



@RobertHaken 

Software & Cloud Architect | Microsoft MVP: Development | HAVIT, s.r.o.



# Blazor

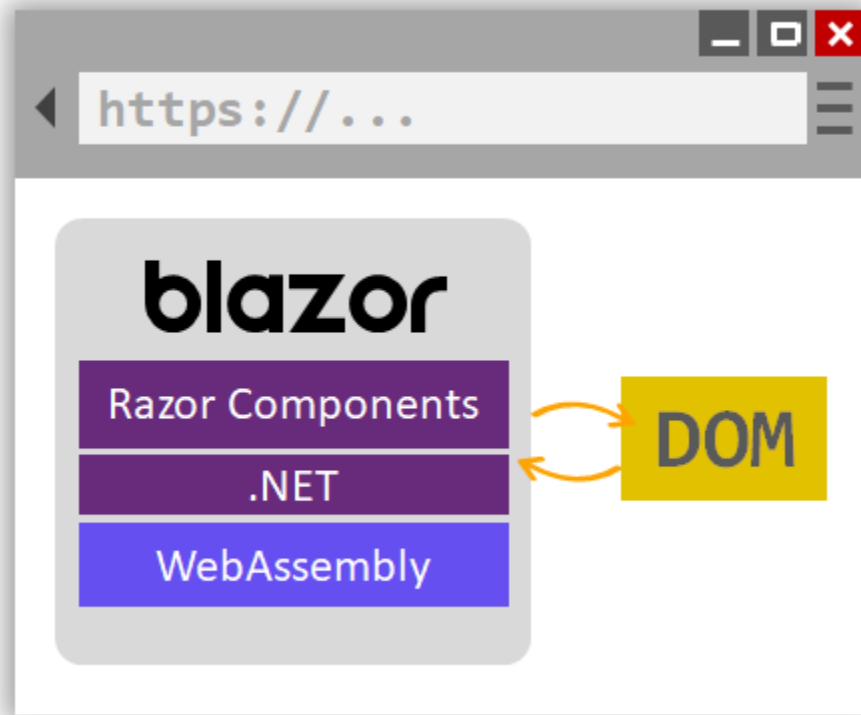
a novinky v ASP.NET Core 3.0

<https://askme.blazor.cz>

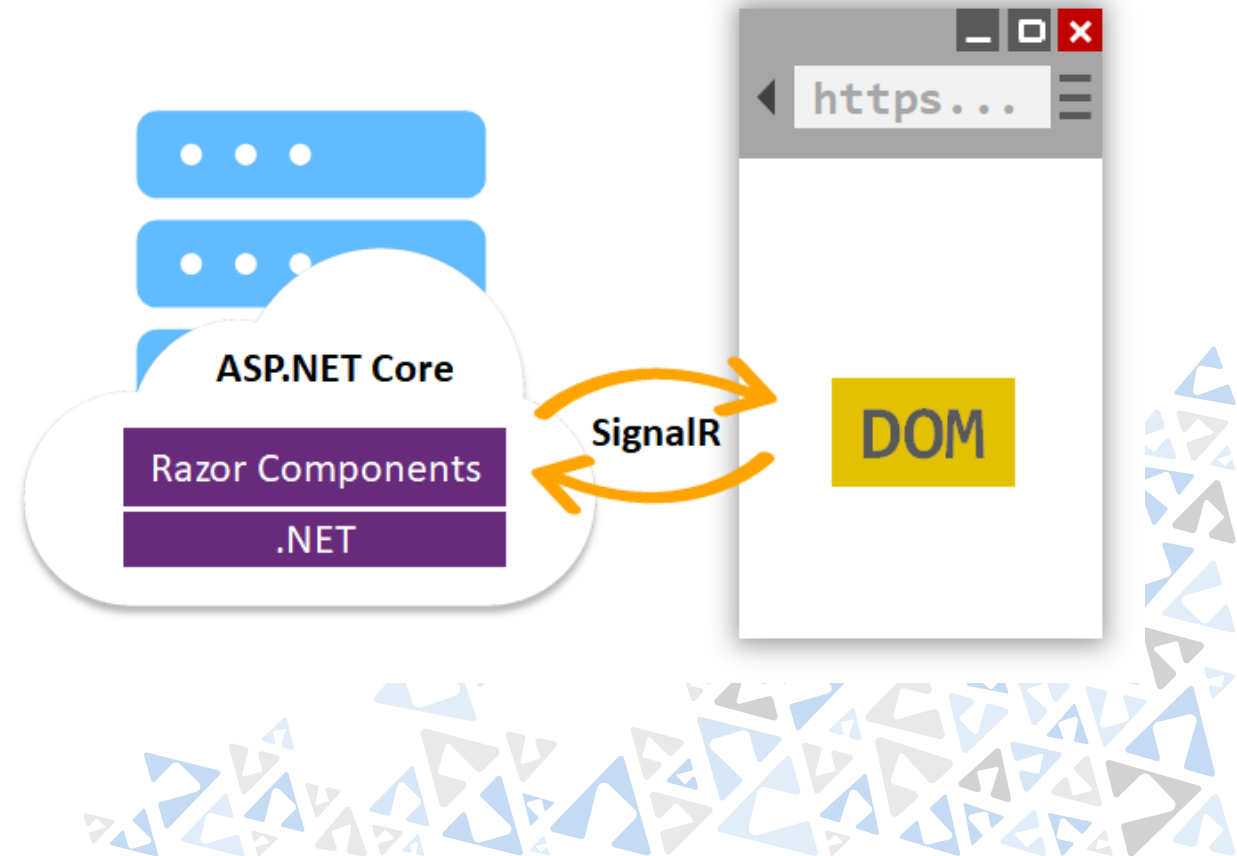


# Blazor

Client-side hosting model



Server-side hosting model



# Supported platforms

	Blazor client-side	Blazor server-side
Google Chrome (incl. Android)	YES	YES
Microsoft Edge	YES	YES
Mozilla Firefox	YES	YES
Safari (incl. iOS)	YES	YES
Microsoft Internet Explorer	NO	v11

<https://caniuse.com/#search=webassembly>



# Blazor Now

A component model for building composable UI

