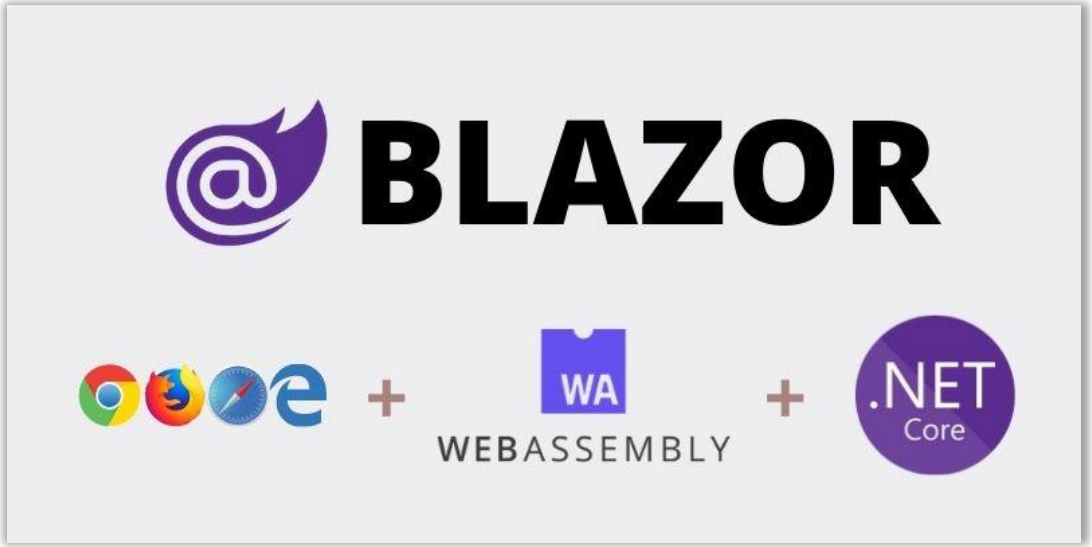
Blazor Cheat Sheet

Why **Blazor**?

1. We can create rich interactive web apps using C# instead of JavaScript.
2. We can share server-side and client-side app logic written in .NET.
3. We can render the UI from server as HTML and CSS for wide browser support, including mobile browsers.
4. We can run web apps on the clients in offline mode using WebAssembly.
5. We can integrate with modern hosting platforms, such as Docker.
6. We can leverage the existing .NET ecosystem of .NET libraries.
7. We can enjoy the benefit of .NET's performance, reliability, and security.
8. We can stay productive with Visual Studio on Windows, Linux, and macOS

What is **Blazor**?

Blazor is a free, open-source, single-page apps (SPA) development framework that enables developers to build interactive web apps using C# on both servers as well as client-side. Blazor does not require any plugin to be installed on the client to execute the C#/.NET code inside a browser. It executes the .NET code using WebAssembly which is a web standard supported by all major browsers. Blazor can also run .NET code and build UI on the server and transfer only the updated DOM to clients over SignalR connections.

