



можно ли в Приложении Blazor вывести на страницу image в формате SVG ?

Да, в приложении Blazor можно вывести изображение в формате SVG на страницу самыми разными способами.^[1] ^[2]

Способы вывода SVG

- Можно использовать стандартный HTML-тег ``, если SVG-файл лежит в папке wwwroot вашего проекта.^[2]
 - SVG напрямую можно встроить в разметку страницы через тег `<svg>...</svg>` – это удобно
- В случае, если SVG хранится как строка или получен из API, его можно вывести посредством Razor-синтаксиса или через метод `@MarkupString`, если нужен динамический SVG.^[3]
- Для сложных сценариев (например, интерактивное редактирование SVG) рекомендуют использовать компоненты Blazor или JavaScript взаимодействия с SVG внутри DOM, либо специальные библиотеки.^[4]

Особенности использования

- SVG выводится корректно в Blazor на всех платформах, и поддерживает привязку данных, события, а также стилизацию через CSS.^[4]
- При необходимости динамически менять содержимое SVG возможно обновлять компонент или непосредственно управлять DOM через JSInterop или Blazor-события.^[3]

Вывод: SVG-формат поддерживается в Blazor и может быть встроен или загружен на страницу разными способами, в зависимости от задачи, в том числе стандартным способом через ``, а также как часть HTML-разметки или динамического контента.^[2]
^[4] ^[3]



Как динамически изменить содержимое SVG в Blazor приложении

В Blazor приложении SVG можно динамически изменять через привязку Razor-переменных и событий к свойствам SVG-элементов прямо в разметке, а также с использованием компонентов и кастомной логики.[\[17\]](#) [\[18\]](#) [\[19\]](#)

Основные подходы к динамике SVG

- SVG-фрагмент вставляется в Razor-страницу не через ``, а непосредственно `<svg>...

- Любые изменения соответствующих данных транслируются на SVG сразу при каждом обновлении состояния компонента.[\[17\]](#)
- Пример (Circle с динамическим цветом и текстом):

```
<svg width="200" height="100">
    <circle cx="50" cy="50" r="40" fill="@CircleColor"></circle>
    <text x="50" y="55" text-anchor="middle">@CircleText</text>
</svg>

<button @onclick="ChangeCircle">Изменить</button>

@code {
    string CircleColor = "red";
    string CircleText = "Пример";

    void ChangeCircle() {
        CircleColor = "blue";
        CircleText = "Изменено";
    }
}
```

Динамические SVG-компоненты

- Логика взаимодействия (например, drag-and-drop, динамические соединительные линии) часто оформляется отдельными Razor-компонентами, где SVG постоянно перерисовывается при изменении параметров.[\[20\]](#) [\[17\]](#)
- Внутри таких компонент можно реализовать любые сценарии: изменение положения, размера, цвета и текста SVG-элементов, а также взаимодействие с пользователем (клик, дрэг).[\[17\]](#)

Особенности

- Для сложных SVG (например, из файла) SVG-код копируют внутрь Razor-компонента и переменные/фрагменты делаю изменяемыми через C#.[\[18\]](#)
- Если требуется обновлять SVG напрямую в DOM (например, для анимации), допускается использовать JSInterop, но чаще полностью достаточно Razor-подхода.

[17]

Таким образом, обновление SVG в Blazor строится на прямой привязке данных и событий — любые свойства внутри <svg> можно связывать с переменными или параметрами и динамически актуализировать с помощью C#-кода компонента. [19] [18] [17]

**

Давайте посмотрим как рисуется график в проекте AspNetCore RazorPage с использованием библиотеки ChartDirector.

И Вы сделаете экспертную оценку - Возможно ли такое делать в Blazor.

Там все очень прост делается через вью.

В начале из примитивов создается SVG. И затем это все рендингится в View.

Смотрим простой код Razor Page

```
// Simplebar.cshtml
@page
@model NetCoreRazorCharts.Pages.SimplebarModel
 @{
 Layout = "_Layout";
 }

 @{
 if (ViewBag.Viewer is Array)
 {
 // Display multiple charts
 for (int i = 0; i < ViewBag.Viewer.Length; ++i)
 {
 @:@Html.Raw(ViewBag.Viewer[i].RenderHTML())
 }
 }
 else
 {
 // Display one chart only
 @:@Html.Raw(ViewBag.Viewer.RenderHTML())
 }
 }

// Simplebar.cshtml.cs
namespace NetCoreRazorCharts.Pages
{
 public class SimplebarModel : PageModel
 {
 public void OnGet()
 {
```

```

{
ViewData["Title"] = "Simple Bar Chart (1)";
RazorChartViewer viewer = new RazorChartViewer(HttpContext, "chart1");
ViewData["Viewer"] = viewer;
createChart(viewer);
}

// Create chart
//
private void createChart(RazorChartViewer viewer)
{
    // The data for the bar chart
    double[] data = { 85, 156, 179.5, 211, 123 };

    // The labels for the bar chart
    string[] labels = { "Mon", "Tue", "Wed", "Thu", "Fri" };

    // Create a XYChart object of size 250 x 250 pixels
    XYChart c = new XYChart(250, 250);

    // Set the plotarea at (30, 20) and of size 200 x 200 pixels
    c.setPlotArea(30, 20, 200, 200);

    // Add a bar chart layer using the given data
    c.addBarLayer(data);

    // Set the labels on the x axis.
    c.xAxis().setLabels(labels);

    // Output the chart
    viewer.Image = c.makeWebImage(Chart.SVG);

    // Include tool tip for the chart
    viewer.ImageMap = c.getHTMLImageMap("", "", "title='{xLabel}: {value} GBytes'");
}
}

```

Возможно ли такое сделать в Blazor ?

Может быть не такое, а другое по реализации
????

Возможность строить SVG-графики в Blazor есть, хотя ChartDirector не имеет официального Blazor-обёртки. Весь базовый сценарий — создание динамического SVG на C#, последующий вывод в View и интерактивность — реализуем в Blazor через Razor-компоненты.^[37]

Экспертная оценка: реализация графиков в Blazor

- **Создание SVG через C#**

В Blazor компонентах допускается построение SVG именно так: можно формировать SVG-текст программно (или описывать дерево SVG-элементов через Razor), а после выводить их с биндингом переменных и параметров через Razor-синтаксис. [\[38\]](#) [\[39\]](#) [\[37\]](#)

- **Рендеринг SVG в компоненте**

В компоненте Razor (.razor) пишется SVG-фрагмент, куда динамические данные (точки, цвета, подписи) подставляются через свойства и параметры компонента. Это примерно тот же принцип, что и в Razor Pages — только ViewBag заменяется на параметры и состояния. [\[39\]](#)

- **Интерактивность**

Можно реализовать взаимодействие: переключение данных, изменение вида графика и навешивание событий прямо в SVG через стандартные подходы Blazor и C#. [\[38\]](#) [\[39\]](#)

- **Поддержка сторонних библиотек**

Для удобной работы с графиками и визуализацией данных в Blazor используют:

- ApexCharts (обёртка над JS)
- Syncfusion Blazor Charts
- Plotly.Blazor
- MudBlazor Charts
- Можно использовать Chart.js через JSInterop, если нужен сторонний движок. [\[40\]](#) [\[37\]](#)

- Все эти библиотеки строят графики через SVG или Canvas и полностью поддерживают динамическое обновление (в том числе через C# модель данных).