Routing

Layouts

Forms and validation

Dependency injection

JavaScript interop

IntelliSense and tooling

Publishing and app size trimming (basic)

Debugging (in browser PoC)



# Blazor Plans

Authentication, Authorization

Live reloading in the browser during development

Server-side rendering

Full .NET debugging both in browsers and in the IDE

Rich IntelliSense and tooling

Publishing and app size trimming



# AskMe – Blazor Edition

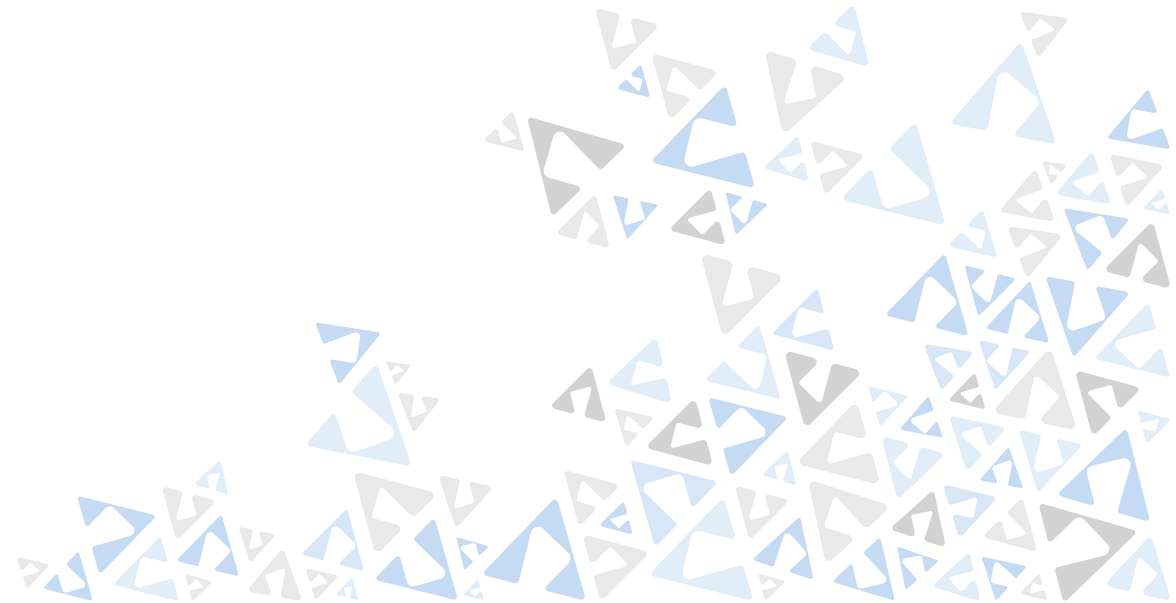


# Page = Component + Route

Namespace/Namespace/ClassName.razor

@functions => @code

@page "/route"





