

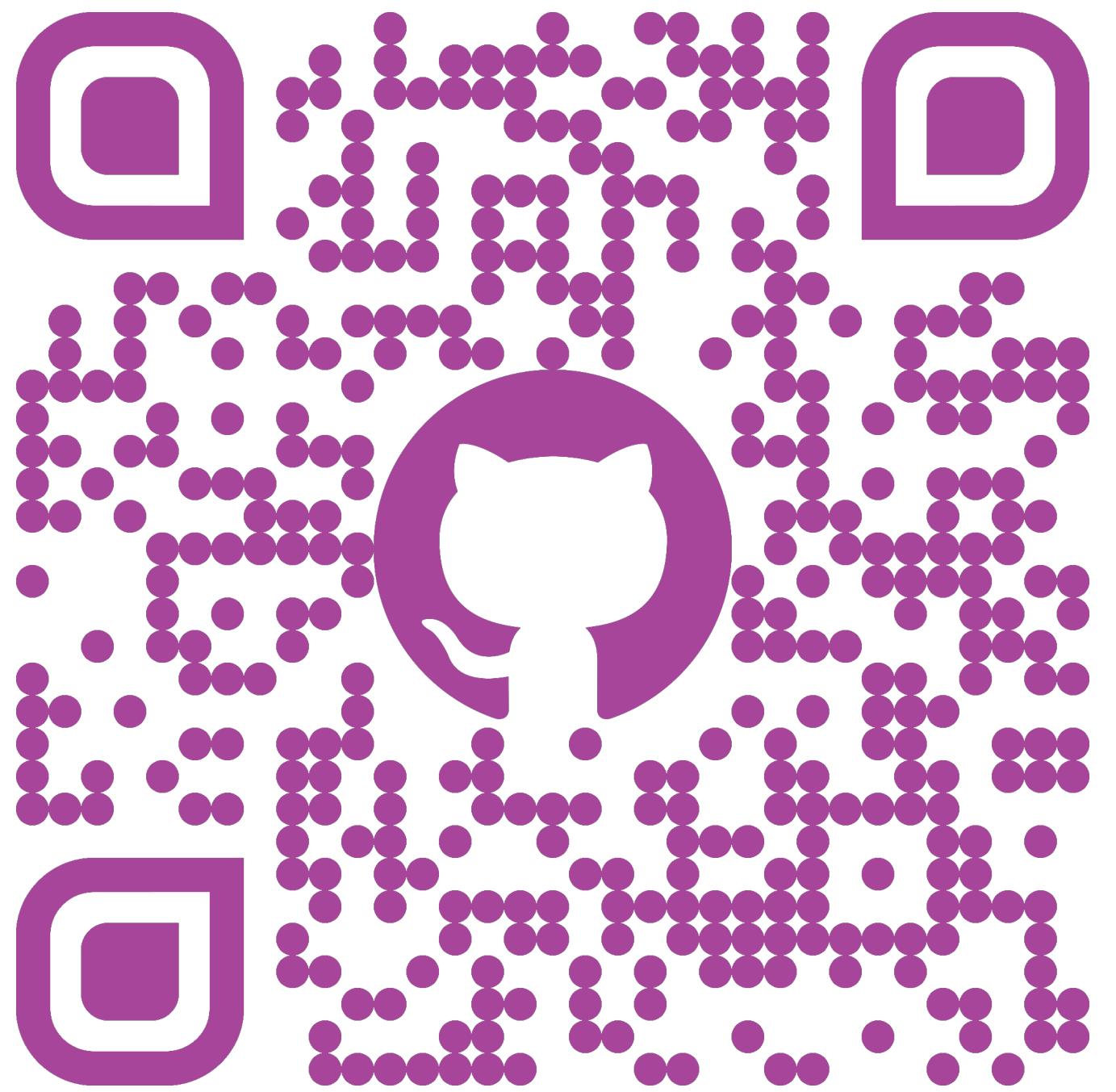
Blazor

Frontend for Backend

Круглов Георгий

о себе

- middle разработчик в Ozon Tech
- студент четвертого курса университета ИТМО
- преподаю ООП в университете ИТМО
- github.com/ronimizy



nb

- имею опыт работы с Blazor около двух лет
- активно развиваю пет-проект с фронтом на Blazor WebAssembly
- очень не люблю JS :)

roadmap

- что такое Blazor?
- язык разметки Razor
- модели хостинга Blazor
 - Blazor Server
 - Blazor WebAssembly
- библиотеки компонентов для Blazor

Relational Queries | Query + CRUD

CELECT

INSE**R**T

UPDATE

DELETE

фронтенд = ?

фронтенд

=

JS

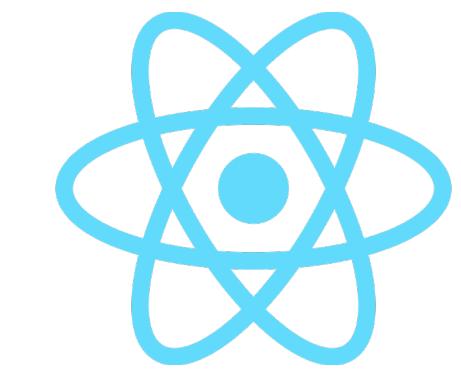
фронтенд

=

JS

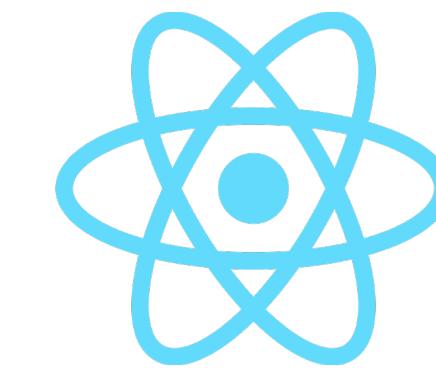
V

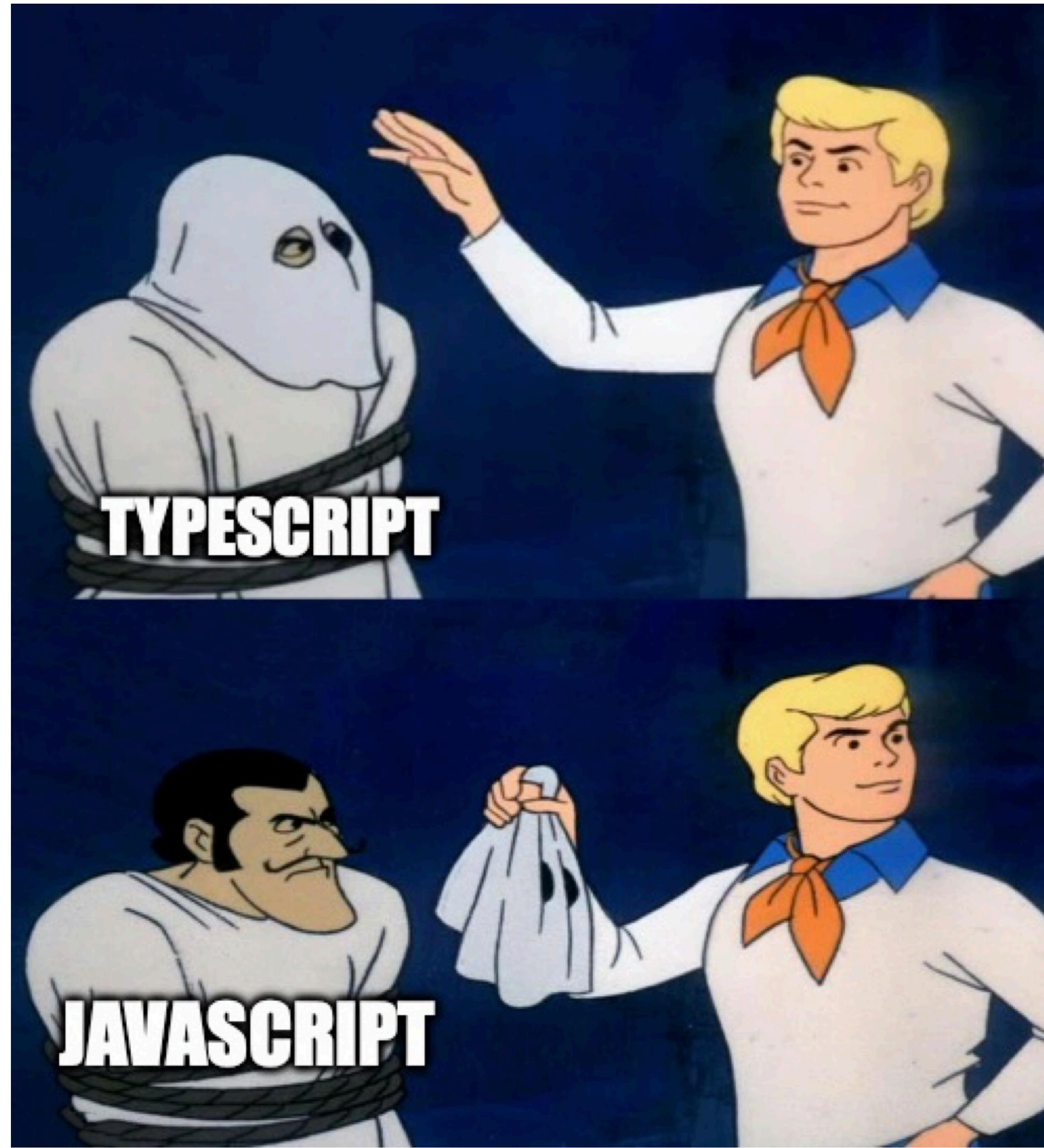
A



фронтенд

=





фронтенд Э



Что такое Blazor?