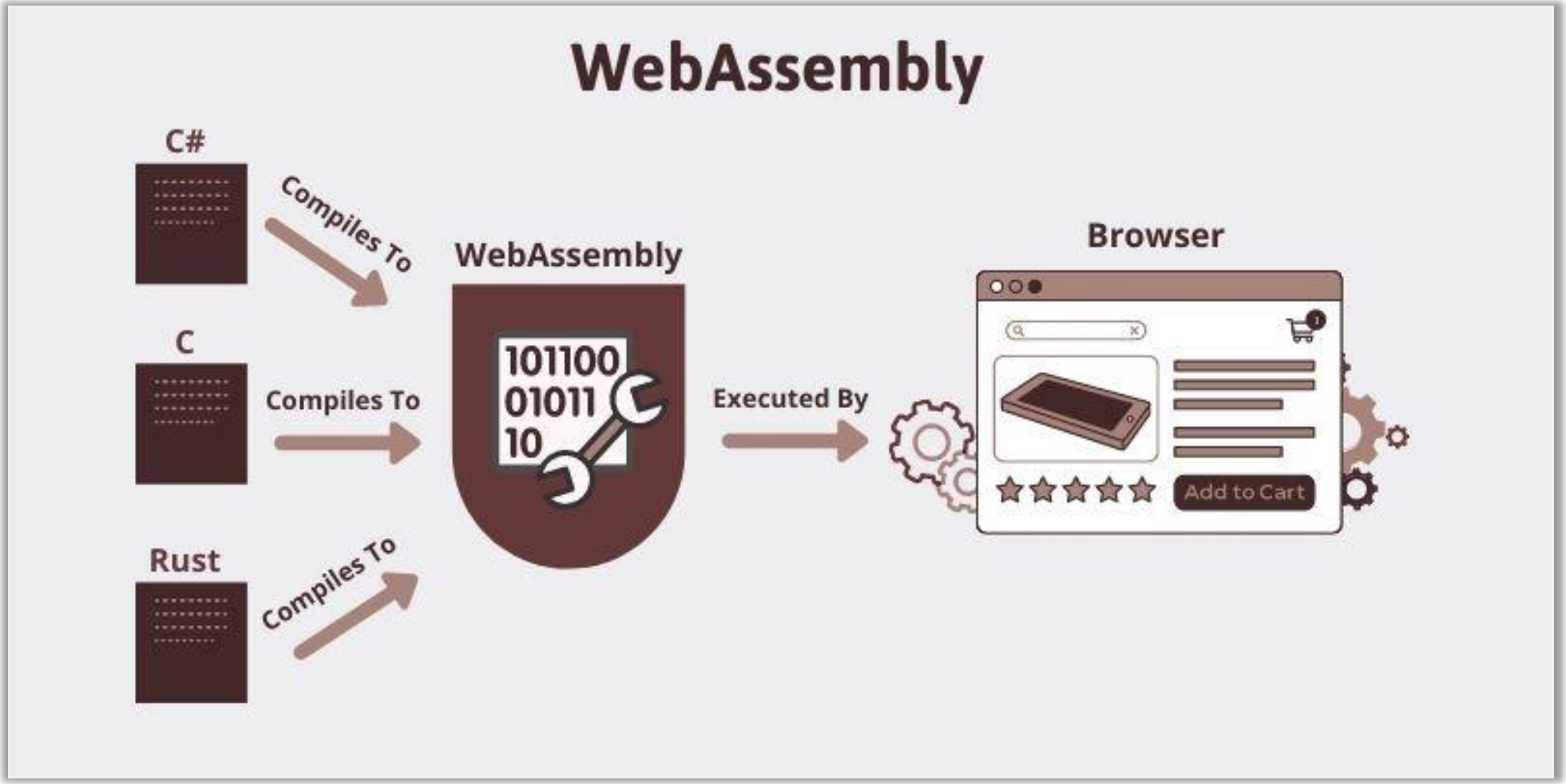


Blazor Supported Platforms

Browser	Version	Browser	Version
Apple Safari, including iOS	Latest Version	Microsoft Edge	Latest Version
Google Chrome, including Android	Latest Version	Mozilla Firefox	Latest Version

What is **WebAssembly**?

WebAssembly (sometimes abbreviated **Wasm**) is a portable binary format (low-level instructions set) designed to run on any host capable of interpreting those instructions. The main goal of WebAssembly is to allow developers to build high-performance web apps but the format is designed to be executed and integrated into other environments as well. WebAssembly is currently supported by all major browsers such as Chrome, Chrome for Android, Edge, Firefox, Safari, Opera, and many more.