# Layouts

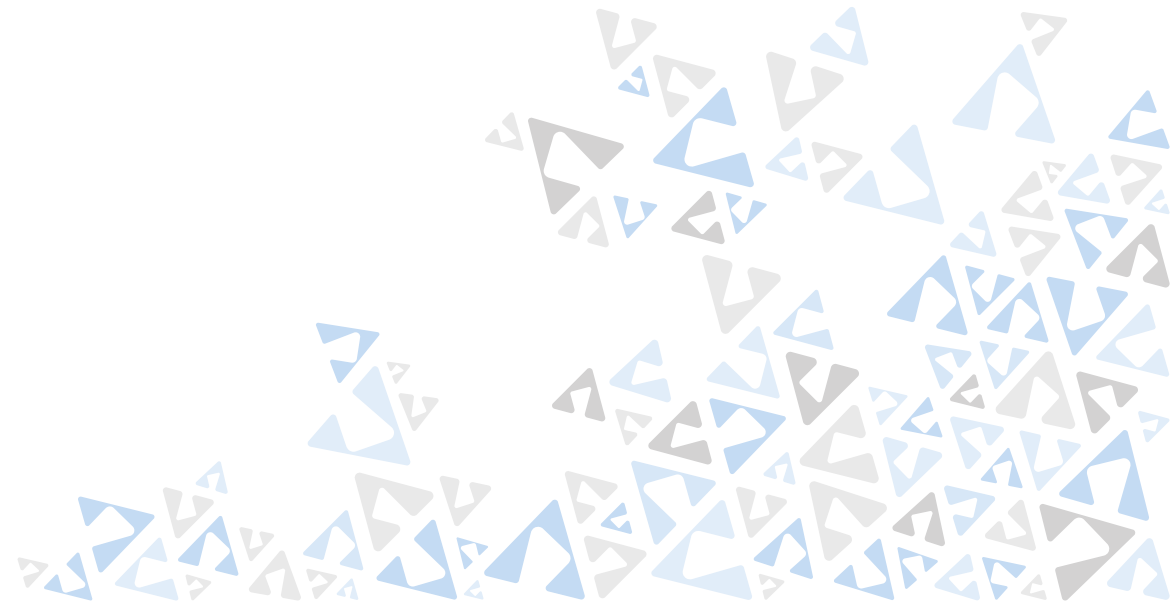
## layout

- just another component
- inherits from `LayoutComponentBase`
- Body property (`@Body` in markup)

## using the layout

- `@layout` directive (compiles to `[LayoutAttribute]`)
- `_Imports.razor`

## nested layouts



# Routing

## @page directive

```
@page "/products"
```

```
@page "/products/{ProductId}" => [Parameter]ProductId
```

```
@page "/products/{ProductId:int}"
```

## App.razor:

```
<Router AppAssembly="typeof(Program).Assembly"  
    FallbackComponent="typeof(Pages.NotFound)" />
```

```
<NavLink href="..." class="..." Match="NavLinkMatch.All | Prefix" />
```

```
IUriHelper.NavigateTo(), .OnLocationChanged, ...
```



# Components

.razor

@functions => @code

parameters - private property + [Parameter]

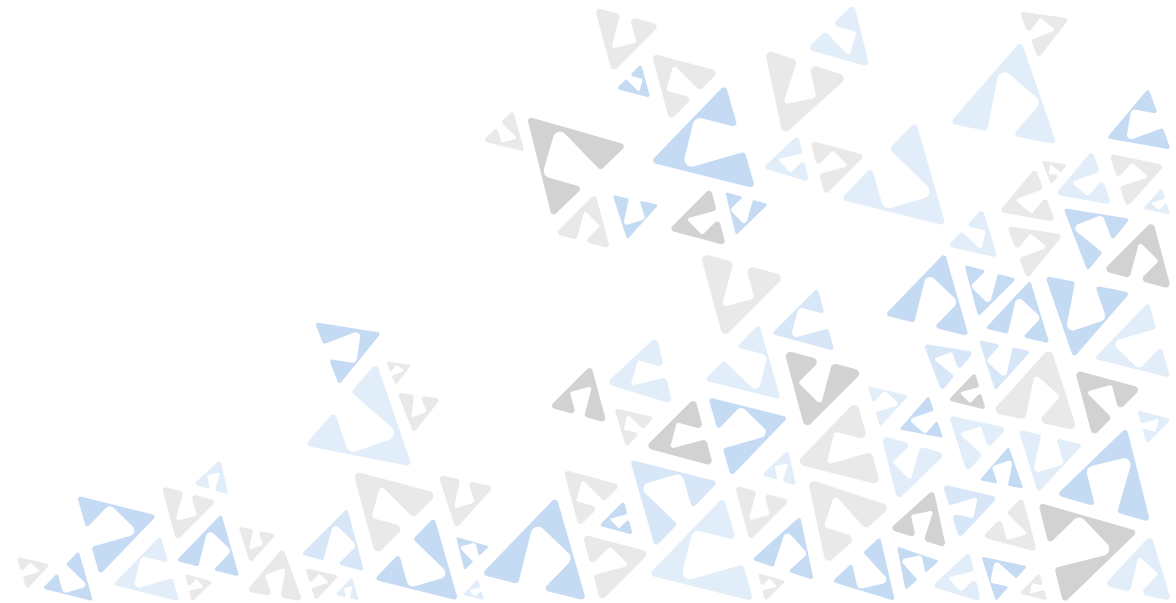
```
private RenderFragment ChildContent { get; set; }
```