- WebUI фреймворк от Microsoft
- использует C# и Razor для генерации разметки
- позволяет создавать интерактивные и отзывчивые приложения

razor

razor

язык разметки

- использует XML синтаксис для разметки
поддерживает HTML теги и Razor компоненты
- использует C# как язык управления шаблонизацией
- результатом шаблонизации является HTML разметка

razor

```
<PageTitle>Counter</PageTitle>
```

```
<h1>Counter</h1>
```

```
<CompositeCounterComponent/>
```

razor

```
public partial class Counter : ComponentBase
{
    protected override void BuildRenderTree(RenderTreeBuilder builder)
    {
        builder.OpenComponent<PageTitle>(0);

        builder.AddAttribute(
            1,
            "ChildContent",
            (RenderFragment)(builder2 => builder2.AddContent(2, "Counter")));

        builder.CloseComponent();

        builder.AddMarkupContent(3, "\n\n");

        builder.AddMarkupContent(4, "<h1>Counter</h1>\n\n");

        builder.OpenComponent<CompositeCounterComponent>(6);
        builder.CloseComponent();
    }
}
```

razor

просмотр сгенерированных классов

добавьте в PropertyGroup проекта

```
<EmitCompilerGeneratedFiles>true</EmitCompilerGeneratedFiles>
```

сгенерированные классы будут находиться в
obj/\$(Configuration)/\$(TargetFramework)/generated

razor

интерполяция значений

```
@_user.Name
```

```
@@username:
```

```
@("<span>Hello World</span>") = <span>Hello World</span>;
```

razor

условные операторы

```
@if (IsAuthenticated)
{
    <button>
        logout
    </button>
}
else
{
    <button>
        login
    </button>
}
```

```
protected override void BuildRenderTree(
    RenderTreeBuilder builder)
{
    if (IsAuthenticated)
    {
        builder.OpenElement(0, "button");
        builder.AddMarkupContent(1, "logout");
        builder.CloseElement();
    }
    else
    {
        builder.OpenElement(3, "button");
        builder.AddMarkupContent(4, "login");
        builder.CloseElement();
    }
}
```

razor

ЦИКЛЫ

```
for (var i = 0; i < _count; i++)
{
    builder.OpenElement(3, "li");
    builder.AddContent(4, i);
    builder.CloseElement();
}

@for (var i = 0; i < _count; i++)
{
    <li>@i</li>
}

@while (_count is not 0)
{
    <li> @_count</li>
    _count--;
}
```

```
while (_count is not 0)
{
    builder.OpenElement(5, "li");
    builder.AddContent(6, _count);
    builder.CloseElement();
    _count--;
}
```

razor

блоки кода

```
    var index = 0;  
    builder.AddMarkupContent(7, "index defined here");  
    while (_count is not 0)  
    {  
        var i = index++;  
        builder.OpenElement(8, "li");  
        builder.AddContent(9, i);  
        builder.CloseElement();  
        _count--;  
    }  
  
@{  
    var index = 0;  
    @: index defined here  
}  
  
@while (_count is not 0)  
{  
    var i = index++;  
    <li>@i</li>  
    _count--;  
}
```

razor

try/catch

```
@try
{
    if (Throw)
    {
        throw new Exception(
            "Exception has been thrown");
    }
    @:No exception has been thrown
}
catch (Exception e)
{
    @e.Message
}
```

```
try
{
    if (Throw)
    {
        throw new Exception(
            "Exception has been thrown");
    }
    builder.AddMarkupContent(
        0,
        "No exception has been thrown");
}
catch (Exception e)
{
    builder.AddContent(1, e.Message);
}
```

razor

директивы типа

```
@using System.Text  
@attribute [Sample]  
@implements IDisposable  
@inherits LayoutComponentBase  
@typeparam T
```

razor

блок @code

```
protected override void BuildRenderTree(
    RenderTreeBuilder builder)
{
    for (var i = 0; i < _count; i++)
    {
        <li>@i</li>
    }
}

private int _count;

private void AddValue()
{
    _count++;
}

}

protected override void BuildRenderTree(
    RenderTreeBuilder builder)
{
    for (var i = 0; i < _count; i++)
    {
        builder.OpenElement(1, "li");
        builder.AddContent(2, i);
        builder.CloseElement();
    }
}

private int _count;

private void AddValue()
{
    _count++;
}
```

razor dependency injection

```
protected override void BuildRenderTree(  
    RenderTreeBuilder builder)  
{  
    builder.AddContent(0, MessageService.GetMessage());  
}  
  
[Inject]  
private MessageService MessageService { get; set; }  
  
@using SpbDotNet.Blazor.Components.Services  
@inject MessageService MessageService  
  
@MessageService.GetMessage()
```

razor page

```
@page "/parameter/{value:int}"  
  
@code {  
  
    [Parameter]  
    public int Value { get; set; }  
  
}
```

```
[Route("/parameter/{value:int}")]
public partial class ParameterPage : ComponentBase
{
    protected override void BuildRenderTree(
        RenderTreeBuilder builder) { }

    [Parameter]
    public int Value { get; set; }
}
```

razor

NavigationManager

```
@inject NavigationManager NavigationManager

@code {

    private void MoveNext()
    {
        NavigationManager.NavigateTo($"/{parameter}/{Value + 1}");
    }

}
```

razor

data binding

InnerCounterComponent.razor

```
<tr>
  <td>
    inner count = @InnerCount
  </td>
  <td>
    <button @onclick="Increment">
      increment inner
    </button>
  </td>
</tr>

@code {
  [Parameter]
  public int InnerCount { get; set; }

  private void Increment()
    => InnerCount++;
}
```

CompositeCounterComponent.razor

```
<InnerCounterComponent InnerCount="ParentCount" />

@code {
  [Parameter]
  public int ParentCount { get; set; }

  private void Increment()
  {
    ParentCount++;
  }
}
```

razor

one-way binding

InnerCounterComponent.razor

```
<tr>
  <td>
    inner count = @InnerCount
  </td>
  <td>
    <button @onclick="Increment">
      increment inner
    </button>
  </td>
</tr>

@code {
  [Parameter]
  public int InnerCount { get; set; }

  private void Increment()
    => InnerCount++;
}
```

CompositeCounterComponent.razor

```
<tr>
  <td>
    parent count = @ParentCount
  </td>
  <td>
    <button @onclick="@Increment">
      increment parent
    </button>
  </td>
</tr>

<InnerCounterComponent InnerCount="ParentCount" />

@code {
  [Parameter]
  public int ParentCount { get; set; }

  private void Increment()
  {
    ParentCount++;
  }
}
```

razor

one-way binding

Counter

This counter consists of two components.

One is parent, other is inner.

Each of them can increment the value, but increments of inner component does not apply to parent, as there is no two-way binding between the values

parent count = 0

inner count = 0

razor

EventCallback

```
<button @onclick="HandleButtonClicked">
    A
</button>

<button @onclick="OnButtonBClicked">
    B
</button>

@code {

    private int _count;

    [Parameter]
    public EventCallback<int> OnButtonAClicked { get; set; }

    [Parameter]
    public EventCallback OnButtonBClicked { get; set; }

    private async Task HandleButtonClicked()
    {
        _count++;
        await OnButtonAClicked.InvokeAsync(_count);
    }
}
```

razor

two-way binding

BoundInnerCounterComponent.razor

```
<tr>
    <td>
        inner count = @InnerCount
    </td>
    <td>
        <button @onclick="Increment">
            increment inner
        </button>
    </td>
</tr>

@code {
    [Parameter]
    public int InnerCount { get; set; }

    [Parameter]
    public EventCallback<int> InnerCountChanged { get; set; }

    private Task Increment()
    {
        InnerCount++;
        return InnerCountChanged.InvokeAsync(InnerCount);
    }
}
```

BoundCompositeCounterComponent.razor

```
<tr>
    <td>
        parent count = @ParentCount
    </td>
    <td>
        <button @onclick="@Increment">
            increment parent
        </button>
    </td>
</tr>

<BoundInnerCounterComponent @bind-InnerCount="ParentCount"/>

@code {
    [Parameter]
    public int ParentCount { get; set; }

    private void Increment()
    {
        ParentCount++;
    }
}
```

razor

two-way binding

```
protected override void BuildRenderTree(RenderTreeBuilder builder)
{
    builder.OpenComponent<BoundInnerCounterComponent>(12);

    builder.AddComponentParameter(
        13,
        "InnerCount",
        RuntimeHelpers.TypeCheck(ParentCount));

    builder.AddComponentParameter(
        14,
        "InnerCountChanged",
        RuntimeHelpers.TypeCheck(EventCallback.Factory.Create(
            this,
            RuntimeHelpers.CreateInferredEventCallback(this, i => ParentCount = i, ParentCount))));

    builder.CloseComponent();
    builder.CloseElement();
}
```

razor

two-way binding

BoundCounter

This counter consists of two components.

One is parent, other is inner.

