PREPARED BY JONATHAN RENATIUS

.NET 8 MVC TO BLAZOR

AGENDA

1. Introductions + Setup
2. What’s New in .NET 8
3. What is Blazor
4. MVC or Blazor
5. Blazor Server and WebAssembly
6. Coding in Blazor
7. Mini Lab
8. Review and Discussions

LEARNING OBJECTIVES

- Concept of Modern Web Applications
- Difference between MVC and Blazor Server and Blazor WASM
- How to Develop in .NET 8 Blazor

PREREQUISITES

- Visual Studio 2022 with .NET 8 Runtime Installed
- Experience with .NET 4.8 MVC
- Knowledge of C#, HTML, CSS, Javascript

GITHUB REPO

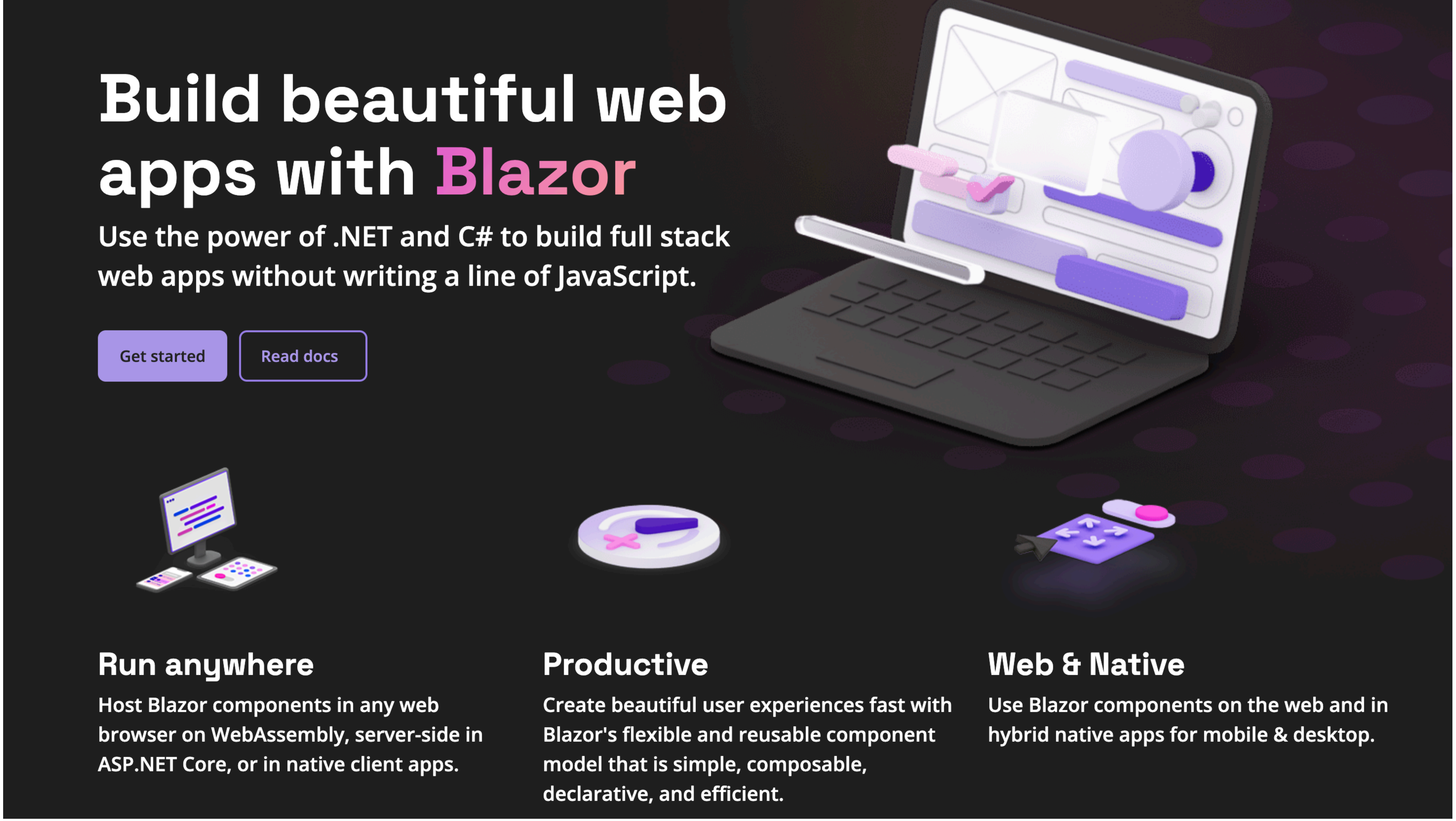
- https://github.com/jonathan-kairos/mvc-to-blazor