@inherits

@using + \_Imports.razor

@inject

@implements



# Components Lifecycle methods

**OnInit**[Async]()

**SetParameters**(ParameterCollection parameters)

**OnParametersSet**[Async]()

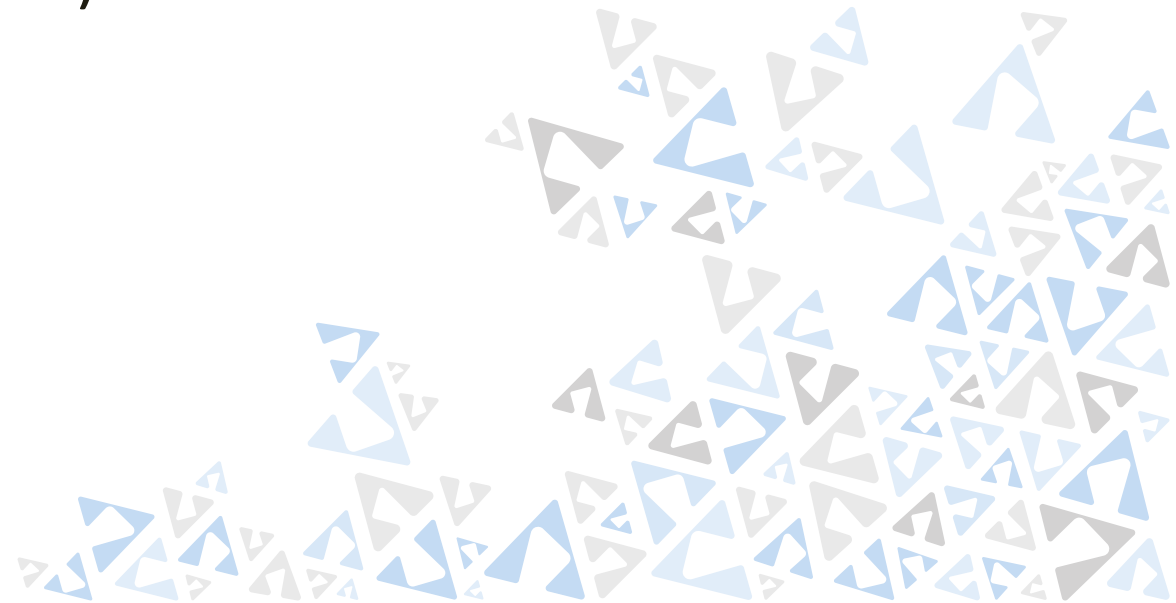
bool **ShouldRender**()

void **BuildRenderTree**(RenderTreeBuilder builder)

**OnAfterRender**[Async]()

*IDisposable.Dispose()*

*StateHasChanged()*



# Coded Components

derived from `ComponentBase`

override `BuildRenderTree(RenderTreeBuilder builder)`

`builder`

`.OpenElement(sequence, elementName); + .CloseElement();`

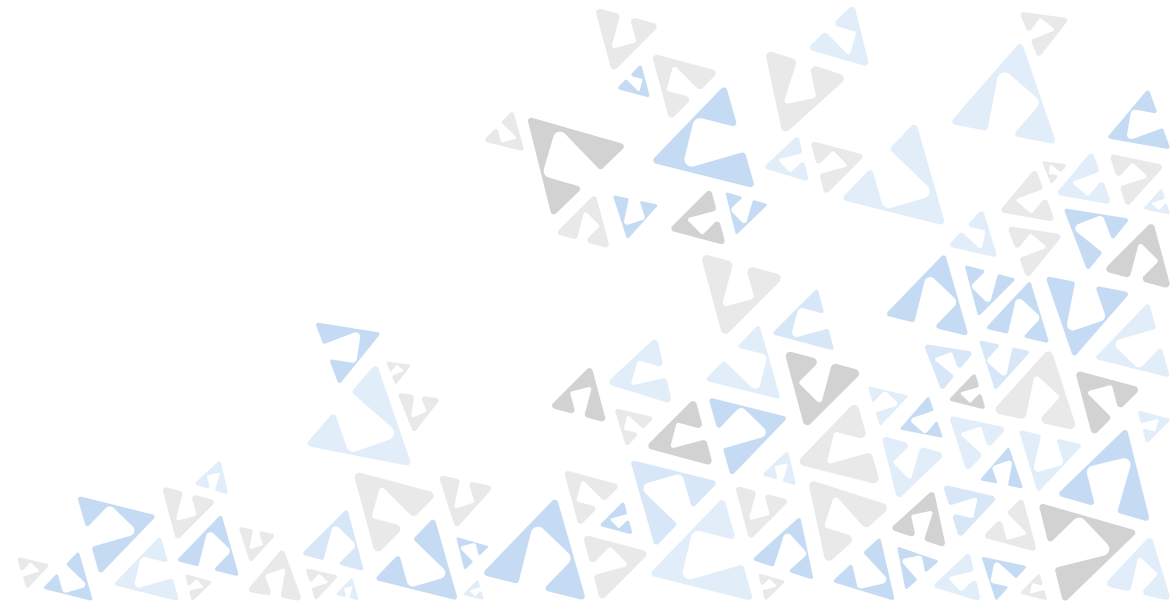
`.AddContent(sequence, ...);`

`.AddMarkupContent(sequence, ...);`

`.AddAttribute(...)`

`....`

`[Inject], [Parameter], [Route], [Layout], ...`



# Components – Event Handling

on<event>

synchronous + asynchronous

event-args

- UIEventArgs
- UIChangeEventArgs
- UIKeyboardEventArgs
- UIMouseEventArgs
- custom

onclick="@ (args => DoSomething(args, itemNumber))"