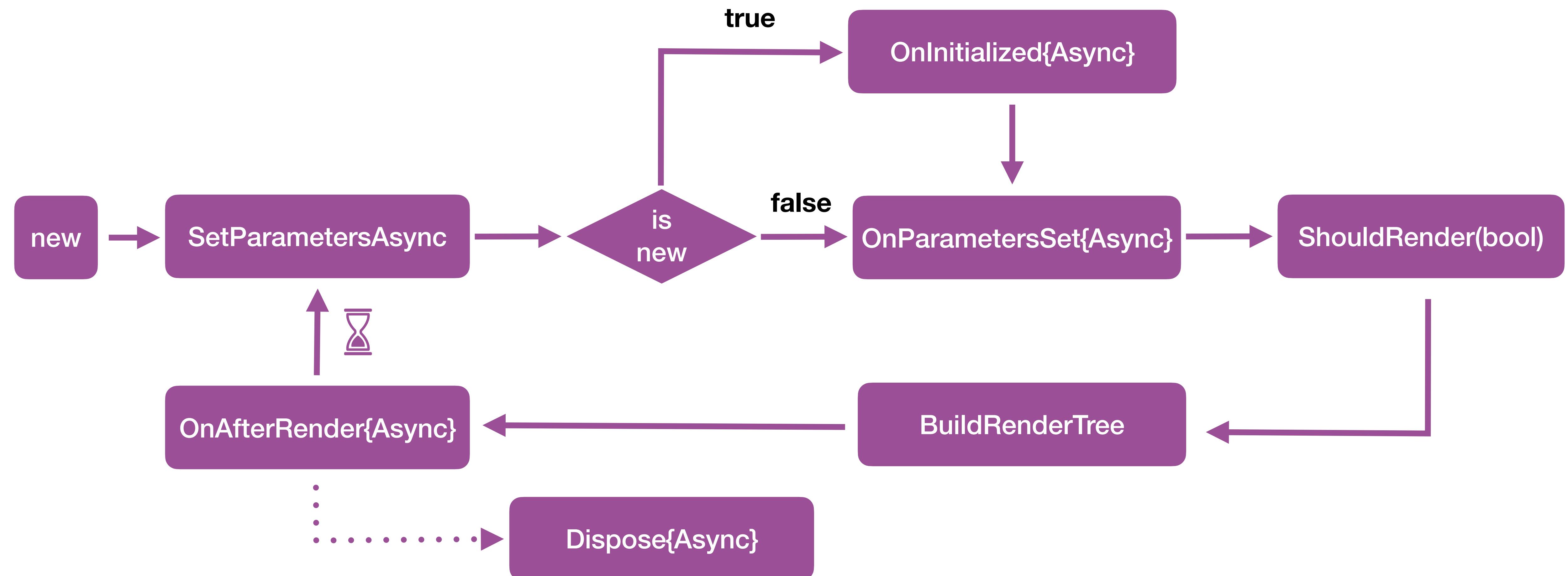
Each of them can increment the value, and inner increments are propagating to parent, as there is a two-way binding via @bind-Count

parent count = 0 

inner count = 0

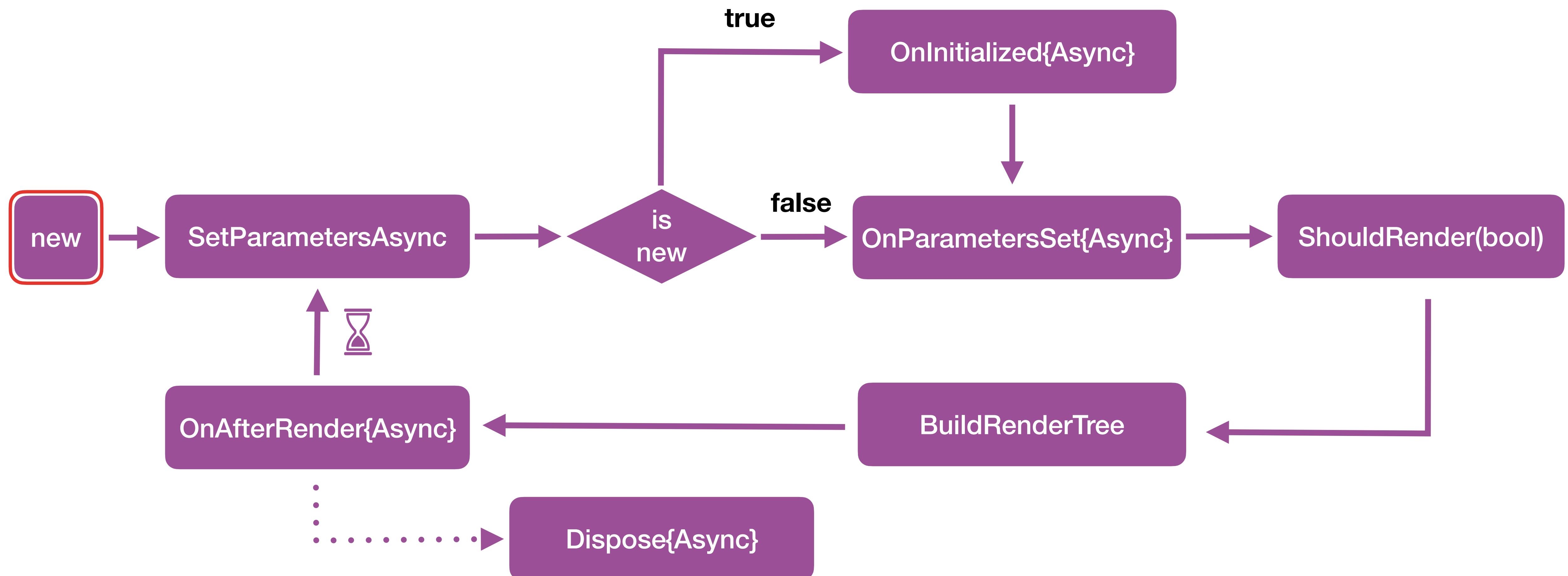
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



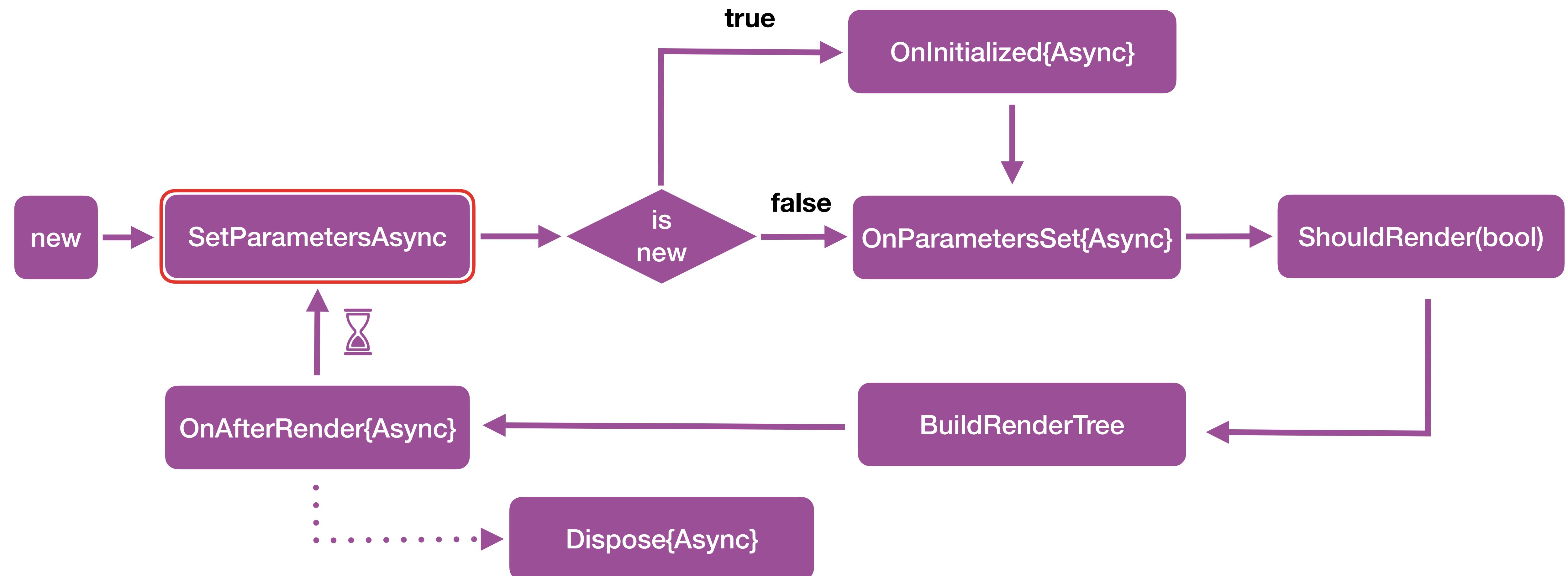
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



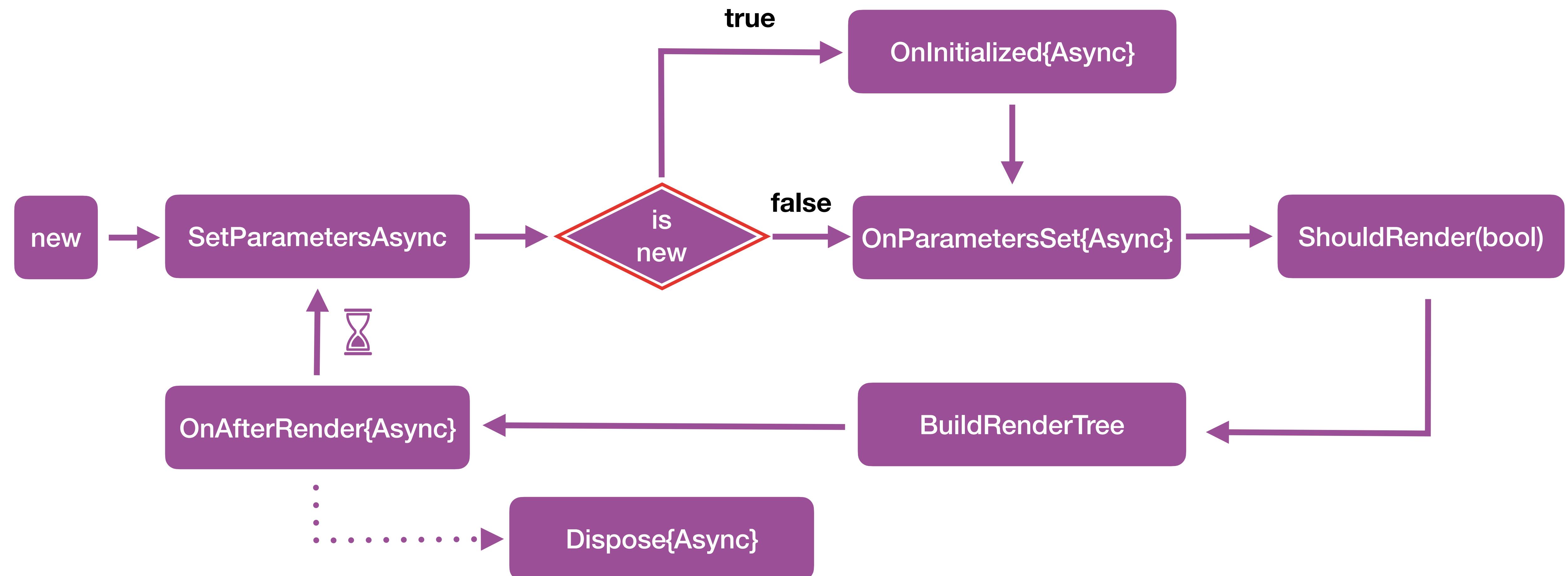
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