Краткое сравнение подходов

Возможности	Razor Page (ChartDirector)	Blazor
Формировать SVG с C#	Да	Да [38] [39]
Вывод через View	Да	Да (через Razor Component) [38]
Интерактивность	Ограничена	Да (события, биндинг, JSInterop) [39] [40]
Сложные диаграммы	Да (через ChartDirector API)	Да (через сторонние Blazor Chart-библиотеки) [40] [37]

Вывод

Создать SVG-графики в Blazor аналогично тому, как это делается через ChartDirector в Razor Page, возможно полностью — как вручную, так и с помощью библиотек. Все плюсы Blazor — реактивность, богатая интерактивность и компонентный подход — делают задачу даже проще для динамических графиков. Трудности возможны если нужна

глубокая интеграция именно ChartDirector, но работа с SVG, динамическими данными и кастомизацией полностью поддерживается в Blazor.[\[40\]](#) [\[37\]](#) [\[39\]](#) [\[38\]](#)

**

viewer.Image = c.makeWebImage(Chart.SVG);

Вот что бы в этой строке мы создаем SVG

что нам мешает передать этот Image в?
?

В ChartDirector строка `viewer.Image = c.makeWebImage(Chart.SVG);` действительно создает

Почему возникает проблема передачи SVG?

- Возвращаемое значение – это готовый SVG-документ: строка вида `""` ,
• В Razor Page для вывода используется `@Html.Raw(...)`, что позволяет "сырую" SVG-разметку вставлять внутрь HTML (View) без экранирования.[\[49\]](#)
- В Blazor такой "сырой" онлайн-вставки нет по умолчанию, потому что Blazor экранирует строки для безопасности, чтобы избежать XSS.

Как правильно передать и отрендерить SVG в Blazor

- Можно использовать компонент с `@MarkupString`:

```
@((MarkupString)SvgString)
```

где `SvgString` — строка, содержащая весь сгенерированный SVG.[\[49\]](#)

- Если SVG нормальный, без внешних скриптов и inline-стилей, такой способ безопасен и позволяет выводить ChartDirector SVG в Blazor компоненте аналогично Razor Page.

Ограничения и альтернативы

- Вставлять весь SVG-документ как строку можно только с помощью @MarkupString, тогда он появится в DOM.^[49]
- Нельзя вручную модифицировать уже вставленный SVG, если он цельным фрагментом — для динамики лучше создавать SVG через Razor синтаксис, а ChartDirector использовать как генератор строки/графики по событию.^[50]
- Если требуется реактивное изменение частей графика, стоит парсить содержимое SVG или использовать Blazor Chart библиотеки, где управление элементами SVG происходит через параметры компонента.

Практический вывод

- В Blazor можно использовать ChartDirector для генерации SVG и выводить результат с помощью @((MarkupString)SvgString), если это действительно чистая SVG строка, как возвращает ChartDirector.^[49]

- Главное ограничение: не прямое помещение значения во внутрь `<svg>...</svg>`, а только

Таким образом, с технической точки зрения, нет фундаментальных препятствий — просто требуется использовать @MarkupString для рендера полного SVG. Это немного отличается по механике от Razor Page, но задача решается аналогично.^[49]

**

В моем сценарии файл SVG будет создоваться каждую секунду ПОЛНОСТЬЮ.

И надо эту полную строку помещать в
Напиши код как это будет выглядеть в Blazor

В Blazor для вашего сценария, когда SVG-строка формируется каждую секунду заново и содержимое нужно отображать на странице, можно использовать биндинг свойства и компонент Razor с помощью таймера. Чтобы "сырую" SVG-строку помещать в DOM, используют тип MarkupString:

```
@page "/svg-live"
<h3>Динамический SVG</h3>

<div>
```

```

@((MarkupString)SvgString)
</div>

@code {
    private string SvgString;

    protected override void OnInitialized()
    {
        // Имитация обновления SVG каждую секунду
        var timer = new System.Timers.Timer(1000);
        timer.Elapsed += (sender, args) =>
        {
            // Например, формируем новую SVG строку
            SvgString = GenerateSvg(DateTime.Now.Second);
            InvokeAsync(StateHasChanged);
        };
        timer.AutoReset = true;
        timer.Start();
    }

    // Генератор SVG для примера
    private string GenerateSvg(int x)
    {
        return $@"<svg width='200' height='100'>
            <circle cx='{x * 3 % 200}' cy='50' r='30' fill='red'/>
            <text x='10' y='90' font-size='20'>sec: {x}</text>
        </svg>";
    }
}

```

- Свойство `SvgString` содержит всю сгенерированную SVG-строку (документ).
- Каждый раз, когда приходит событие таймера (раз в секунду), строка пересоздается.
- Для вывода в HTML—, используется `@((MarkupString)SvgString)`, и SVG появляется в DOM как полноценная графика.

Этот подход абсолютно универсален для случаев, когда нужно регулярно обновлять и полностью менять SVG-содержимое в Blazor на лету. [\[59\]](#)

*

Для того чтобы нам проверить наши гипотезы и наш код Какой проект Blazor лучше подойдет для наших экспериментов

- Blazor Web App
- Blazor Server App
- Blazor Webassembly
- ?

Для экспериментов с динамическим SVG и обновлением данных каждую секунду лучше всего подойдут **Blazor Server App** или **Blazor Web App (Server Mode)**. Они дают максимально простую и быструю отладку событий, мгновенное обновление UI, не требуют сборки и публикации клиентской части и отлично работают для быстрой проверки гипотез.^[60] ^[61]

Рекомендации по выбору проекта

- **Blazor Server App**
 - Быстрее всего запускается, не требует много настроек.
 - Позволяет динамически обновлять интерфейс, вызывая C# код на сервере.
 - Обновление страницы происходит мгновенно по SignalR, прекрасно подходит для периодического генерации SVG на сервере.
 - Требует постоянного соединения с сервером (WebSocket/SignalR).
- **Blazor Web App (Server Mode)**
 - Новый вариант (с .NET 8), сочетающий плюсы Server и WebAssembly.
 - Можно запускать страницы как через сервер, так и полностью клиентски, выбирая нужный режим.
 - Идеален для отработки разных сценариев и миграции на WebAssembly позже.^[61]
- **Blazor WebAssembly**
 - Весь код работает на стороне клиента (в браузере).
 - Потенциально сложнее для быстрой проверки, потому что нужен билд, возможны ограничения по времени исполнения, "тяжелые" операции или большой SVG лучше делать на сервере.
 - Хорошо подойдет для финального решения — но для "набив руки" и быстрой проверки удобнее Server.