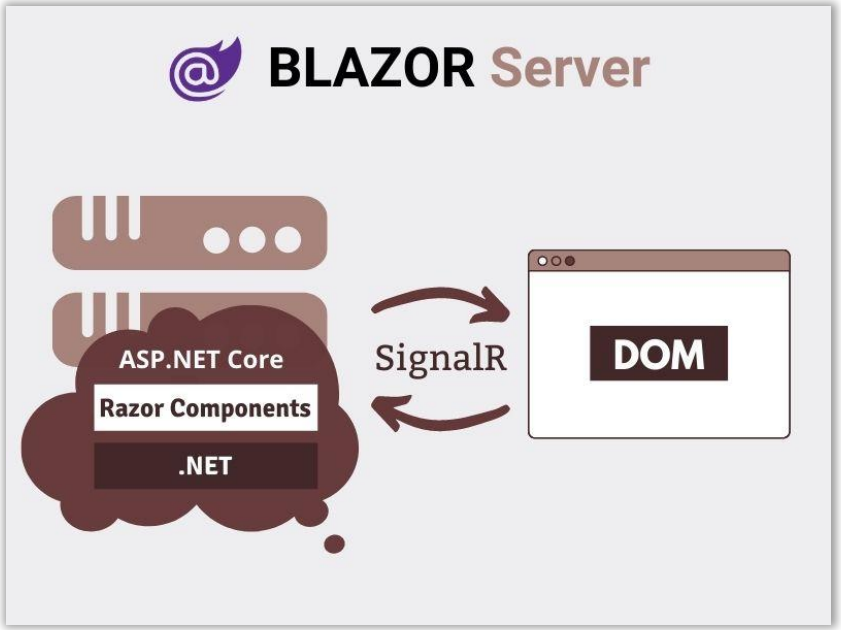


Blazor Hosting Models

The Blazor component model is the core of Blazor and it is designed in such a way that it keeps both calculating the UI changes and rendering the UI separate from each other. This is why the basic component model doesn't change no matter what method you are using to render your apps. There are two main hosting models available at the moment.

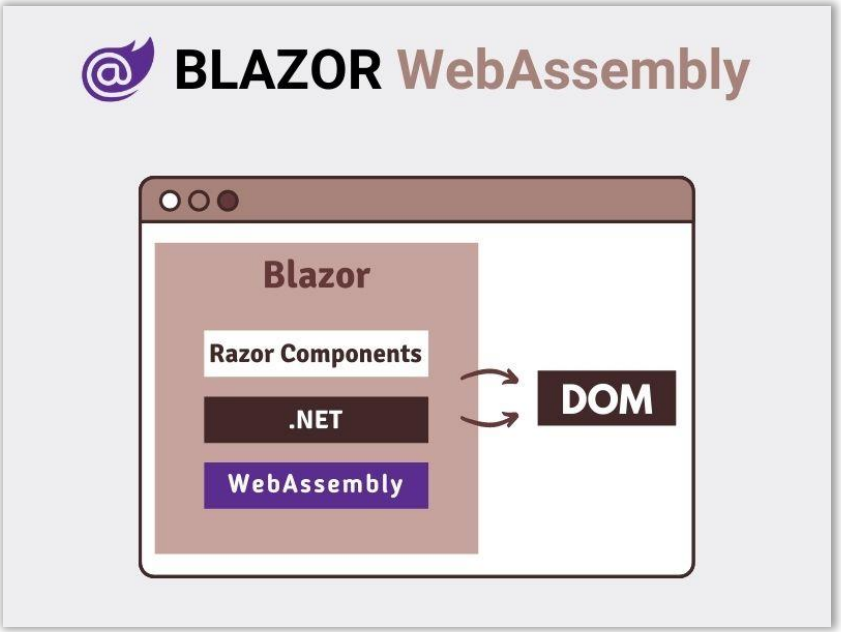
Blazor Server

Blazor Server apps run on the server where they enjoy the support of full .NET Core runtime. All the processing is done on the server and UI/DOM changes are transmitted back to the client over the SignalR connection. This two-way SignalR connection is established when the user loads the application in the browser the very first time. As your .NET code is already running on the server, you don't need to create APIs for your front-end. You can directly access services, databases, etc., and do anything you want to do on traditional server-side technology.



Blazor WebAssembly

This hosting model is a direct competitor of modern and popular SPA frameworks such as Angular, Vue, and React and this is the main reason most developers are interested to learn Blazor. It allows developers to write all front-end UI logic in C# instead of JavaScript. In this hosting model, application DLLs, any dependencies, and a small size Mono .NET runtime are downloaded to the client in the first request. Once everything is available at the client, the Mono runtime loads and executes the application code.