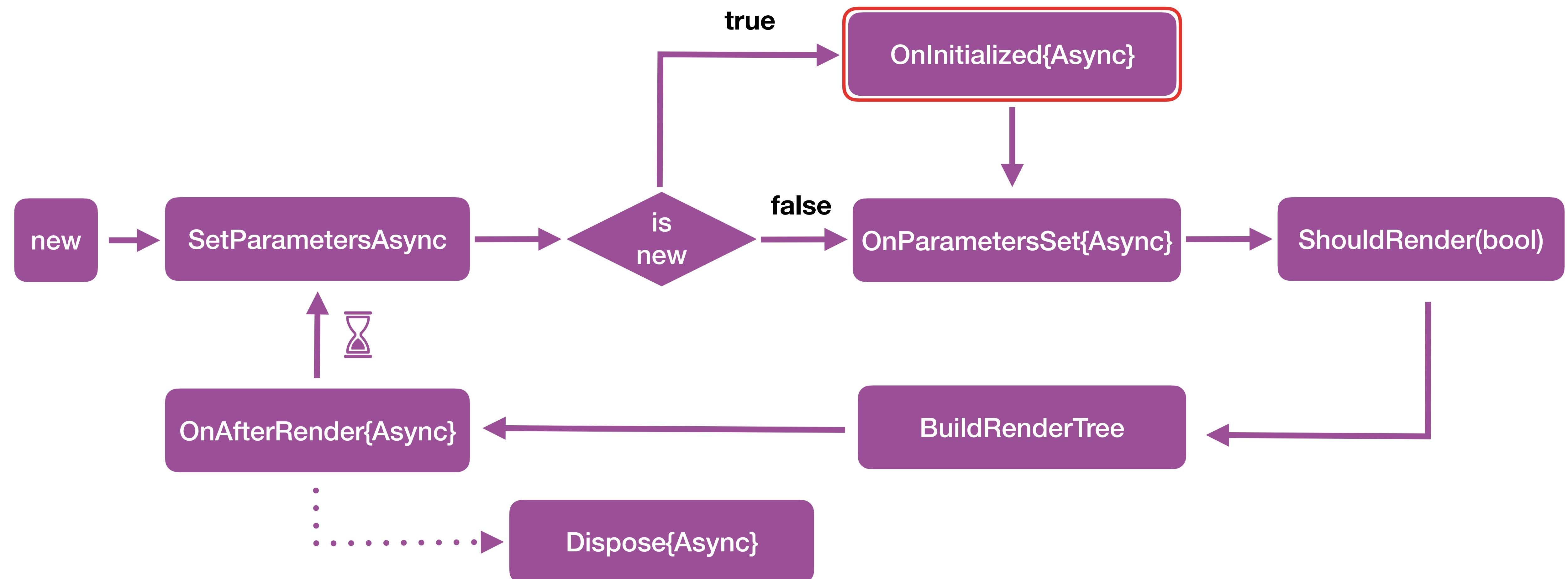
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



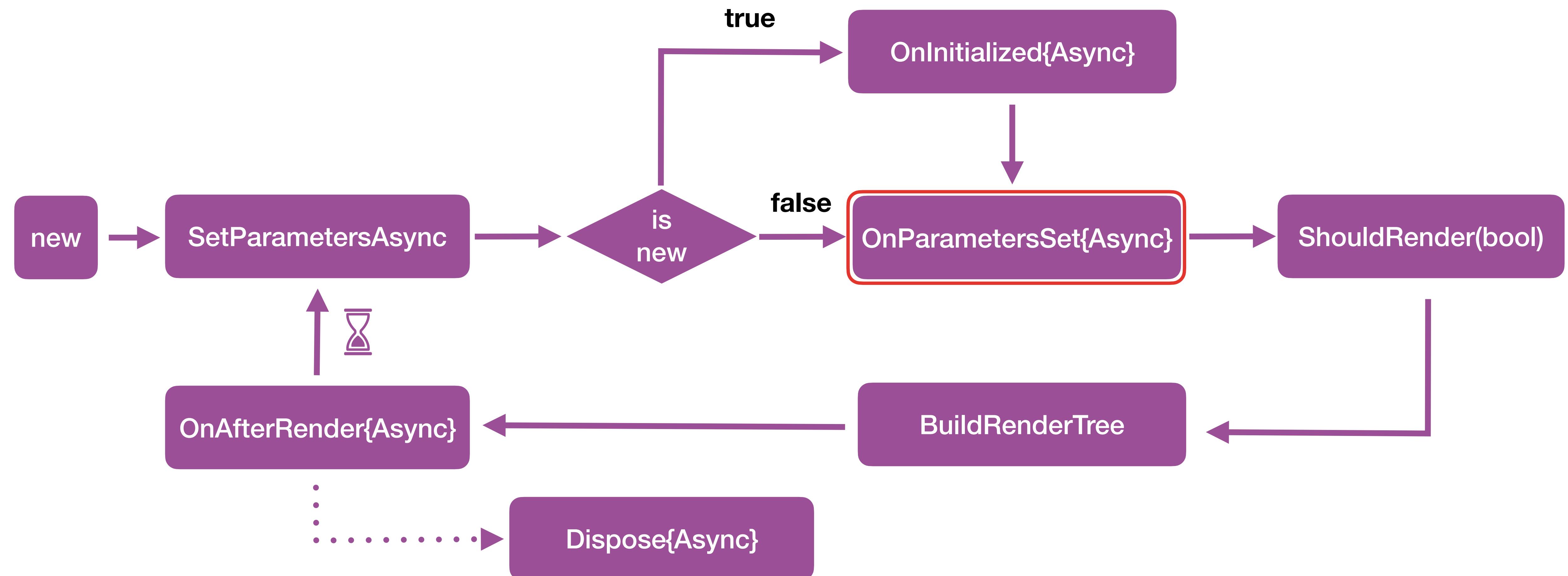
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



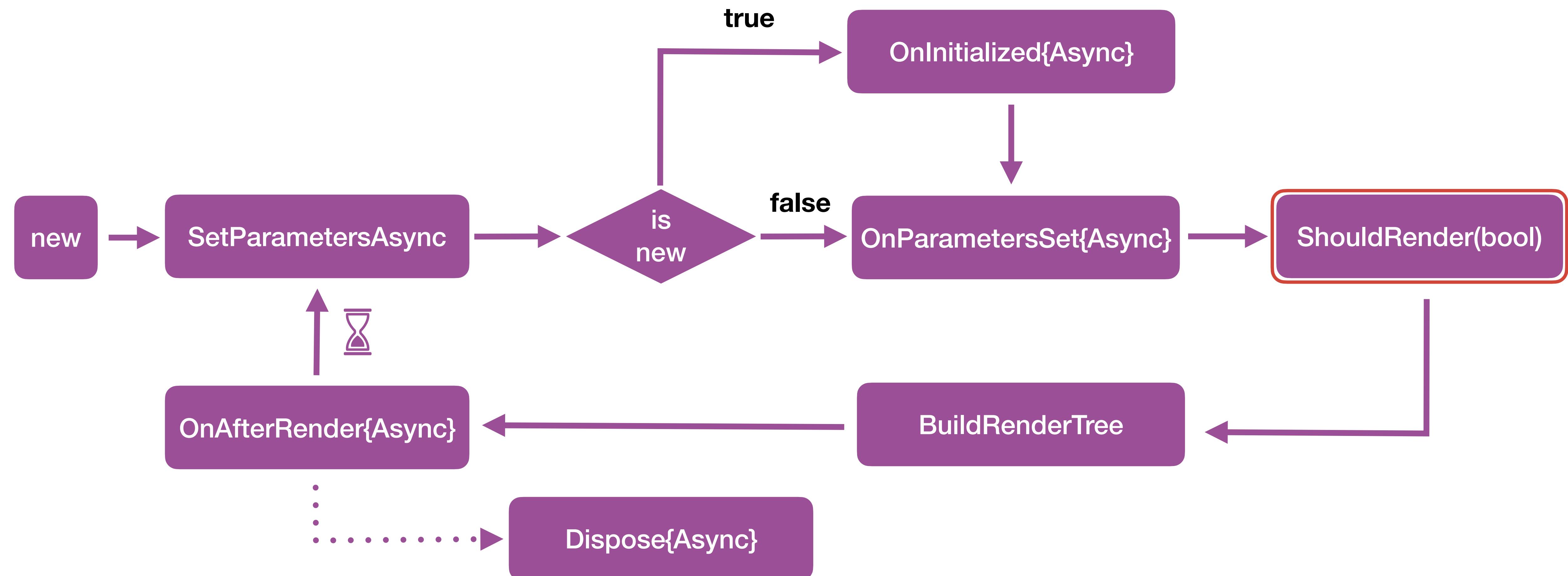
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