Итог

- Для быстрых экспериментов с динамическим SVG и live-обновлением лучше всего подойдет **Blazor Server App** или **Blazor Web App (Server Mode)**: минимальные задержки, простота запуска и легкость отладки кода на C#. ^[60] ^[61]
- Для чистого клиентского рендера и офлайн-работы — Blazor WebAssembly, но это скорее второй шаг.

Выбор Server Mode в начале экспериментов — оптimalен для тестирования всех гипотез и примеров динамических SVG. Позже, если нужно, логику можно будет перенести в WebAssembly-проект.^[61] ^[60]

Blazor Server App - только .Net 7.0,

поэтому выбираем Blazor Web App .Net 9

Выбор **Blazor Web App на .NET 9** — отличный современный вариант для ваших экспериментов! В этом проекте можно работать как в Server, так и в WebAssembly режиме (или гибридно), что особенно удобно для тестирования динамического SVG и обновляемых графиков.^[62]

Преимущества Blazor Web App (.NET 9)

- **Гибридные режимы:** Можно запускать компоненты в Server-рендеринге (ближе к классическому Blazor Server) или сразу переключиться на WebAssembly, когда понадобится.^[62]
- **Максимум новых возможностей:** В .NET 9 поддерживаются все свежие фичи Blazor (разделение рендеринга, расширенные компоненты, улучшенное взаимодействие с JS и DOM).
- **Удобная отладка:** Можно быстро работать с C#, видеть live-обновления и ускорить весь цикл экспериментирования.

Практически

Для ваших целей — генерации, вставки и динамического обновления SVG каждую секунду — Blazor Web App даст такую же производительность интерактивного UI, как и Blazor Server, но с возможностью легко мигрировать логику на клиент.

Создавайте **Blazor Web App (.NET 9)** и реализуйте эксперименты с динамикой SVG именно там — это актуальный и универсальный выбор.^[62]

**

Самое главное Blazor Web App (.NET 9) - там есть поддержка .Net Aspire - но это уже позже добавим

Да, в **Blazor Web App на .NET 9** действительно появилась полноценная поддержка платформы **.NET Aspire** — это ключевое преимущество для масштабных, облачных и микросервисных сценариев, которые можно внедрить позже по мере роста ваших требований.^[63]

- **.NET Aspire** интегрируется с Web App, позволяя конфигурировать облачные сервисы, оркестрацию и DevOps, а также автоматизировать развертывание и мониторинг приложений.
- Такой подход отлично подойдет для будущего развития, когда базовые эксперименты с динамическими SVG, графиками и UI переведут ваш проект на новый уровень — распределённая система, масштабирование, интеграция с API и облаками.^[63]

Можно смело запускать эксперименты в Blazor Web App (.NET 9), а возможности .NET Aspire добавить когда будут нужны продвинутые фичи или инфраструктура.[\[63\]](#)

**

Я запустил Blazor - он работает.

Как узнать в каком режиме я работаю

- В Сервере
- В WebAssembly
- ?

В Blazor Web App (.NET 9) можно узнать, в каком режиме работает компонент — серверном или WebAssembly — программно через API рендера, либо по конфигурации (Startup/Program.cs и параметрам рендеринга).[\[64\]](#) [\[65\]](#)

Как узнать режим выполнения в коде

Внутри компонента Razor можно вызвать:

```
@page "/mode-check"

<p>
    Текущий режим рендеринга: @RendererInfo.Name
</p>
<p>
    Интерактивность: @(RendererInfo.IsInteractive ? "Да" : "Нет")
</p>

@code {
    // Свойства доступны в ComponentBase начиная с .NET 8/9
    public RendererInfo RendererInfo => base.RendererInfo;

    // Варианты значений: "Server", "WebAssembly", "Static", "WebView"
}
```

- `RendererInfo.Name`:
Возвращает строку "Server" — серверный интерактивный режим,
"WebAssembly" — клиентский режим WASM,
"Static" — SSR неинтерактивно.
- `RendererInfo.IsInteractive`:
`true`, если компонент интерактивен (может реагировать на события).

Также назначенный режим можно получить через `ComponentBase.AssignedRenderMode`.

Как определить режим вручную

- Проверьте Startup/Program.cs:

Вызовы AddInteractiveWebAssemblyRenderMode() или AddInteractiveServerRenderMode()
явно задают нужный режим.^[64]

- В _Host.cshtml (или App.razor):

Если есть строчка <component type="..." render-mode="WebAssemblyPrerendered" /> — это WebAssembly.

Если "ServerPrerendered" — серверный режим.^[66]

Когда нужна ручная проверка

- Если проект Blazor Web App, то компоненты и страницы могут работать в разных режимах, и проверка через RendererInfo — самый точный способ узнать реальное место исполнения.^[66] ^[64]

Вывод:

Рекомендуется использовать свойство RendererInfo.Name внутри Razor-компонента, чтобы точно узнать в каком режиме работает ваш компонент (Server или WebAssembly).^[64]

**

public static void Main(string[] args)

```
{  
    var builder = WebApplication.CreateBuilder(args);  
  
    // Add services to the container.  
    builder.Services.AddRazorComponents()  
        .AddInteractiveServerComponents();  
  
    var app = builder.Build();  
  
    // Configure the HTTP request pipeline.  
    if (!app.Environment.IsDevelopment())  
    {  
        app.UseExceptionHandler("/Error");  
        // The default HSTS value is 30 days. You may want to change this for production scenarios  
        app.UseHsts();  
    }  
  
    app.UseHttpsRedirection();  
  
    app.UseAntiforgery();  
  
    app.MapStaticAssets();  
    app.MapRazorComponents<App>()  
        .AddInteractiveServerRenderMode();  
  
    app.Run();
```

}

Тут не написано Server или WebAssembly ?

В приведённом коде действительно явно не указано, что используется WebAssembly-режим. Ваша строка:

```
builder.Services.AddRazorComponents()  
    .AddInteractiveServerComponents();
```

и дальше:

```
app.MapRazorComponents<App>()  
    .AddInteractiveServerRenderMode();
```

означают, что приложение **работает в интерактивном серверном режиме** (Interactive Server), что абсолютно аналогично классическому Blazor Server.^[74] ^[75] ^[76]

Как определяется режим?

- AddInteractiveServerComponents() и AddInteractiveServerRenderMode() — это и есть включение интерактивного серверного рендера для всех компонентов по умолчанию.
- Если бы был нужен WebAssembly-режим, были бы вызваны методы типа AddInteractiveWebAssemblyComponents() и аналогичная настройка в MapRazorComponents.
- В .NET 9 режим работы можно также прописывать директивой @rendermode в самом Razor-компоненте.^[74]

Итог

Ваш код запускает приложение **только в интерактивном серверном режиме** (Blazor Server render mode).