WHAT'S NEW?

.NET 4.8 TO .NET 8

- Cross-platform
- Dependency Injection Support
- Project and App Configuration
- C#12
- Performance

WHAT IS BLAZOR?



Build beautiful web apps with Blazor

Use the power of .NET and C# to build full stack web apps without writing a line of JavaScript.

Run anywhere

Host Blazor components in any web browser on WebAssembly, server-side in ASP.NET Core, or in native client apps.

Productive

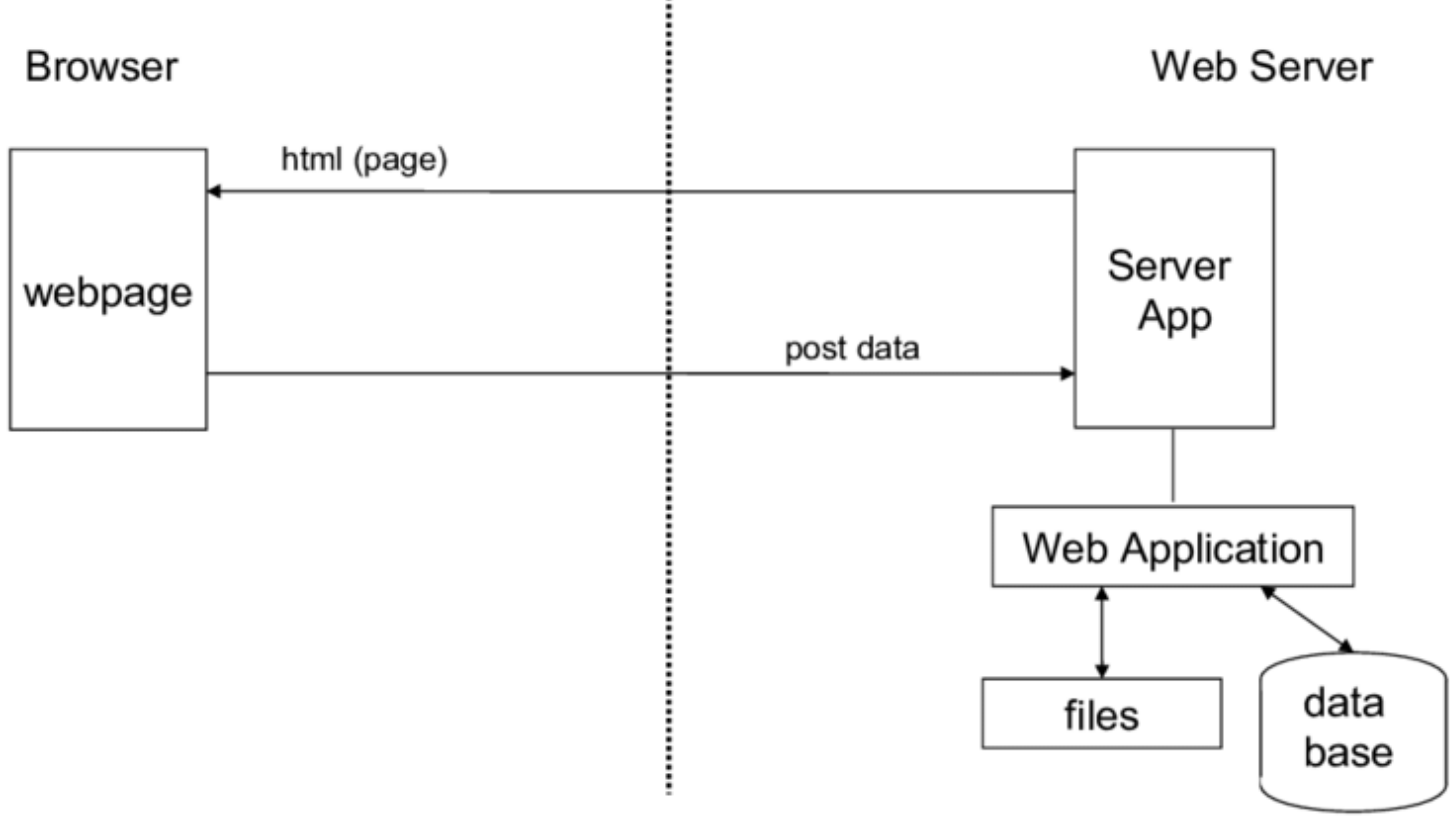
Create beautiful user experiences fast with Blazor's flexible and reusable component model that is simple, composable, declarative, and efficient.