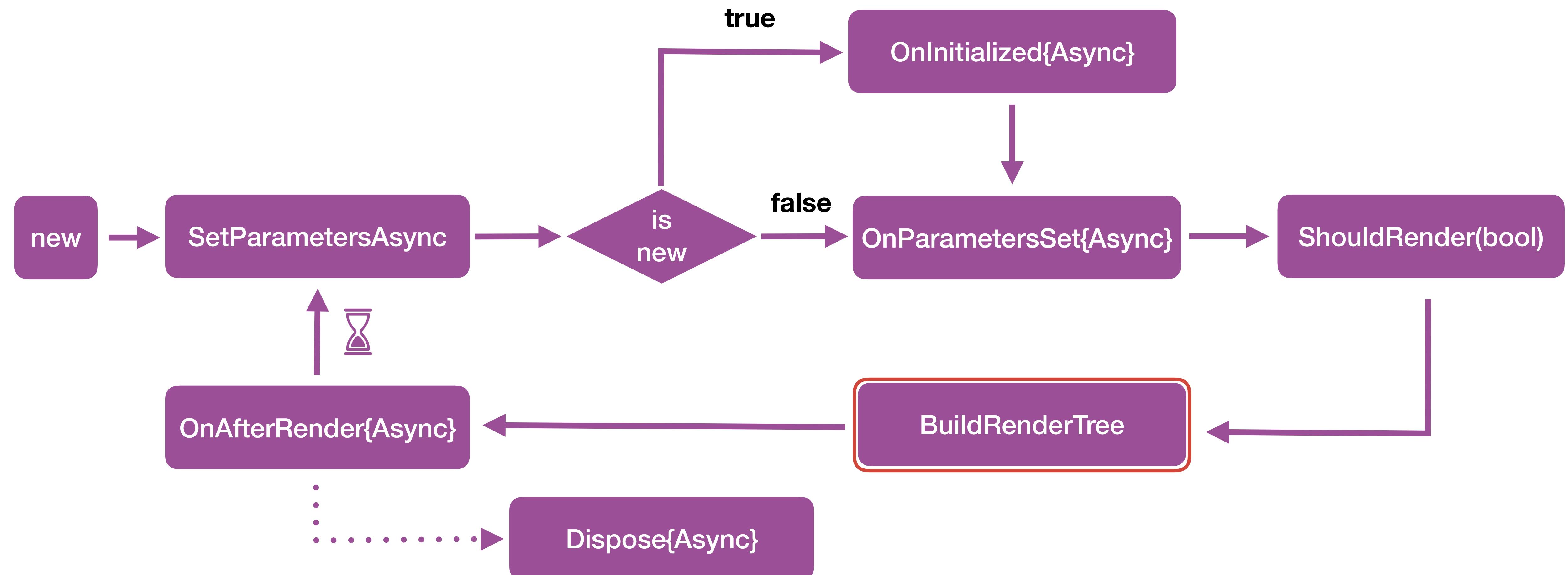
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



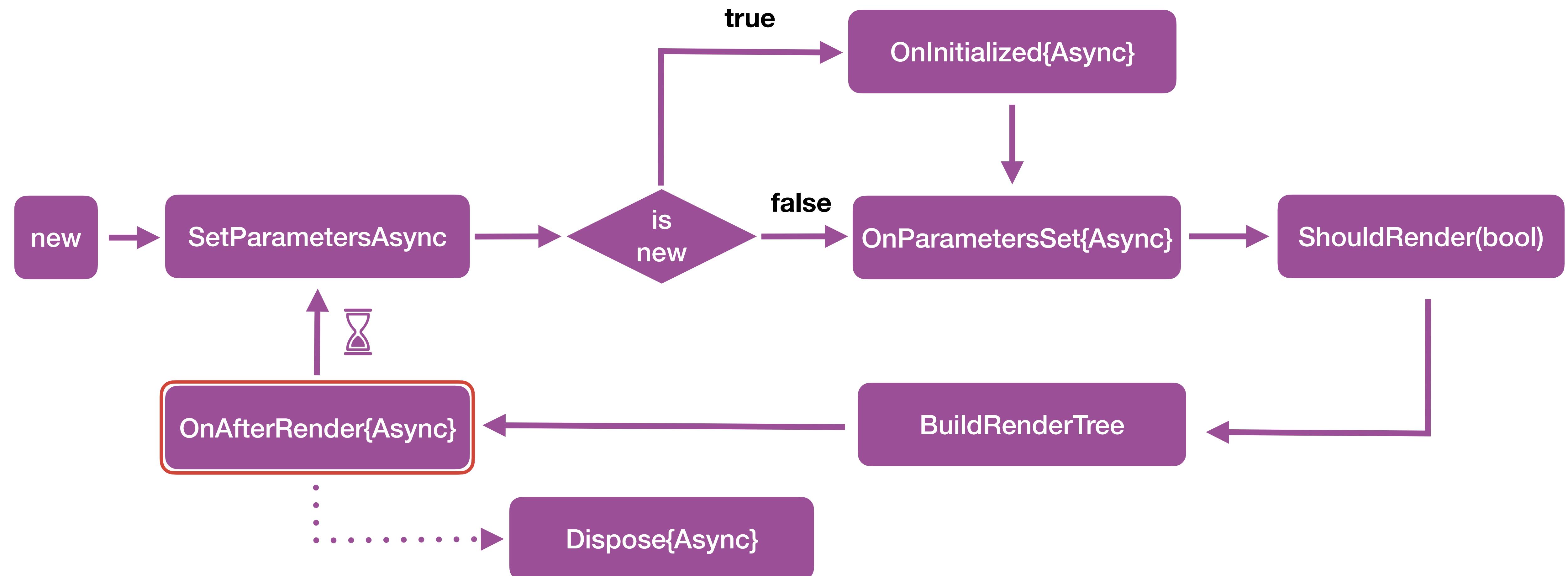
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



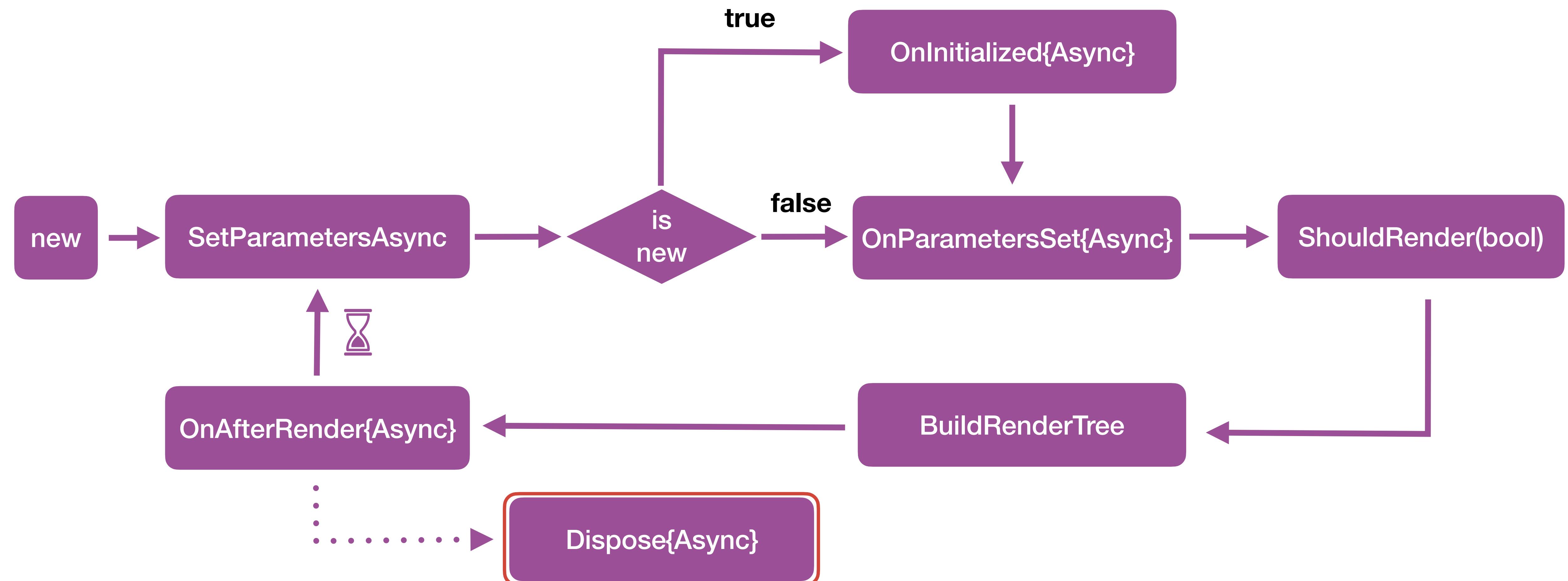
razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ



razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ

```
@_formattedString

@code {

    private string _formattedString = string.Empty;

    [Parameter]
    public int Count { get; set; }

    protected override void OnInitialized()
    {
        _formattedString = $"Count = {Count}";
    }
}
```

razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ

```
@_formattedString  
  
@code {  
  
    private string _formattedString = string.Empty;  
  
    [Parameter]  
    public int Count { get; set; }  
  
    protected override void OnInitialized()  
    {  
        _formattedString = $"Count = {Count}";  
    }  
}
```

Invalid lifecycle

This component uses `OnInitialized` method to update string with counter value.
As this method is invoked only one time after component object creation,
value would not be updated on increment