Blazor App Default Project Structure

File/Folder	App Type	Description
Program.cs	Server App, WebAssembly App	This file contains the **Main** method which is the entry point of the project that sets up the ASP.NET Core host.
Startup.cs	Server App	Contains the app's startup logic.
App.razor	Server App, WebAssembly App	The root component of the app that sets up client-side routing using the Router component.
_Imports.razor	Server App, WebAssembly App	Includes common Razor directives to include in the app's components (.razor), such as @using directives for namespaces.
appsettings.json	Server App, WebAssembly App	Provide configuration settings for the app.
Pages	Server App, WebAssembly App	This folder contains the routable components/pages
Pages/_Host.cshtml	Server App	The root page of the app implemented as a Razor Page.
Pages/Index.razor	Server App, WebAssembly App	This component renders the home page of the app.
Pages/Error.razor	Server App, WebAssembly App	Rendered when an unhandled exception occurs in the app.
Pages/Counter.razor	Server App, WebAssembly App	Implements the Counter page.
Pages/FetchData.razor	Server App, WebAssembly App	Implements the Fetch data page.
wwwroot	Server App, WebAssembly App	This folder contains the static files such as images, fonts, icons, CSS and JavaScript files, etc.
wwwroot/index.html	WebAssembly App	The root page of the app implemented as a HTML Page.
Shared	Server App, WebAssembly App	Contains shared components and related stylesheets.
Shared/MainLayout.razor	Server App, WebAssembly App	The MainLayout component is the app's default layout component created from Blazor project template.
Shared/NavMenu.razor	Server App, WebAssembly App	The default NavMenu component that generates the bootstrap navigation bar.
Properties/launchSettings.json	Server App, WebAssembly App	Contains development environment configurations.

Blazor Route Templates

Example	Description
@page “/”	Navigate to home page.
@page “/counter”	Navigate to counter page.
@page “/counter1” @page “/counter2”	Multiple route templates are allowed.
@page “/products/**{Id}**”	Navigate to products page and receive **Id** as route parameter. e.g. /products/5
@page “/products/**{type]/{page}**”	Multiple route parameters are supported e.g. /products/5/1
@page “/products/**{Id:int}**”	Route parameter constraints can be defined. In this example **Id** parameter value must be **int** type value.
@page “/products/**{type:guid]/{page:int}**”	Multiple route parameters with constraints are supported

Blazor Route Constraints

Constraint	Example	Example Matches
bool	{active:bool}	true, FALSE
datetime	{dob:datetime}	2016-12-31, 2016-12-31 7:32pm
decimal	{price:decimal}	49.99, -1,000.01
double	{weight:double}	1.234, -1,001.01e8
float	{weight:float}	1.234, -1,001.01e8
guid	{id:guid}	CD2C1638-1638-72D5-1638-DEADBEEF1638, {CD2C1638-1638-72D5-1638-DEADBEEF1638}
int	{id:int}	123456789, -123456789
long	{ticks:long}	123456789, -123456789

Blazor Directives

Directive	Description
@attributes	Allows a component to render non-declared attributes.
@bind	Allows a component to use data binding features.
@bind:format	Used to format data bound DateTime into string.
@bind-{PROPERTY}	Used to bind properties of other components.
@on{EVENT}	Provides event handling features for components.
@on{EVENT}:preventDefault	Prevents the default action for the event.
@on{EVENT}:stopPropagation	Stops event propagation for the event.
@key	Causes the components diffing algorithm to guarantee preservation of elements or components based on the key's value.
@ref	Provide a way to reference a component instance so that you can issue commands to that instance.
@typeparam	Declares a generic type parameter for the generated component class.

Blazor Common Services

Member	Lifetime	Description
HttpClient	Scoped	Provides methods for sending HTTP requests and receiving HTTP responses from a resource identified by a URI.
IJSRuntime	**WebAssembly:** Singleton **Server:** Scoped	Represents an instance of a JavaScript runtime where JavaScript calls are dispatched.
NavigationManager	**WebAssembly:** Singleton **Server:** Scoped	Contains helpers for working with URIs and navigation state.