WebAssembly в такой конфигурации не используется.

Если захочется включить WebAssembly, потребуется изменить конфигурацию и добавить соответствующий сервис и render mode.^[75] ^[76] ^[74]

**

Что Интересного я увиел в этом коде.

```
// Add services to the container.  
  
builder.Services.AddRazorComponents()  
    .AddInteractiveServerComponents();  
  
app.MapRazorComponents<App>()  
    .AddInteractiveServerRenderMode();
```

То есть мы можем разработать в Blazor с RazorPage как в Asp.Net CoreRazorPage

?

ТО есть мы можем работать со страницами RazorPage ?

Это было бы хорошо для графиков ChartDirector

Да, в современных версиях **Blazor Web App (.NET 9)** действительно можно совмещать Razor Pages и Blazor компоненты в одном проекте — они прекрасно интегрируются друг с другом. [\[84\]](#) [\[85\]](#)

Как это работает

- Вы можете использовать Razor Pages как раньше (например, для страниц, где удобно работать с классической моделью MVC/ViewData/ViewBag).
- Одновременно вы можете внедрять Blazor компоненты (интерактивные UI-элементы, например, динамические графики) прямо в Razor Page через специальные теги-компоненты или директивы. [\[84\]](#)
- Прямо в Razor Page допускается вызывать Blazor компонент:

```
<component type="typeof(MyBlazorComponent)" render-mode="ServerPrerendered" />
```

- И наоборот: Blazor компоненты могут работать в одном проекте с классическими Razor Pages и View, используя общие сервисы DI и ту же логику доступа к данным. [\[85\]](#)

Чем это полезно для ChartDirector и графиков

- Это позволяет строить гибридные приложения, где для некоторых страниц — например, статистики или отчетности — графики ChartDirector формируются на сервере и вставляются в Razor Page.
- Для других секций можно использовать интерактивные Blazor-компоненты с живой SVG-графикой, динамикой и обновлением данных через C#.

Вывод:

В новых Blazor Web App действительно возможно и удобно совмещать работу с Razor Pages и Blazor компонентами. Это дает гибкость для миграции старых решений или интеграции сторонних библиотек (например, ChartDirector), а также использования интерактивных Blazor-фичей в одном проекте. [\[85\]](#) [\[84\]](#)



Тогда напишите код, который создает в нашем Приложение Blazor страницу Razor Page с любым содержанием.

Давайте это сделаем.

В Blazor Web App (.NET 9) можно добавить Razor Page, как в классическом ASP.NET Core. Вот пошаговый пример, как создать страницу Razor Page с любым содержанием в таком

проекте.

1. Добавьте Razor Page

В папке Pages создайте файл, например:

Pages/ExamplePage.cshtml

```
@page
@model ExamplePageModel

@{
    ViewData["Title"] = "Пример Razor Page";
}

<h2>Привет, это Razor Page в Blazor Web App!</h2>
<p>Любое содержимое здесь допустимо.</p>
<p>Текущее время: @DateTime.Now</p>
```

2. Создайте модель страницы

В той же папке создайте файл

Pages/ExamplePage.cshtml.cs

```
using Microsoft.AspNetCore.Mvc.RazorPages;

public class ExamplePageModel : PageModel
{
    public void OnGet()
    {
        // Можно здесь задать любые данные для ViewData/ViewBag
        ViewData["Hello"] = "Данные для Razor Page";
    }
}
```

3. Проверьте регистрацию маршрута Razor Pages

В Blazor Web App (.NET 9) Razor Pages обычно доступны по умолчанию, но убедитесь, что в `Program.cs` есть:

```
builder.Services.AddRazorPages();
...
app.MapRazorPages();
```

4. Переход к странице

Откройте /ExamplePage в браузере — увидите вашу Razor Page с нужным содержимым.

Такой подход позволяет использовать стандартные Razor Pages параллельно с Blazor компонентами в одном проекте, открывая возможности для классических серверных страниц и интеграции сторонних инструментов. [\[92\]](#) [\[93\]](#)

**

Я по аналогии с RAzorPage написал такой код.

```
// Chart.cshtml.cs
using Microsoft.AspNetCore.Mvc.RazorPages;
using ChartDirector;

//namespace NetCoreRazorCharts.Pages

namespace BlazorWebApp01.Components.Pages
{
    public class ChartModel : PageModel
    {
        public void OnGet()
        {
            ViewData["Title"] = "Simple Bar Chart (1)";
            RazorChartViewer viewer = new RazorChartViewer(HttpContext, "chart1");
            ViewData["Viewer"] = viewer;
            createChart(viewer);
        }

        //
        // Create chart
        //
        private void createChart(RazorChartViewer viewer)
        {
            // The data for the bar chart
            double[] data = { 85, 156, 179.5, 211, 123 };

            // The labels for the bar chart
            string[] labels = { "Mon", "Tue", "Wed", "Thu", "Fri" };

            // Create a XYChart object of size 250 x 250 pixels
            XYChart c = new XYChart(250, 250);

            // Set the plotarea at (30, 20) and of size 200 x 200 pixels
            c.setPlotArea(30, 20, 200, 200);

            // Add a bar chart layer using the given data
            c.addBarLayer(data);

            // Set the labels on the x axis.
        }
    }
}
```

```

        c.xAxis().setLabels(labels);

        // Output the chart
        viewer.Image = c.makeWebImage(Chart.SVG);

        // Include tool tip for the chart
        viewer.ImageMap = c.getHTMLImageMap("", "", "title='{xLabel}: {value} GBytes'");
    }
}

// Chart.cshtml
@page
@model BlazorWebApp01.Components.Pages.ChartModel
@{
Layout = "_Layout";
}

@{
if (ViewBag.Viewer is Array)
{
    // Display multiple charts
    for (int i = 0; i < ViewBag.Viewer.Length; ++i)
    {
        @:@Html.Raw(ViewBag.Viewer[i].RenderHTML())
    }
}
else
{
    // Display one chart only
    @:@Html.Raw(ViewBag.Viewer.RenderHTML())
}
}

```

Я загрузил ChartDirector как зависимость.

Код компилируется.

Теперь надо где-то "ручку" прикрутить.

Как создать новую ссылку на мой код в Меню Blazor ?

Чтобы ваш ChartDirector-график был доступен через "ручку" в меню Blazor, нужно просто добавить новую ссылку (NavLink) на вашу Razor Page в общий компонент навигации (например, Sidebar или Header).

Пример: Добавление ссылки на Razor Page в Blazor меню

1. Найдите компонент меню

В шаблоне Blazor Web App это обычно файл `NavMenu.razor` (или общий `_Layout.cshtml` если используется классическая Razor-линия).