 Count = 0

razor

ЖИЗНЕННЫЙ ЦИКЛ КОМПОНЕНТОВ

```
@_formattedString  
  
@code {  
  
    private string _formattedString = string.Empty;  
  
    [Parameter]  
    public int Count { get; set; }  
  
    protected override void OnParametersSet()  
    {  
        _formattedString = $"Count = {Count}";  
    }  
  
}
```

Correct lifecycle

This component uses `OnParametersSet` method to update string with counter value.
As this method is invoked every time when parameters being set,
value is updated on increment

Count = 0

razor

RenderFragment

```
@if (IsAuthorized)
{
    @AuthorizedContent
}
else
{
    @NotAuthorizedContent
}

@code {
    [Parameter]
    public bool IsAuthorized { get; set; }

    [Parameter]
    public RenderFragment? AuthorizedContent { get; set; }

    [Parameter]
    public RenderFragment? NotAuthorizedContent { get; set; }
}
```

```
<AuthorizationDisplayComponent
    IsAuthorized="IsAuthorized">

    <AuthorizedContent>
        You are authorized!
    </AuthorizedContent>

    <NotAuthorizedContent>
        You are not authorized :(
    </NotAuthorizedContent>

</AuthorizationDisplayComponent>
```

toggle authorized

You are not authorized :(

blazor

blazor

hosting модели

- Blazor Server
SSR, приложение - тонкий клиент, общение с сервером через SignalR
- Blazor WebAssembly
SPA, приложение исполняется в WebAssembly

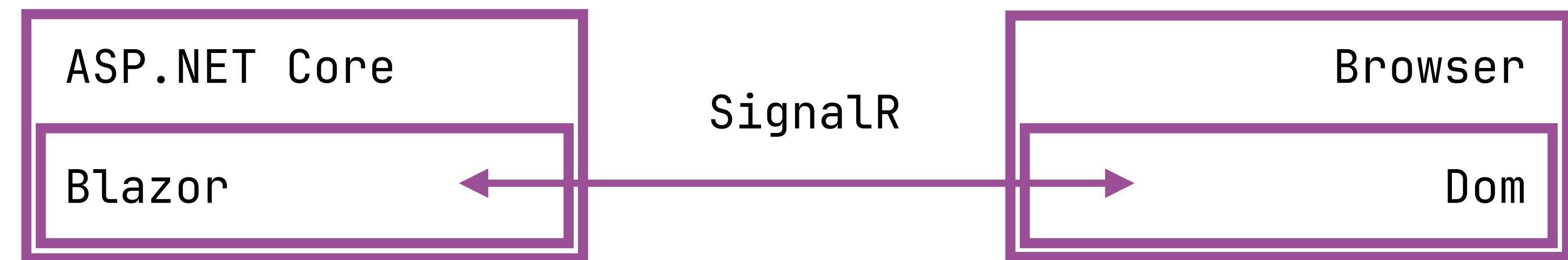
Blazor Hybrid

- интеграция Razor компонентов в desktop UI фреймворки
MaUI, WPF, Windows Forms

blazor server

blazor server

схема



blazor server

процесс инициализации

- пользователь получает страницу index.html
- она содержит разметку получаемой страницы
- в ней выполняется скрипт, подключающий обработчики Blazor и устанавливающий SignalR соединение с сервером

blazor server

circuits

- хранит контекст пользователя
- к circuit'ам привязан DI scope, создаваемый на каждого пользователя

blazor server

circuits

```
public class UserState
{
    public int Count { get; set; }
}

builder.Services.AddScoped<UserState>();

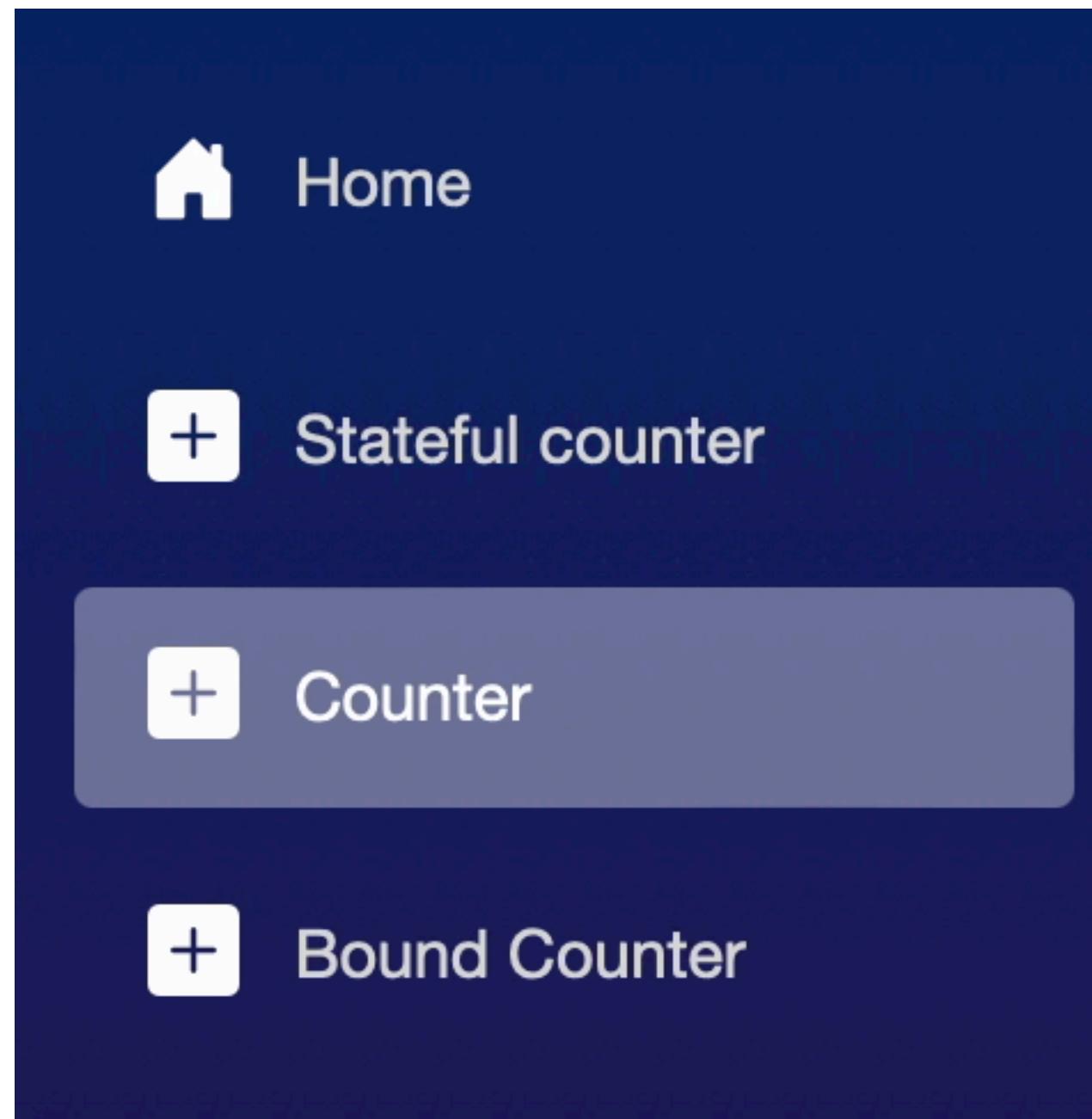
@using SpbDotNet.Blazor.Components.Services
@inject UserState State

<button @onclick="@(() => State.Count++)">
    increment
</button>

<div>
    Count = @State.Count
</div>
```

blazor server

circuits



Counter

This counter consists of two components.

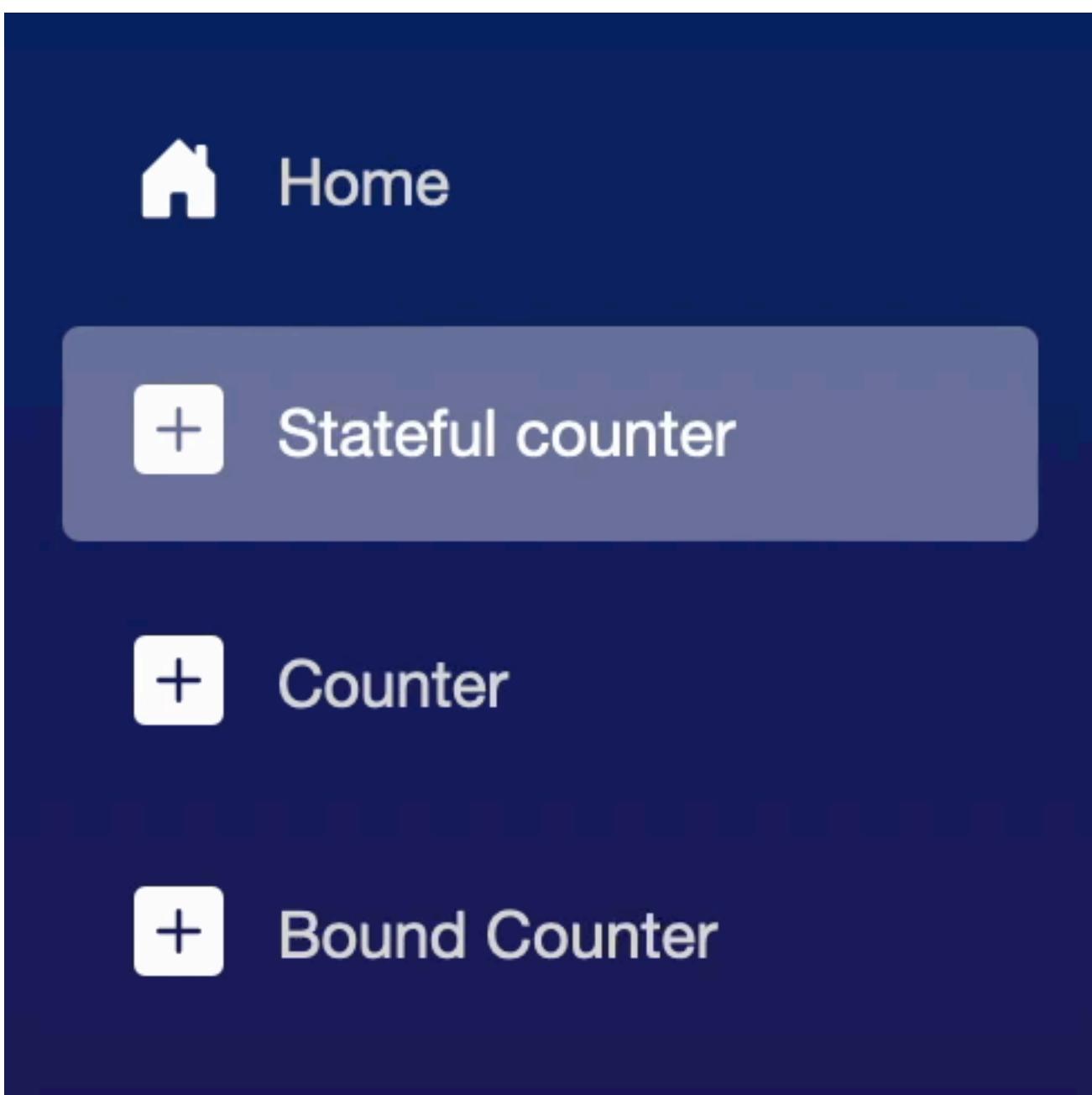
One is parent, other is inner.