Blazor Form Components

Component	Render As	Description
InputCheckbox	<input type="checkbox">	An input component for editing Boolean values.
InputDate<TValue>	<input type="date">	An input component for editing date values.
InputFile	<input type="file">	The InputFile component renders as an HTML input of type file.
InputNumber<TValue>	<input type="number">	An input component for editing numeric values.
InputRadio	<input type="radio">	The InputRadio component renders as an HTML input of type radio.
InputRadioGroup	<input type="radio">	Used to group multiple InputRadio components together.
InputSelect<TValue>	<select>	A dropdown selection component.
InputText	<input>	An input component for editing String values.
InputTextArea	<textarea>	A multiline input component for editing String values.
ValidationSummary		The ValidationSummary component summarizes all validation messages
ValidationMessage<TValue>		Displays a list of validation messages for a specified field

Blazor NavigationManager Service Helpers

Member	Description
Uri	Gets the current absolute URI.
BaseUri	Gets the base URI (with a trailing slash)
NavigateTo	Navigates to the specified URI. If forceLoad is true
ToAbsoluteUri	Converts a relative URI into an absolute URI.
LocationChanged	An event that fires when the navigation location has changed.

Blazor Events

Event	Class	Document Object Model (DOM) Events
Clipboard	ClipboardEventArgs	oncut, oncopy, onpaste
Drag	DragEventArgs	ondrag, ondragstart, ondragenter, ondragleave, ondragover, ondrop, ondragend
Error	ErrorArgs	onerror
Focus	FocusEventArgs	onfocus, onblur, onfocusin, onfocusout
Input	ChangeEventArgs	onchange, oninput
Keyboard	KeyboardEventArgs	onkeydown, onkeypress, onkeyup
Mouse	MouseEventArgs	onclick, oncontextmenu, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout
Mouse pointer	PointerEventArgs	onpointerdown, onpointerup, onpointercancel, onpointermove, onpointerover, onpointerout, onpointerenter, onpointerleave, ongotpointercapture, onlostpointercapture
Mouse wheel	WheelEventArgs	onwheel, onmousewheel
Progress	ProgressEventArgs	onabort, onload, onloadend, onloadstart, onprogress, ontimeout
Touch	TouchEventArgs	ontouchstart, ontouchend, ontouchmove, ontouchenter, ontouchleave, ontouchcancel
Event	EventArgs	**General:** onactivate, onbeforeactivate, onbeforedeactivate, ondeactivate, onfullscreenchange, onfullscreenerror, onloadeddata, onloadedmetadata, onpointerlockchange, onpointerlockerror, onreadystatechange, onscroll **Clipboard**: onbeforecut, onbeforecopy, onbeforepaste **Input**: oninvalid, onreset, onselect, onselectionchange, onselectstart, onsubmit **Media**: oncanplay, oncanplaythrough, oncuechange, ondurationchange, onemptied, onended, onpause, onplay, onplaying, onratechange, onseeked, onseeking, onstalled, onstop, onsuspend, ontimeupdate, ontoggle, onvolumechange, onwaiting

Blazor Built-in Components

Component	Description
App	The root component of the app that sets up client-side routing using the Router component
Router	Enables routing to Razor components in a Blazor app
Found	Gets or sets the content to display when a match is found for the requested route.
NotFound	Gets or sets the content to display when no match is found for the requested route.
Navigating	Get or sets the content to display when asynchronous navigation is in progress.
NavMenu	The default NavMenu component generates the bootstrap navigation bar.
NavLink	Renders an anchor tag, automatically toggling its 'active' class based on whether its 'href' matches the current URI.
LayoutView	Displays the specified content inside the specified layout and any further nested layouts.
MainLayout	The MainLayout component is the app's default layout component created from Blazor project template.
CascadingValue	Provides a cascading value to all descendant components.
Virtualize	Provides functionality for rendering a virtualized list of items.
Authentication	Handles remote authentication operations
AuthorizeView	Selectively displays UI content depending on whether the user is authorized.
Authorized	The content that will be displayed if the user is authorized.
NotAuthorized	The content that will be displayed if the user is not authorized.
Authorizing	The content that will be displayed while asynchronous authorization is in progress.

ezzylearning.net

An online tutorial website that provides a comprehensive array of tutorials on topics ranging from classic Windows based Application development to the latest in ASP.NET Core, Blazor, ASP.NET MVC, AngularJS, C# and many more.