2. Добавьте NavLink на вашу страницу

Например, если ваша Razor Page называется `/Chart`:

```
<!-- NavMenu.razor или любой файл меню -->
<NavLink class="nav-link" href="/Chart" Match="NavLinkMatch.All">
    <span class="oi oi-bar-chart"></span> Диаграмма ChartDirector
</NavLink>
```

- `href="/Chart"` — маршрут вашей Razor Page (@page в `Chart.cshtml`).
- `NavLink` — компонент Blazor для навигации (выделяет выбранную страницу).

3. Если используется Razor Layout:

Добавьте обычную HTML ссылку в соответствующее место:

```
<a href="/Chart">Диаграмма ChartDirector</a>
```

Итог

Теперь в вашем меню появится новая кнопка, ведущая прямо на Razor Page с вашим графиком `ChartDirector`.

Это работает для любой Razor Page внутри Blazor Web App — страницы доступны через соответствующие маршруты и отлично интегрируются с навигацией Blazor.^[94] ^[95]

Рекомендация:

Используйте компонент `NavLink` для стиля и автоматического выделения активной ссылки, либо добавьте обычную HTML-ссылку для простоты.^[95]

**

В проектах RazorPage в каталоге Pages обычно находятся файлы `.cshtml` и `.cshtml.cs`.

Я по привычке так и сделал.

Каталог Pages

```
// Chart.cshtml
@page
@model BlazorWebApp01.Components.Pages.ChartModel
@{
    Layout = "_Layout";
}
```

```

@{
if (ViewBag.Viewer is Array)
{
// Display multiple charts
for (int i = 0; i < ViewBag.Viewer.Length; ++i)
{
@:@Html.Raw(ViewBag.Viewer[i].RenderHTML())
}
}
else
{
// Display one chart only
@:@Html.Raw(ViewBag.Viewer.RenderHTML())
}
}

```

Chart.cshtml.cs

```

using Microsoft.AspNetCore.Mvc.RazorPages;
using ChartDirector;

namespace BlazorWebApp01.Components.Pages
{
public class ChartModel : PageModel
{
public void OnGet()
{
ViewData["Title"] = "Simple Bar Chart (1)";
RazorChartViewer viewer = new RazorChartViewer(HttpContext, "chart1");
ViewData["Viewer"] = viewer;
createChart(viewer);
}

    //
    // Create chart
    //
    private void createChart(RazorChartViewer viewer)
    {
        // The data for the bar chart
        double[] data = { 85, 156, 179.5, 211, 123 };

        // The labels for the bar chart
        string[] labels = { "Mon", "Tue", "Wed", "Thu", "Fri" };

        // Create a XYChart object of size 250 x 250 pixels
        XYChart c = new XYChart(250, 250);
    }
}

```

```

    // Set the plotarea at (30, 20) and of size 200 x 200 pixels
    c.setPlotArea(30, 20, 200, 200);

    // Add a bar chart layer using the given data
    c.addBarLayer(data);

    // Set the labels on the x axis.
    c.xAxis().setLabels(labels);

    // Output the chart
    viewer.Image = c.makeWebImage(Chart.SVG);

    // Include tool tip for the chart
    viewer.ImageMap = c.getHTMLImageMap("", "", "title='{xLabel}: {value} GBytes'");
}

}

}

```

В нгавигации тоже ввел еще одну фрагмент

```
<span class="bi bi-list-nested-nav-menu" aria-hidden="true"></span> Chart
```

В меню появилась ссылка Chart.

Нажимаю на Chart

Высвечивается пустая страница с Error 404

Я посмотрел как выполнены в коде другие опции "Count", "Wheather"

Все не так.

В каталоге Pages хранятся файлы не .cshtml - как в Классическом RAzorPAge и как сделал я. В этом файле pages храняться файлы .razor.

Поэтому моя страница Chart не работает.

Как быть ?

Я увидел поддержку RAzorPage и сделал код как в странице RAzorPage классического RAzorPage. Но оказалось, что в blazor все по другому - в Pages хранятся файлы .razor . Вопрос куда мне воткнуть мой проавильный RAzorPage code ?

Вы столкнулись с классической особенностью Blazor Web App: по умолчанию в каталоге Pages действительно хранятся **.razor**-компоненты, а не классические Razor Page (.cshtml/.cshtml.cs). [113] [114]

В Blazor Web App шаблон не настраивает маршрутизацию для Razor Page (.cshtml), если вы не добавите её явно. Поэтому страница по адресу /Chart даёт 404, хотя файл и код корректны.

Решение: Как добавить Razor Page (.cshtml) в Blazor Web App

1. Зарегистрируйте поддержку Razor Pages в Program.cs:

Убедитесь, что есть строки:

```
builder.Services.AddRazorPages();
...
app.MapRazorPages();
```

2. Каталог хранения:

- Ваши .cshtml и .cshtml.cs должны находиться в папке Pages (или подпапке), но НЕ внутри Components — иначе они не будут распознаны Razor Pages маршрутизацией.
- Переместите Chart.cshtml и Chart.cshtml.cs из BlazorWebApp01.Components.Pages просто в Pages (корень папки Pages).

3. Маршрут ссылки:

- Если файл лежит по пути /Pages/Chart.cshtml, то URL будет /Chart (без расширения).
- В меню должно быть:

```
<NavLink class="nav-link" href="/Chart">
    <span class="bi bi-list-nested-nav-menu"></span> Chart
</NavLink>
```

4. Перезапустите приложение:

- После изменения структуры каталогов и включения Razor Pages маршрутизации перезапустите приложение.

Почему так?

- Страницы .razor — это Blazor Components, полностью работают на Blazor pipeline.
- Страницы .cshtml + .cshtml.cs — это Razor Page (MVC), работают только если включена маршрутизация Razor Pages и файлы лежат в стандартных директориях (Pages, а не в Components или других папках).

Пример структуры проекта

```
BlazorWebApp01/
Pages/
    Chart.cshtml
    Chart.cshtml.cs
    Index.razor
    Counter.razor
    Weather.razor
Components/
    (ваши .razor-фрагменты)
```

Итог

Чтобы классический Razor Page (`Chart.cshtml`) заработал в Blazor Web App:

- Переместите `.cshtml/.cshtml.cs` в папку `Pages` (корень).
- Проверьте регистрацию Razor Pages в `Program.cs`.
- Сделайте ссылку `NavLink` вида `href="/Chart"`.
- Не забудьте про перезапуск приложения.

Теперь ваш `ChartDirector`-график будет открываться по ссылке в меню так же, как и старые Razor Page — параллельно с компонентами Blazor.[\[114\]](#) [\[113\]](#)

**

Получилось, но не очень.