Each of them can increment the value, but increm

parent count = 0 increment parent
inner count = 0 increment inner

blazor server

circuits



StatefulCounterPage

increment

Count = 0

blazor server

vs Razor Pages (ASP.NET MVC)

- Blazor Server – отдаёт интерактивные страницы
Razor Pages – отдаёт статику
- Blazor Server – инкрементально обновляет разметку
Razor Pages – обновляет разметку полностью

blazor server

virtual DOM

BrokenCounterComponent.razor

```
@using Microsoft.JSInterop  
@inject IJSRuntime Js  
  
increment  


Count = @_count

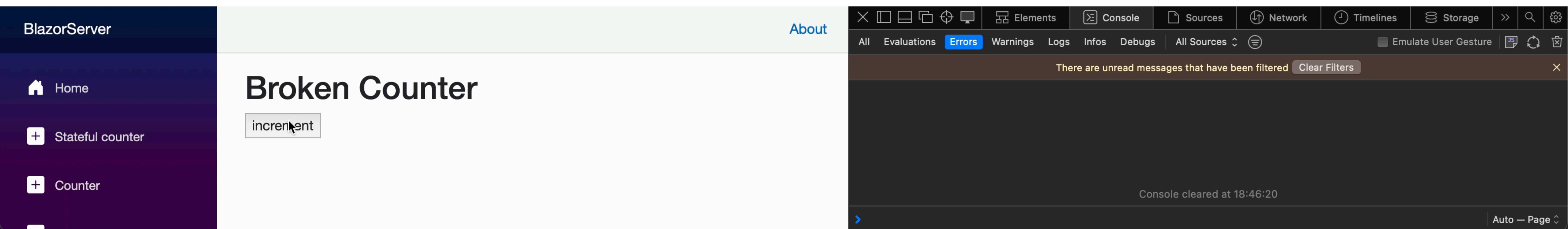
  
{  
  
    private int _count;  
  
    protected override async Task OnAfterRenderAsync(  
        bool firstRender)  
    {  
        await Js.InvokeVoidAsync(  
            "remove_element",  
            "broken-counter");  
    }  
}
```

App.razor

```
...  
  
  
...
```

blazor server

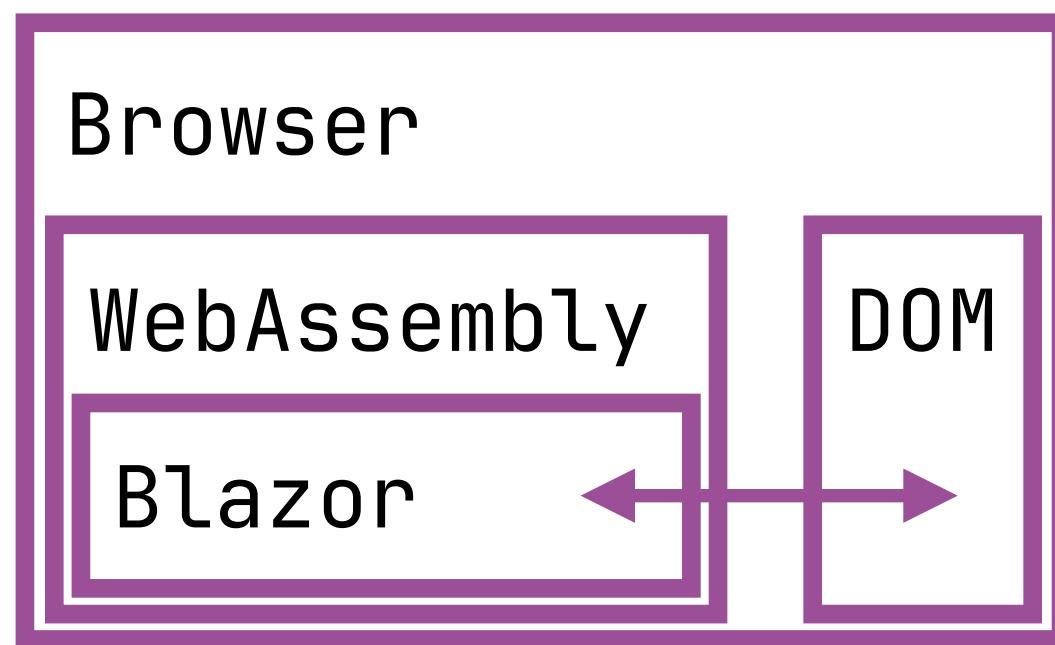
virtual DOM



blazor webassembly

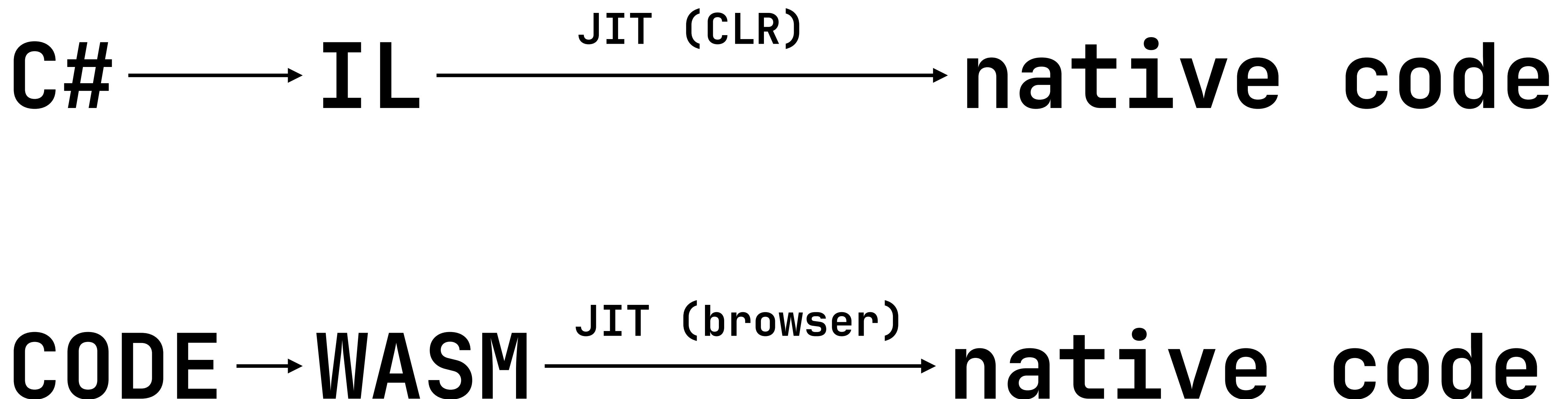
blazor webassembly

схема



webassembly

принцип работы



webassembly

загрузка

```
// importObject - list of functions imported into wasm sandbox
WebAssembly.instantiateStreaming(fetch("myModule.wasm"), importObject).then(
  (obj) => {
    // Bind exported func:
    window.exported_func = obj.instance.exports.exported_func;
  },
);
```

blazor webassembly

запуск в браузере

- браузер получает `index.html` и начинает выполнение скрипта `blazor.webassembly.js`
- скачивает `blazor.boot.json` (манифест приложения)
- скачивает и инициализирует Mono рантайм собранный под WASM
- скачивает все сборки (.dll файлы) приложения
- запускает JS `initialiser`'ы
методы `beforeStart(options, extensions)`, `afterStarted(blazor)`, определённые в `{AssemblyName}.lib.module.js`
- инициализирует объект `window.Blazor`
- запускает точку входа в Blazor хост

blazor webassembly

выполнение

- приложения Blazor WebAssembly выполняются в одном потоке
`Console.WriteLine(Environment.ProcessorCount); // 1`
- выполнение C# кода в WebAssembly инициируется JS'ным event loop'ом

blazor webassembly

AOT

```
<PropertyGroup>
    <RunAOTCompilation>true</RunAOTCompilation>
</PropertyGroup>
```