[Parameter] private EventCallback<TArgs> OnSomething { get; set; }



# Built-in Components

<NavLink /> + <Router />

<EditForm Model="@MyModel" OnValidSubmit="@HandleValidSubmit">

  <DataAnnotationsValidator />

  <ValidationSummary />

  <InputText id="name" bind-Value="@MyModel.Name" />

  <ValidationMessage For="@(() => MyModel.Name)" />

  <InputTextArea />

  <InputSelect />

  <InputNumber />

  <InputCheckBox />

  <InputDate />

</EditForm>

+ 3<sup>rd</sup> party 😊



# Data Binding

both Components & DOM elements

```
<input bind="@MyValue" />
```

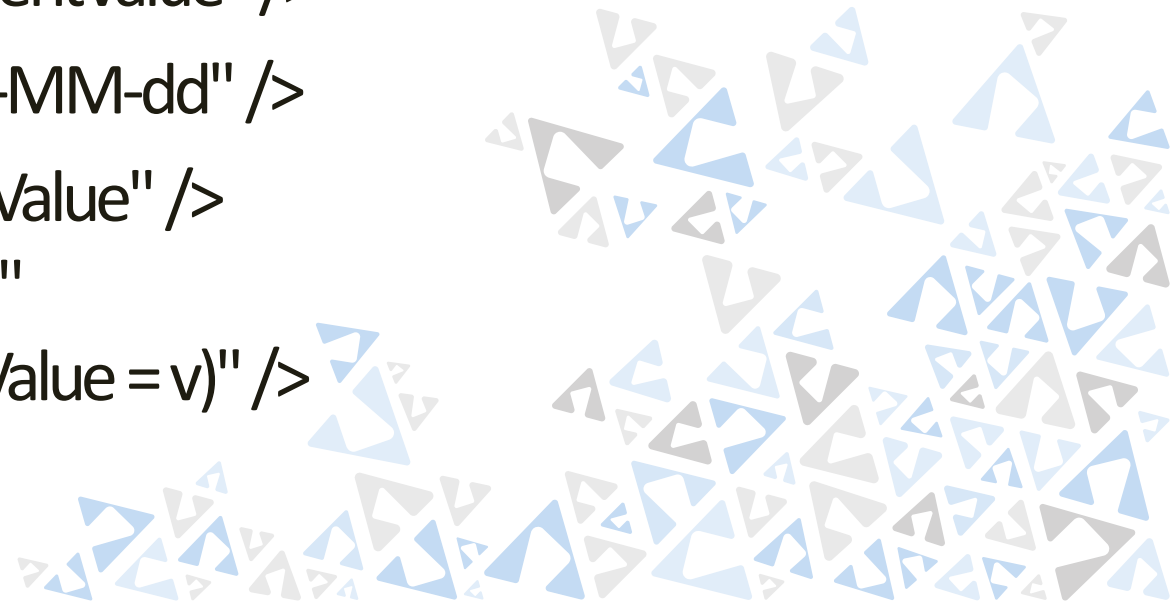
```
<input value="@MyValue"  
  onchange="@((UIChangeEventArgs __e) => MyValue = __e.Value)" />
```

```
<input type="text" bind-value-oninput="@CurrentValue" />
```

```
<input bind="@StartDate" format-value="yyyy-MM-dd" />
```

```
<MyComponent bind-MyParameter="@SomeValue" />
```

```
<MyComponent MyParameter="@SomeValue"  
  MyParameterChanged="@(v => SomeValue = v)" />
```