Что хорошо

1. График выводиться в соответствие с кодом `RAzorPage` через `view`
Это уже почти Победа - но не очень. То есть код с `ChartDirector` - легко выводит график.
2. Но весь `Layout` меняется - исчезает боковая Панель Меню - поэтому надо дублировать весь `Layout` из `Component.Layout`

Выводы

1. Вобщем синдром двух `Layout`. Надо подумать как это устраниить.
2. Рассмотреть возможность работать с `CHartDirector` через родной механизм `Bazor` - то есть через `CComponent.Pages.Chart` то есть через механизм `RAzor`.
Пример
`@page "/weather"`
`@attribute [StreamRendering]`

`Weather`

Weather

This component demonstrates showing data.

```
@if (forecasts == null)  
{  
    Loading...  
}
```

```
else  
{
```

```

<th aria-label="Temperature in Celsius">Temp. (C)</th>
<th aria-label="Temperature in Fahrenheit">Temp. (F)</th>

@foreach (var forecast in forecasts)
{
}



| Date                               | Summary                |                        |                   |
|------------------------------------|------------------------|------------------------|-------------------|
| @forecast.Date.ToShortDateString() | @forecast.TemperatureC | @forecast.TemperatureF | @forecast.Summary |



}

@code {
private WeatherForecast[]? forecasts;

protected override async Task OnInitializedAsync()
{
    // Simulate asynchronous loading to demonstrate streaming rendering
    await Task.Delay(500);

    var startDate = DateOnly.FromDateTime(DateTime.Now);
    var summaries = new[] { "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Bal" };
    forecasts = Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = startDate.AddDays(index),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = summaries[Random.Shared.Next(summaries.Length)]
    }).ToArray();
}

private class WeatherForecast
{
    public DateOnly Date { get; set; }
    public int TemperatureC { get; set; }
    public string? Summary { get; set; }
    public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
}
}

```

КАК РАЗ МНЕ НУЖНО ЧЕРЕЗ КАЖДУЮ СЕКУНДУ ГРАФИК ВЫВОДИТЬ. ТАК ЧТО ВЗЯТЬ ЭТОТ КОД ЗА ОСНОВУ.

Главный вывод такой,
что CChartDirector спокойно работает в Blazor через механизм RazorChart вот так
public class ChartModel : PageModel
{

```

public void OnGet()
{
    ViewData["Title"] = "Simple Bar Chart (1)";
    RazorChartViewer viewer = new RazorChartViewer(HttpContext, "chart1");
    ViewData["Viewer"] = viewer;
    createChart(viewer);
}

// Create chart
//
private void createChart(RazorChartViewer viewer)
{
    // The data for the bar chart
    double[] data = { 85, 156, 179.5, 211, 123 };

    // The labels for the bar chart
    string[] labels = { "Mon", "Tue", "Wed", "Thu", "Fri" };

    // Create a XYChart object of size 250 x 250 pixels
    XYChart c = new XYChart(250, 250);

    // Set the plotarea at (30, 20) and of size 200 x 200 pixels
    c.setPlotArea(30, 20, 200, 200);

    // Add a bar chart layer using the given data
    c.addBarLayer(data);

    // Set the labels on the x axis.
    c.xAxis().setLabels(labels);

    // Output the chart
    viewer.Image = c.makeWebImage(Chart.SVG);

    // Include tool tip for the chart
    viewer.ImageMap = c.getHTMLImageMap("", "", "title='{xLabel}: {value} GBytes'");
}
}

```

НО надо CHartDirector научить работать как здесь через .razor компонент.

```

@page "/weather"
@attribute [StreamRendering]

```

Weather

Weather

This component demonstrates showing data.

```
@if (forecasts == null)
{
    Loading...
}

}
else
{

<th aria-label="Temperature in Celsius">Temp. (C)</th>
<th aria-label="Temperature in Fahrenheit">Temp. (F)</th>

@foreach (var forecast in forecasts)
{
}



| Date                               | Summary                |                        |                   |
|------------------------------------|------------------------|------------------------|-------------------|
| @forecast.Date.ToShortDateString() | @forecast.TemperatureC | @forecast.TemperatureF | @forecast.Summary |



}

@code {
private WeatherForecast[]? forecasts;

protected override async Task OnInitializedAsync()
{
    // Simulate asynchronous loading to demonstrate streaming rendering
    await Task.Delay(500);

    var startDate = DateOnly.FromDateTime(DateTime.Now);
    var summaries = new[] { "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy" };
    forecasts = Enumerable.Range(1, 5).Select(index => new WeatherForecast
    {
        Date = startDate.AddDays(index),
        TemperatureC = Random.Shared.Next(-20, 55),
        Summary = summaries[Random.Shared.Next(summaries.Length)]
    }).ToArray();
}

private class WeatherForecast
{
    public DateOnly Date { get; set; }
    public int TemperatureC { get; set; }
    public string? Summary { get; set; }
}
```

```
        public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);  
    }  
  
}
```

Ваши наблюдения абсолютно точны: использование Razor Page (.cshtml) в Blazor Web App приводит к разрыву layout'ов — появляются две независимые зоны UI, и боковое меню Blazor исчезает при выводе классической Razor Page. Лучшее решение для современных задач — интегрировать ChartDirector в виде Blazor компонента (.razor), чтобы график органично вписывался в основной layout и использовал преимущества реактивности Blazor.[\[115\]](#) [\[116\]](#)

Как использовать ChartDirector в .razor компоненте

Вот шаблонный пример того, как можно "научить" ChartDirector работать через компонент Blazor — так, чтобы layout, меню и вся реактивность оставались на месте.

1. Создайте новый компонент (например, `ChartDemo.razor`):

```
@page "/chartdemo"  
@attribute [StreamRendering]  
  
<PageTitle>Chart Demo</PageTitle>  
  
<h2>ChartDirector в Blazor компоненте</h2>  
@if (svgString == null)  
{  
    <p><em>Loading chart...</em></p>  
}  
else  
{  
    @((MarkupString)svgString)  
}  
  
@code {  
    private string? svgString;  
  
    protected override async Task OnInitializedAsync()  
    {  
        await Task.Delay(500); // для имитации загрузки, можно убрать  
  
        svgString = GenerateChartSvg();  
        StateHasChanged();  
    }  
  
    private string GenerateChartSvg()  
    {  
        // инициализация ChartDirector-объектов через C#  
        double[] data = { 85, 156, 179.5, 211, 123 };  
        string[] labels = { "Mon", "Tue", "Wed", "Thu", "Fri" };  
        var c = new ChartDirector.XYChart(250, 250);  
        c.setPlotArea(30, 20, 200, 200);
```

```

        c.addBarLayer(data);
        c.xAxis().setLabels(labels);
        string svg = c.makeWebImage(ChartDirector.Chart.SVG);
        return svg;
    }
}

```

- Используется ChartDirector для создания SVG.
- SVG полностью выводится через MarkupString, прямо как в Razor Page — но теперь в компоненте Blazor.