Web & Native

Use Blazor components on the web and in hybrid native apps for mobile & desktop.

BLAZOR IS AN SPA FRAMEWORK IN .NET FOR BUILDING INTERACTIVE APPS USING C#

TRADITIONAL WEB



https://www.researchgate.net/profile/Jan-Martin-Jansen/publication/230607097/figure/fig1/AS:669384917938180@1536605209359/The-traditional-architecture-of-an-Internet-Application.png

MVC OR BLAZOR?

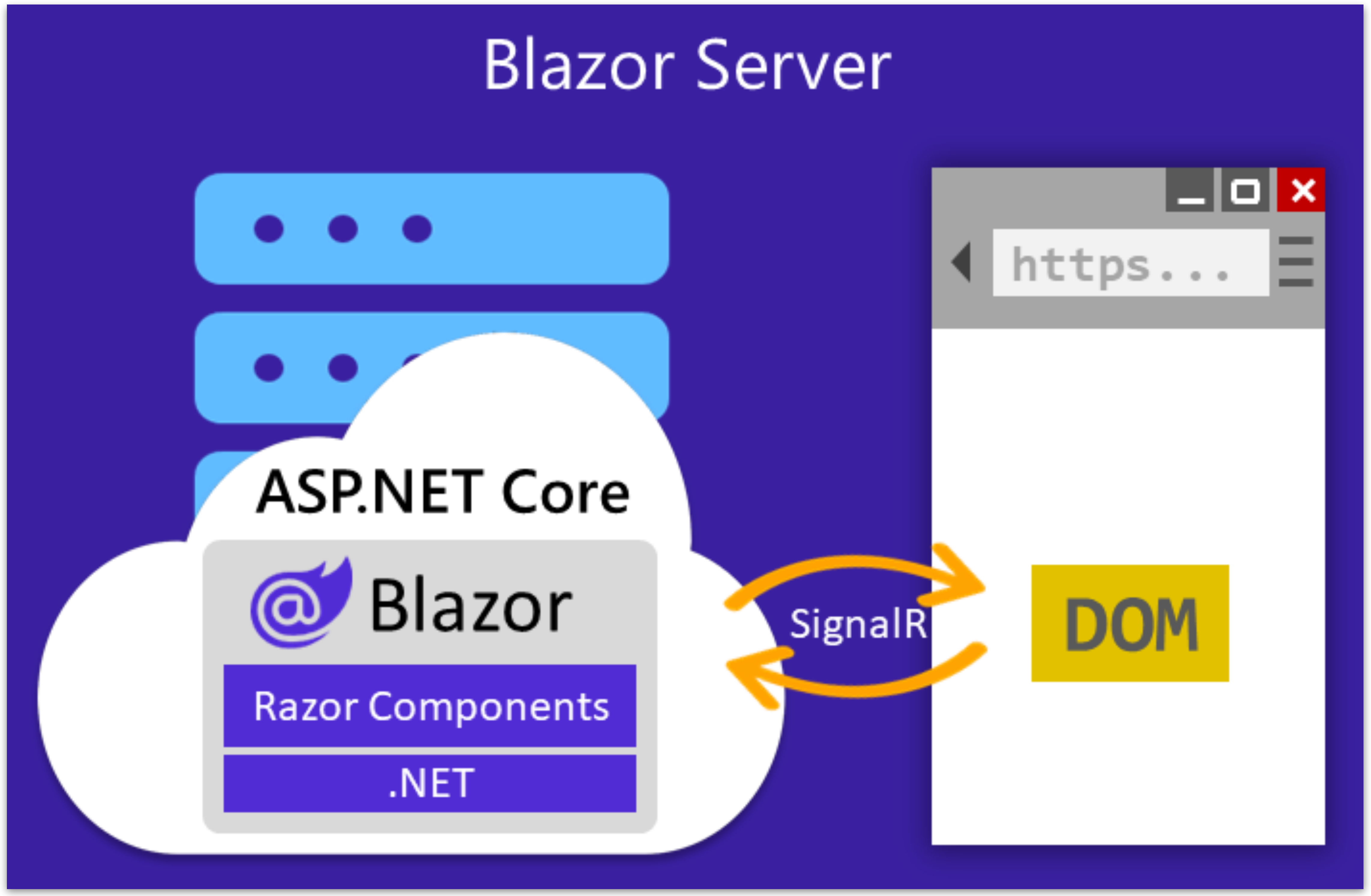
WHY MVC?