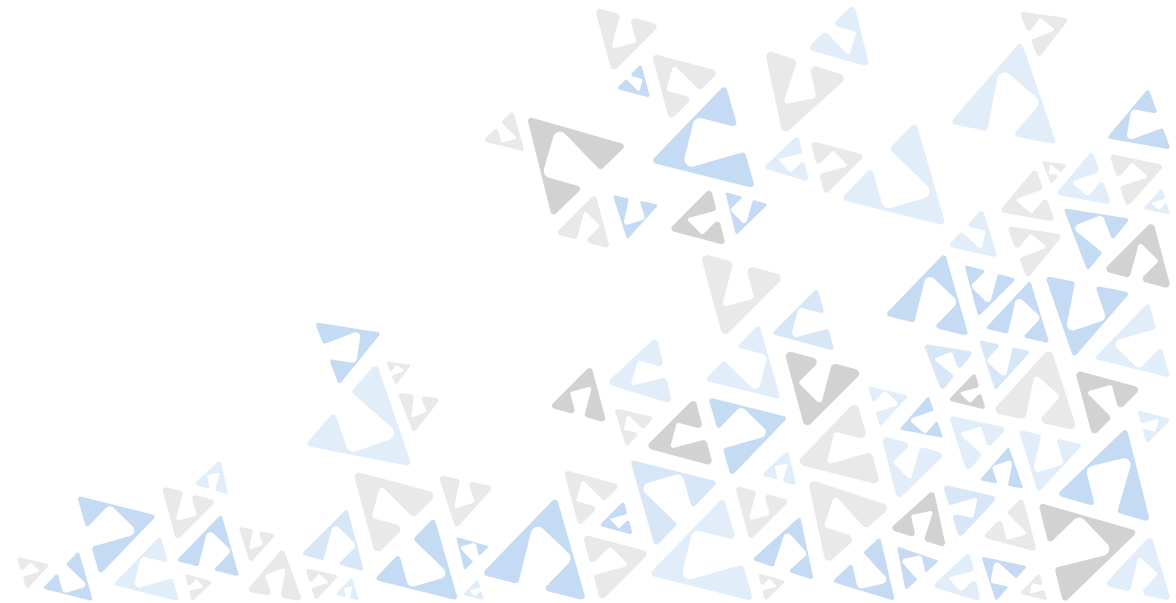
# Templated Components

RenderFragment / RenderFragment<T> parameters

@typeparam TItem

context implicit parameter: @context

<ItemTemplate Context="item"> ...@item.Xy... </ItemTemplate>



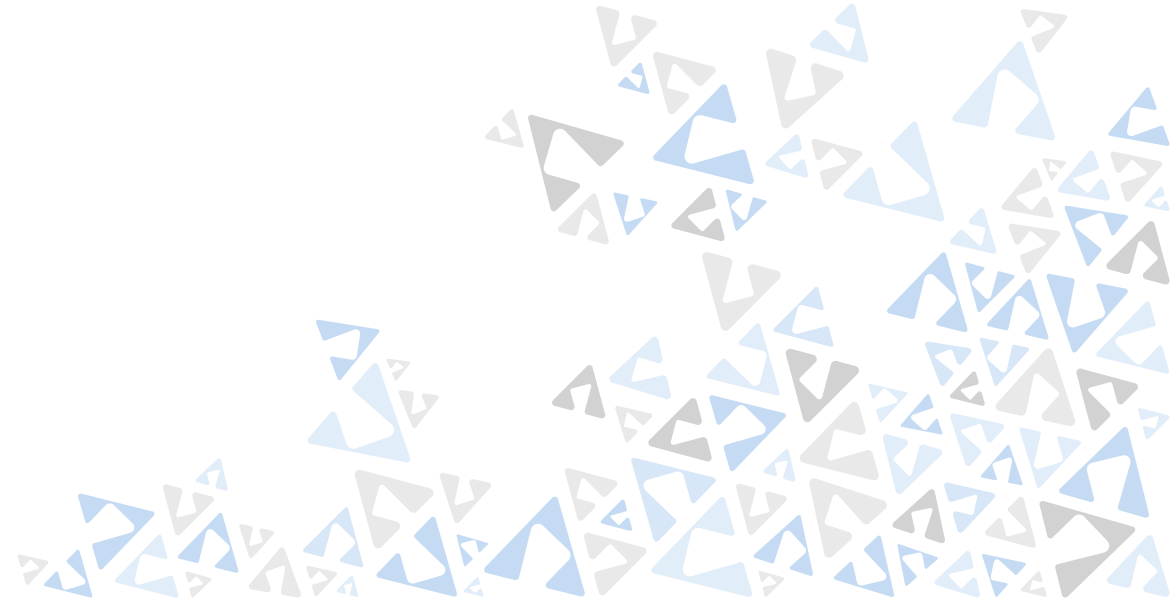
# Razor Templates

@<tag>...</tag>

```
@{  
    RenderFragment template = @<p>The time is @DateTime.Now.</p>;  
    RenderFragment<Pet> petTemplate = (pet) => @<p>Name: @pet.Name.</p>  
}
```

@template

@petTemplate(new Pet { Name = "Bzuk" })