2. Для live-обновления графика раз в секунду:

```

@page "/chartlive"
@attribute [StreamRendering]

<PageTitle>Live Chart</PageTitle>

@if (svgString == null)
{
    <p><em>Loading...</em></p>
}
else
{
    @((MarkupString)svgString)
}

@code {
    private string? svgString;
    private int tick = 0;

    protected override void OnInitialized()
    {
        var timer = new System.Timers.Timer(1000);
        timer.Elapsed += (s, e) =>
        {
            tick++;
            svgString = GenerateChartSvg(tick);
            InvokeAsync(StateHasChanged);
        };
        timer.Start();
    }

    private string GenerateChartSvg(int state)
    {
        double[] data = { 85 + state, 156, 179.5, 211, 123 };
        string[] labels = { "Mon", "Tue", "Wed", "Thu", "Fri" };
        var c = new ChartDirector.XYChart(250, 250);
        c.setPlotArea(30, 20, 200, 200);
        c.addBarLayer(data);
        c.xAxis().setLabels(labels);
        string svg = c.makeWebImage(ChartDirector.Chart.SVG);
        return svg;
    }
}

```

```
    }  
}
```

- График обновляется каждую секунду, layout Blazor сохраняется.

Важные выводы

- Интеграция ChartDirector в `.razor` компоненте позволяет полностью использовать Blazor-меню и layout — никаких конфликтов между layout'ами.^[115]
- Для интерактивности и динамики (например, обновление графика), используйте C# код внутри компонента и выводите SVG через `MarkupString`, как с Weather-фичей.
- Код ChartDirector, который создаёт график, переносится в C#-методы компонента.

Итог

Чтобы избавиться от "синдрома двух layout'ов", надо переносить логику генерации SVG ChartDirector из Razor Page (`.cshtml`) в `.razor` компонент Blazor. Это даёт свежее меню, поддержку layout'a, интеграцию с остальным Blazor UI и максимум реактивности, включая live-обновление графиков.^{[116] [115]}

**

Наши мнения сошлись воедино – это радует

Да, ваши выводы полностью совпадают с современными рекомендациями: ChartDirector легко интегрируется в Blazor через генерацию SVG прямо в `.razor` компоненте, что позволяет использовать весь функционал Blazor (layout, реактивность, меню, live-обновления) без конфликтов между Razor Page и Blazor-компонентами.^{[117] [118]}

- Такой подход избавляет от "двух layout'ов" и обеспечивает единый UI.
- Реактивность и частые обновления графика реализуются просто — достаточно передавать свежую SVG-строку из ChartDirector в `MarkupString` в коде компонента.
- Вызовы ChartDirector на стороне сервера совместимы с интерактивным Blazor Server рендером, а на WebAssembly — можно хранить логику на API/сервере или генерировать простые SVG вручную.

Итог:

ChartDirector действительно можно использовать для генерации SVG-графиков в Blazor, но лучший UX достигается через нативные Blazor компоненты и встроенный layout. Ваши эксперименты — отличная основа для гибридных решений, а дальнейшая интеграция ChartDirector или других chart-библиотек в Blazor проходит так же удобно через компоненты и `MarkupString`.^{[118] [117]}

**

1. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/images-and-documents?view=aspnetcore-9.0>
2. https://www.reddit.com/r/dotnet/comments/1gb4gj0/why_does_net_still_use_xaml_a_frustrated/
3. <https://www.cyberforum.ru/asp-net-core/thread2740415.html>
4. <https://habr.com/ru/articles/581192/>
5. <https://ru.stackoverflow.com/questions/1347140/Как-спарсить-страницу-с-blazor-сайта>
6. https://www.reddit.com/r/Blazor/comments/18myfqj/svg_icons/
7. https://быстрыеотчеты.рф/public_download/docs/FRNet/FRNetProgrammerManual-ru.pdf
8. <https://dotnet.microsoft.com/ru-ru/apps/aspnet/web-apps/blazor>
9. https://www.reddit.com/r/Blazor/comments/oihuf4/how_to_draw_over_an_image/
10. https://www.stimulsoft.com/documentation/ru/programming-manual/reports_web_blazor_using_web_viewer_settings.htm
11. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/?view=aspnetcore-9.0>
12. <https://stackoverflow.com/questions/63706078/how-to-show-a-svg-file-which-is-loaded-in-a-variable-on-a-html-page-blazor-ser>
13. <https://ru.stackoverflow.com/questions/881670/Отобразить-svg-в-image>
14. https://www.reddit.com/r/androiddev/comments/13vhj38/how_do_you_handle_images_storing_in_your_app/
15. <https://habr.com/ru/articles/648759/>
16. <https://www.stimulsoft.com/ru/products/ultimate>
17. <https://habr.com/ru/articles/582874/>
18. <https://stackoverflow.com/questions/72560925/blazor-webassembly-manipulate-items-in-svg-image-at-runtime>
19. <https://www.cyberforum.ru/asp-net-core/thread2740415.html>
20. <https://habr.com/ru/articles/581192/>
21. <https://expediteappsnet.wordpress.com/2020/01/20/blazor-and-svg/>
22. https://www.reddit.com/r/Blazor/comments/zvxuqy/issues_with_refreshing_items_in_blazor_virtualize/
23. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/dynamiccomponent?view=aspnetcore-9.0>
24. <https://temofeev.ru/info/articles/javascript-redaktor-svg-diagramm-kotoryyy-veisit-v-6-5-raz-menshe-bootstraps-biblioteka-s-otkrytym-iskh/>
25. <https://habr.com/ru/articles/577174/>
26. <https://dotnet.microsoft.com/ru-ru/apps/aspnet/web-apps/blazor>
27. https://ru.stackoverflow.com/questions/563297/svg_Динамическая-погрузка-создание-добавление-на-страницу
28. <https://ru.stackoverflow.com/questions/1131829/Как-манипулировать-изображением-внутри-элемента-svg>
29. <https://www.stimulsoft.com/ru/changes/reports-blazor/2023.2>
30. <https://www.interestprograms.ru/source-codes-blazor-components>
31. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/fundamentals/startup?view=aspnetcore-9.0>
32. https://www.reddit.com/r/Blazor/comments/oyhen0/i_have_worked_more_on_my_blazor_svg_editor_it/

33. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/?view=aspnetcore-9.0>
34. https://www.reddit.com/r/Blazor/comments/1bnr6nw/having_issue_with_svg_element_not_rendering_is/
35. <https://habr.com/ru/articles/648759/>
36. <https://csharp.webdelphi.ru/dinamicheskie-komponenty-blazor-dynamiccomponent/>
37. https://www.reddit.com/r/dotnet/comments/1cd4622/what_do_people_use_for_charting_with_blazor/
38. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/?view=aspnetcore-9.0>
39. <https://www.cyberforum.ru/asp-net-core/thread2740415.html>
40. https://www.reddit.com/r/Blazor/comments/19d6682/data_visualizationcharts_and_blazor/
41. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/render-modes?view=aspnetcore-9.0>
42. https://www.stimulsoft.com/documentation/ru/programming-manual/reports_web_blazor_using_web_viewer_settings.htm
43. <https://ru.stackoverflow.com/questions/1512993/график-в-веб-приложении>
44. <https://habr.com/ru/articles/648759/>
45. <https://habr.com/ru/articles/581192/>
46. https://www.reddit.com/r/Blazor/comments/zfty41/i_made_a_few_small_demo_projects_to_showcase_how/
47. <https://learn.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore-9.0>
48. <https://temofeev.ru/info/articles/javascript-redaktor-svg-diagramm-kotoryyy-visit-v-6-5-raz-menshe-bootstrapping-biblioteka-s-otkrytym-iskh/>
49. <https://www.cyberforum.ru/asp-net-core/thread2740415.html>
50. https://www.reddit.com/r/dotnet/comments/1cd4622/what_do_people_use_for_charting_with_blazor/
51. <https://bhv.ru/tag/blazor/>
52. <https://habr.com/ru/articles/582874/>
53. https://vk.com/@we_use_js-javascript-redaktor-svg-diagramm-kotoryi-visit-v-65-raz-mens
54. <https://habr.com/ru/articles/648759/>
55. <https://blog.aspose.com/ru/drawing/use-system-drawing-blazor-webassembly-app-csharp/>
56. <https://habr.com/ru/articles/581192/>
57. <https://coderoad.ru/56504754/Создание-графиков-в-Blazor>
58. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/advanced-scenarios?view=aspnetcore-8.0>
59. <https://www.cyberforum.ru/asp-net-core/thread2740415.html>
60. <https://www.cyberforum.ru/asp-net-core/thread2740415.html>
61. https://www.reddit.com/r/Blazor/comments/19d6682/data_visualizationcharts_and_blazor/
62. https://www.reddit.com/r/Blazor/comments/19d6682/data_visualizationcharts_and_blazor/
63. https://www.reddit.com/r/Blazor/comments/19d6682/data_visualizationcharts_and_blazor/
64. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/render-modes?view=aspnetcore-9.0>
65. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/?view=aspnetcore-9.0>
66. <https://habr.com/ru/articles/546414/>
67. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/hosting-models?view=aspnetcore-9.0>
68. https://www.reddit.com/r/dotnet/comments/1n4vztb/server_wasm_or_autorender_mode/

69. https://www.reddit.com/r/csharp/comments/1azzi9e/difference_in_blazor_webassembly_and_server/
70. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/host-and-deploy/?view=aspnetcore-9.0>
71. <https://timeweb.cloud/tutorials/cloud/deploy-blazor-prilozheniya-v-timeweb-cloud-apps>
72. <https://ru.stackoverflow.com/questions/1534713/Разница-между-assembly-и-server-в-blazor>
73. https://www.reddit.com/r/Blazor/comments/1i23lem/why_is_blazor_web_app_with_webassembly/
74. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/render-modes?view=aspnetcore-9.0>
75. <https://dotnet8.patrickgod.com/posts/microsoft-finally-improved-global-interactive-render-modes-in-blazor-with-net-9>
76. <https://www.telerik.com/blogs/blazor-basics-benefits-static-server-side-rendering-blazor>
77. <https://learn.microsoft.com/en-us/aspnet/core/blazor/components/render-modes?view=aspnetcore-9.0>
78. <https://abp.io/community/articles/asp.net-core-blazor-9.0-new-features-summary--x0fovych>
79. <https://stackoverflow.com/questions/79406481/blazor-net-9-web-app-with-rendermode-auto-clarification>
80. https://www.youtube.com/watch?v=hLtQpPs_XeM
81. <https://github.com/dotnet/aspnetcore/issues/51046>
82. https://www.reddit.com/r/Blazor/comments/19drs0l/blazor_auto_render_mode_my_expectations_did_no_t/
83. https://www.linkedin.com/posts/gilbert-blankson-afful-14264582_all-c-net-9-blazor-render-modes-explained-activity-7268916922329563136-7KMr
84. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration?view=aspnetcore-9.0>
85. <https://habr.com/ru/companies/otus/articles/516434/>
86. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration-hosted-webassembly?view=aspnetcore-7.0>
87. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/class-libraries?view=aspnetcore-9.0>
88. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/host-and-deploy/webassembly/multiple-hosted-webassembly?view=aspnetcore-7.0>
89. <https://docs.astralinux.ru/latest/web/csharp/blazor/>
90. https://vk.com/wall-115048328_6645
91. https://www.reddit.com/r/dotnet/comments/1dk7hs1/how_to_choose_between_aspnet_core_mvc_asp_and/
92. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration?view=aspnetcore-9.0>
93. <https://habr.com/ru/companies/otus/articles/516434/>
94. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration?view=aspnetcore-9.0>
95. <https://csharp.webdelphi.ru/marshrutizaciya-v-prilozheniyax-blazor-hybrid/>
96. https://www.reddit.com/r/Blazor/comments/u4b7ex/route_to_razor_component_in_blazor_via_email_link/
97. https://www.reddit.com/r/Blazor/comments/19dqltq/how_to_utilize_in_blazor_net_80/
98. https://www.reddit.com/r/dotnetMAUI/comments/1140lzt/navigate_between_pages_razor_components_in_blazor/
99. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/host-and-deploy/?view=aspnetcore-9.0>
100. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/layouts?view=aspnetcore-9.0>

101. <https://csharp.webdelphi.ru/navigaciya-v-blazor-navigationmanager/>
102. <https://www.youtube.com/watch?v=MBHtJgyzLts>
103. https://www.reddit.com/r/dotnet/comments/17w7r4l/how_do_blazor_and_aspnet_fit_together_i_get_the/
104. https://storage.piter.com/upload/new_folder/978544612921/c_9_Net_5_Pril.pdf
105. <https://docs.astralinux.ru/latest/web/csharp/blazor/>
106. <https://habr.com/ru/companies/otus/articles/516434/>
107. <https://dotnet.microsoft.com/ru-ru/learn/aspnet/blazor-tutorial/intro>
108. https://www.reddit.com/r/Blazor/comments/17wlhcn/is_it_possible_to_mix_razor_pages_and_blazor_in/
109. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/?view=aspnetcore-9.0>
110. <https://learn.microsoft.com/ru-ru/dotnet/architecture/blazor-for-web-forms-developers/pages-routing-layouts>
111. <https://www.pandia.ru/text/77/209/21303.php>
112. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/fundamentals/routing?view=aspnetcore-9.0>
113. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration?view=aspnetcore-9.0>
114. <https://habr.com/ru/companies/otus/articles/516434/>
115. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration?view=aspnetcore-9.0>
116. <https://habr.com/ru/companies/otus/articles/516434/>
117. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/components/integration?view=aspnetcore-9.0>
118. https://www.reddit.com/r/Blazor/comments/146a17d/building_charts_with_chartjs_on_web_built_by/
119. https://www.reddit.com/r/Blazor/comments/1mez1xy/using_chartjs_in_blazor/
120. https://www.reddit.com/r/csharp/comments/xcjg94/anyone_tried_creating_a_dashboard_using_blazor/
121. https://www.reddit.com/r/csharp/comments/t57v5r/server_side_charting/