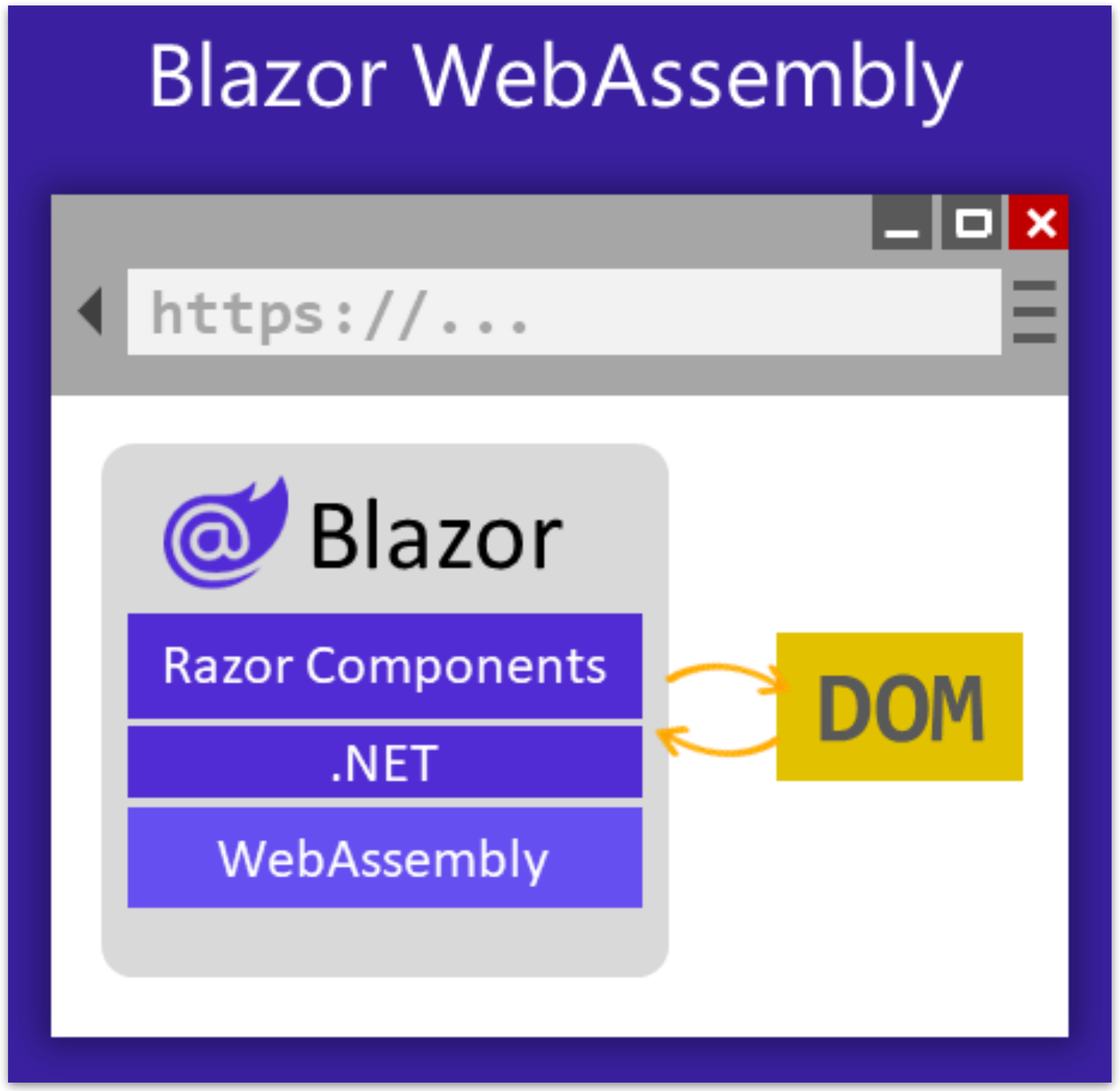
- Mature
- Flexible: Control over HTML, CSS, Javascript
- Integrations: With Libraries and Platforms