# JavaScript Interop

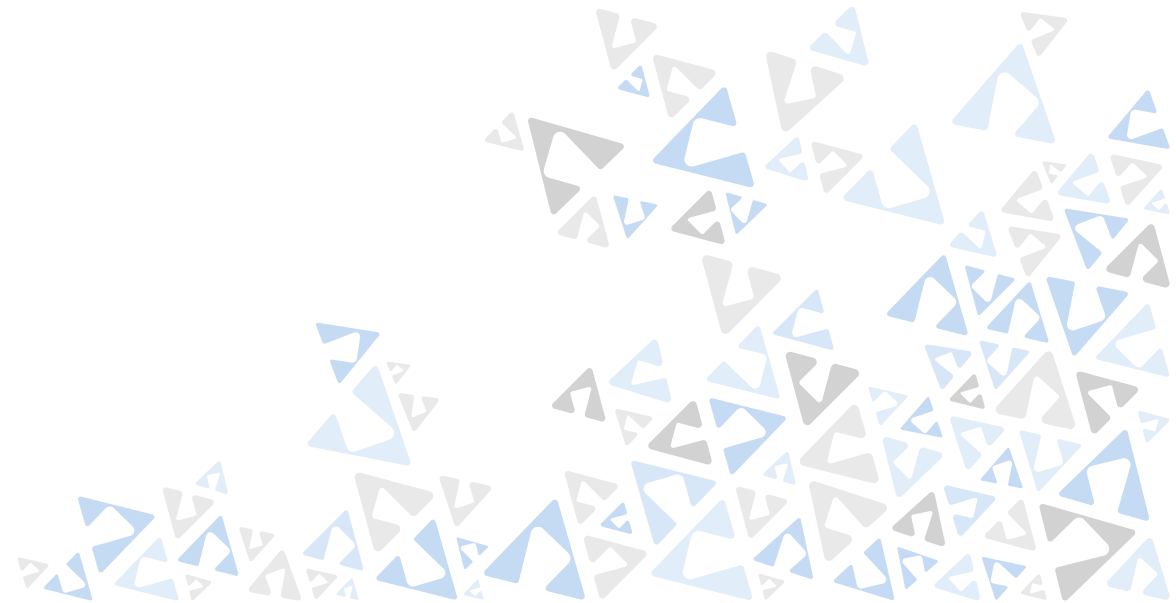
`IJSRuntime.InvokeAsync<T>(string identifier, params object[] args)`

JSON Serializable input + output

window scope

`OnAfterRenderAsync`

`ElementRef`

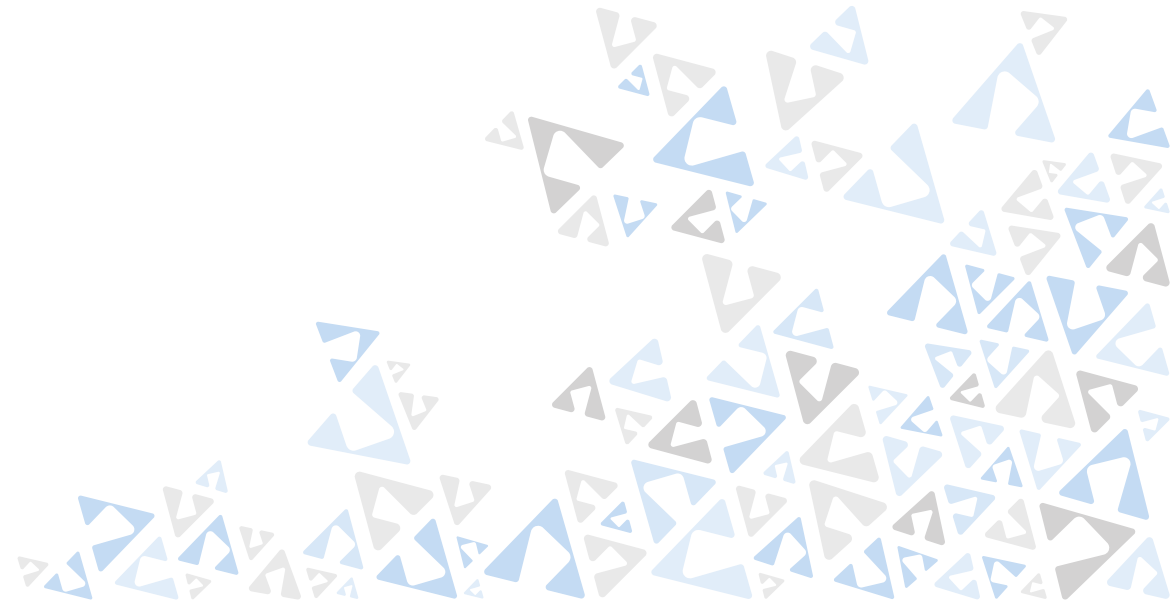


# Invoke .NET from JavaScript

```
DotNet.invokeMethod[Async]('ClassName', 'MethodName')
```

```
[JSInvokable]
```

```
DotNetObjectRef
```



# Tips & Tricks

@((MarkupString)myRawHtml)