библиотеки компонентов

библиотеки компонентов

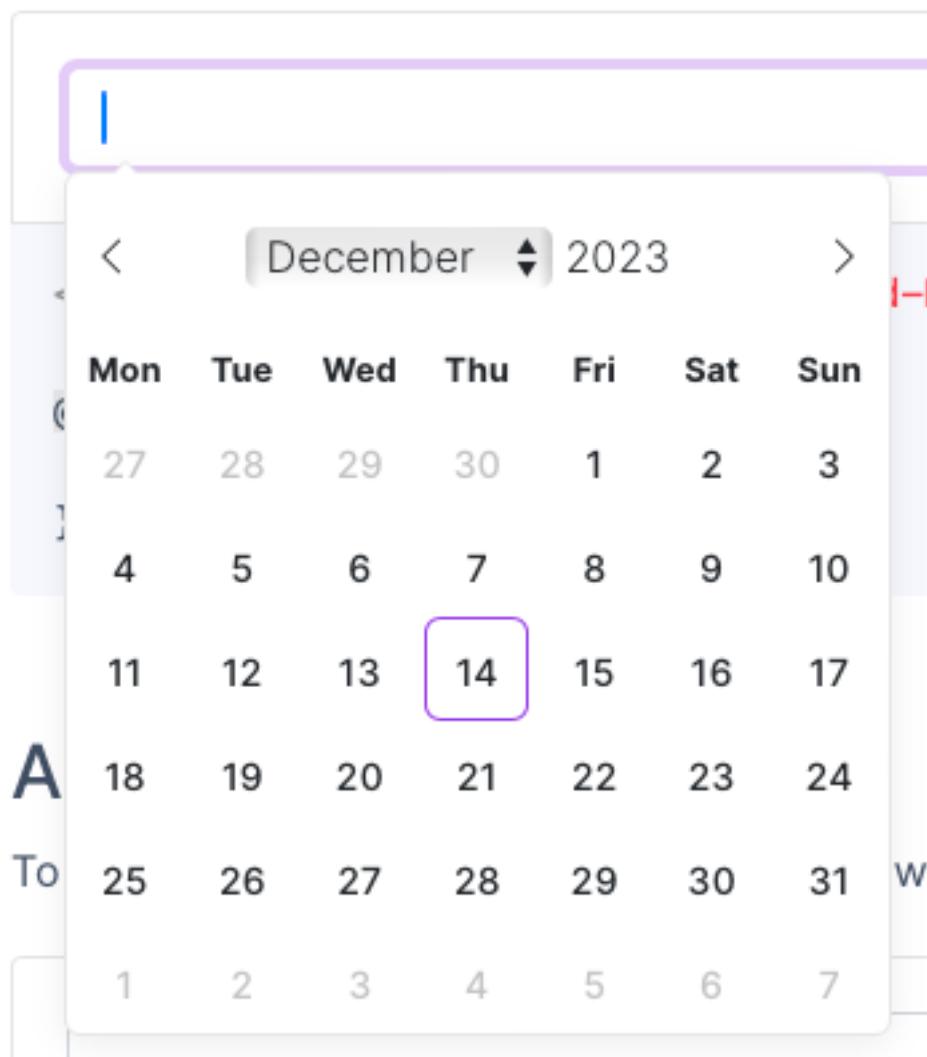
blazorise

- 80+ компонентов
- обёртка стилизаций
в большинстве случаев можно стилизовать компоненты без CSS, пользуясь абстракциями библиотеки
- темы
- различные провайдеры стилей
Bootstrap 4/5, Tailwind, Material, Ant Design, Bulma
- местами странный API-дизайн компонентов

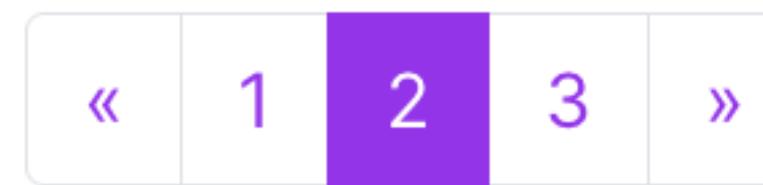
библиотеки компонентов

blazorise

DatePicker



Pagination



Table

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

библиотеки компонентов

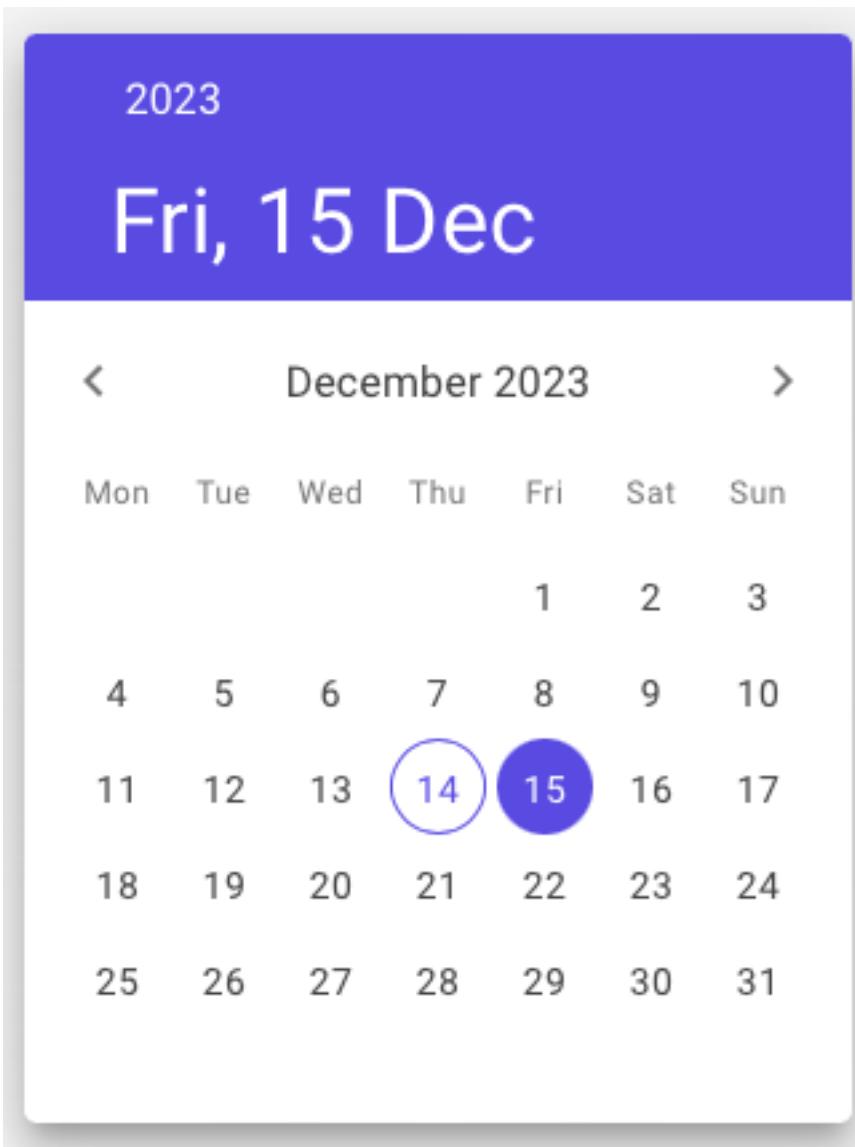
mudblazor

- большой выбор компонентов
- online плейграунд (try.mudblazor.com)
- темы
- обилие css классов для стилизаций
- более приятный внешний вид и общий API-дизайн в сравнении с Blazorise

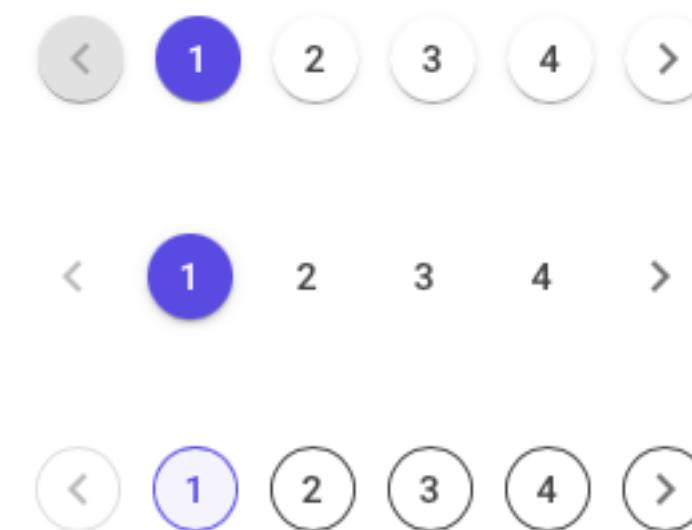
библиотеки компонентов

mudblazor

DatePicker



Pagination



Table

Nr	Sign	Name	Position	Molar mass
1	H	Hydrogen	0	1.00794
2	He	Helium	17	4.002602
3	Li	Lithium	0	6.941
4	Be	Beryllium	1	9.012182

best practices

best practices

структура

- компоненты не должны содержать логики
- стоит отделять слой приложения от компонентов
- декомпозирийте сложные компоненты особенно если они отрисовывают коллекции
- избегайте `backing-class`'ов выносите логику во `ViewModel`'и, не привязанные к компонентам

best practices

реактивность

- позволяет отделить логику приложения от обновления разметки
- nuget.org/packages/Phazor.Components

```
Username: <PhazorReactiveText Value="Username" />

<PhazorReactiveForEach Elements="Friends" Context="friendName">
    <li>@friendName</li>
</PhazorReactiveForEach>

@code {
    [Parameter]
    public IObservable<string>? Username { get; set; }

    [Parameter]
    public IObservable<IEnumerable<string>>? Friends { get; set; }
}
```

ССЫЛКИ

- github.com/ronimizy/SpbDotNet.Blazor