WHY BLAZOR?

- Unified Backend and Frontend
- Component-based: Modular and Reusable
- Cross-platform

BLAZOR SERVER AND WEBASSEMBLY





https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-8.0

HANDS ON

TASK 1: CREATE A NEW COMPONENT

Objective: Learn how to create and use a Razor component.

Create a new Razor component named Counter. This component should include a button that, when clicked, increments a counter.

TASK 2: COMPONENT PARAMETERS

Objective: Understand how to pass parameters to components.

Task: Modify the Counter component to accept a parameter that sets the initial count value.

TASK 3: EVENT HANDLING

Objective: Learn how to handle events in Blazor.

Task: Extend the Counter component to include an event that triggers when the count is incremented and sends the updated count back to the parent component.

TASK 4: DATA BINDING

Objective: Gain familiarity with data binding techniques in Blazor.

Task: Add an input field that allows users to set the initial count value when the button is clicked

TASK 5: LIFECYCLE METHODS

Objective: Understand the component lifecycle in Blazor.

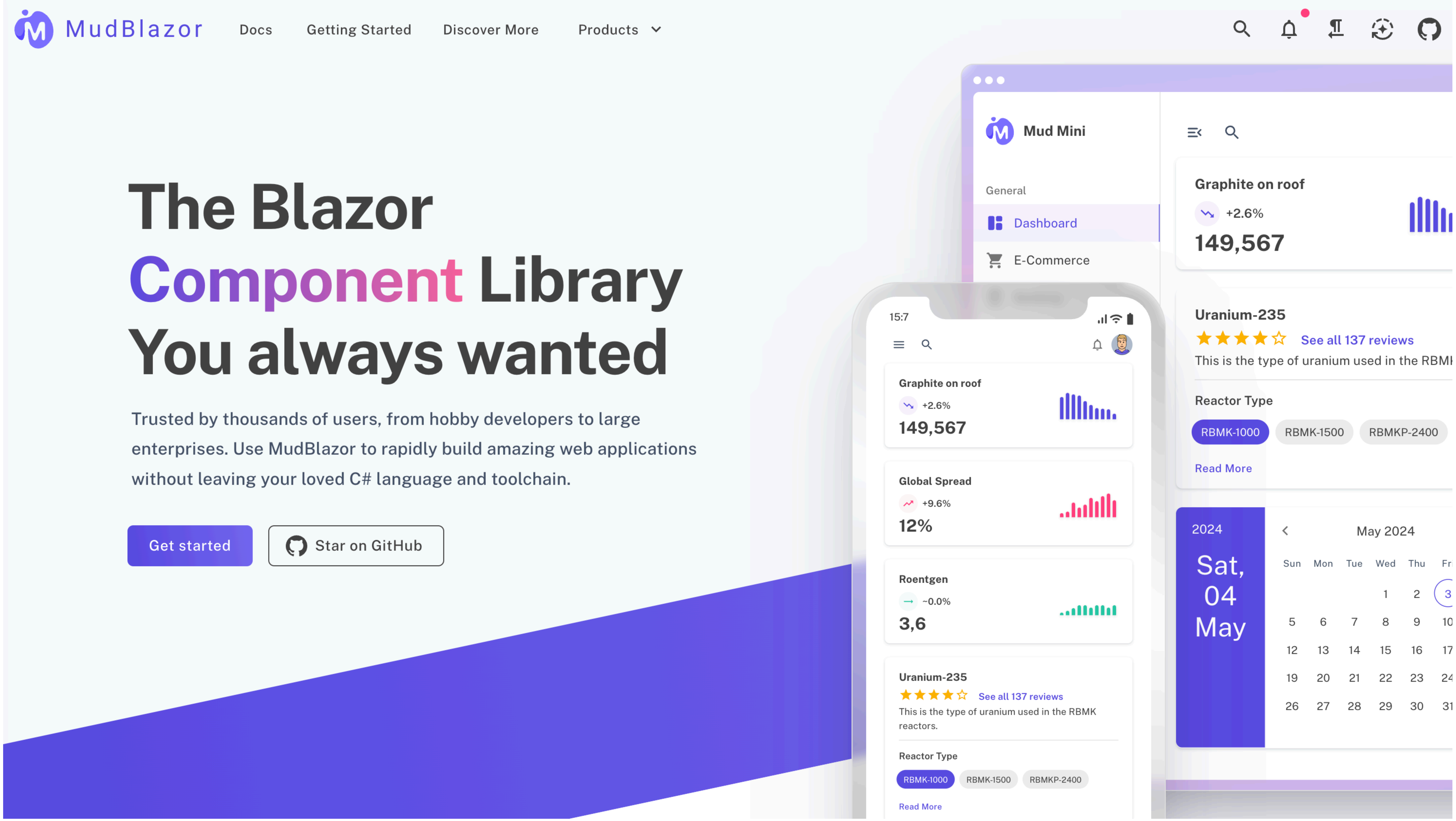
Task: Use the OnParametersSet method to update the currentCount whenever the InitialCount parameter changes

TASK 6: DEPENDENCY INJECTION

Objective: Learn about using services in Blazor components.

Task: Use the CounterService class to get and set the count in Counter component

MUDBLAZOR


MudBlazor Docs Getting Started Discover More Products

The Blazor Component Library You always wanted

Trusted by thousands of users, from hobby developers to large enterprises. Use MudBlazor to rapidly build amazing web applications without leaving your loved C# language and toolchain.

https://mudblazor.com/

INVENTORY TRACKER APP

MINI LAB

MINI LAB TASKS

1. Create a button called Check Stock. If the item quantity is 10 or less, highlight the record red
2. Implement a Create Inventory Item page
3. Implement data validation for the Create based on the InventoryItem model
4. Implement the Inventory History page using MudBlazor

THANK YOU