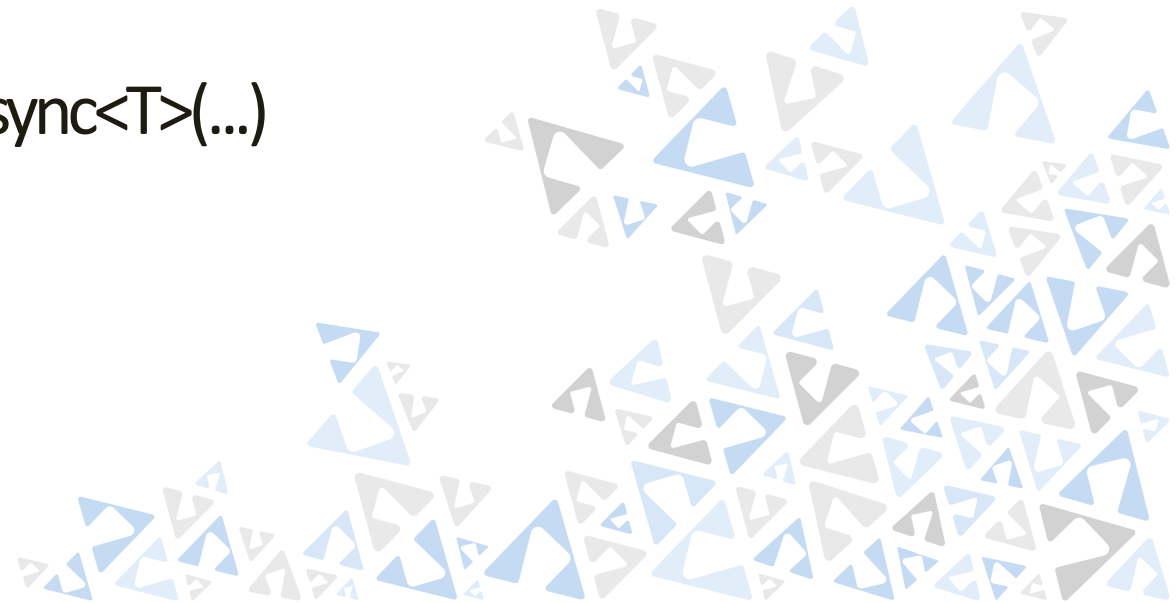
always set **Id** for <InputXy /> components

.csproj: <AppendRuntimeIdentifierToOutputPath>>false</AppendRuntimeIdentifierToOutputPath>

StateHasChanged()

Debugging

Razor Pages / MVC : Html.RenderComponentAsync<T>(...



# Novinky ASP.NET Core 3.0

ASP.NET Core 3.0 will only run on .NET Core

Microsoft.AspNetCore.App shared fw w/o Newtonsoft.Json, EF Core, CodeAnalysis

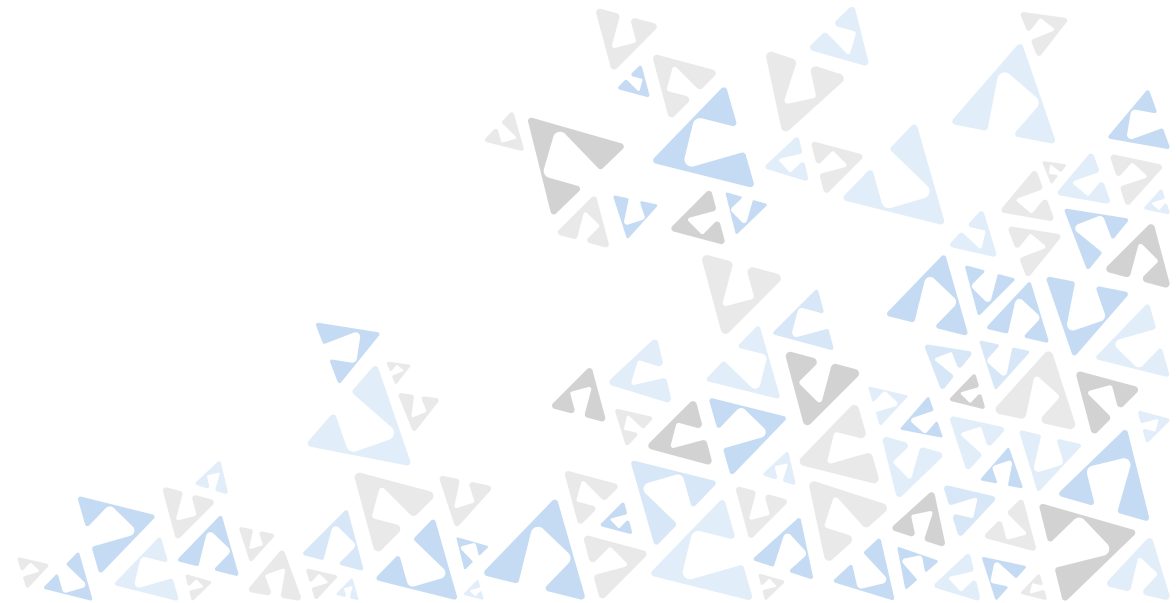
Blazor Server-Side (Razor Components)

SignalR client-to-server streaming

Pipes on HttpContext

Generic host in templates

Endpoint routing



# REFERENCES

Demos <https://github.com/hakenr/AskMe>  
<https://github.com/ridercz/AskMe>

Blog <http://knowledge-base.havit.cz/> + .eu  
[www.blazor.cz](http://www.blazor.cz)

Twitter [@RobertHaken](https://twitter.com/RobertHaken)

YouTube <https://www.youtube.com/user/HAVITcz>

