Creating Blazor Components

WRITING YOUR FIRST BLAZOR COMPONENT

Roland Guijt
MICROSOFT MVP, CONSULTANT, AUTHOR AND SPEAKER

@rolandguijt rolandguijt.com

Module Overview

Writing a basic component

Rendering components

Using Razor class libraries

Structuring code

Event Handling

One-way data binding

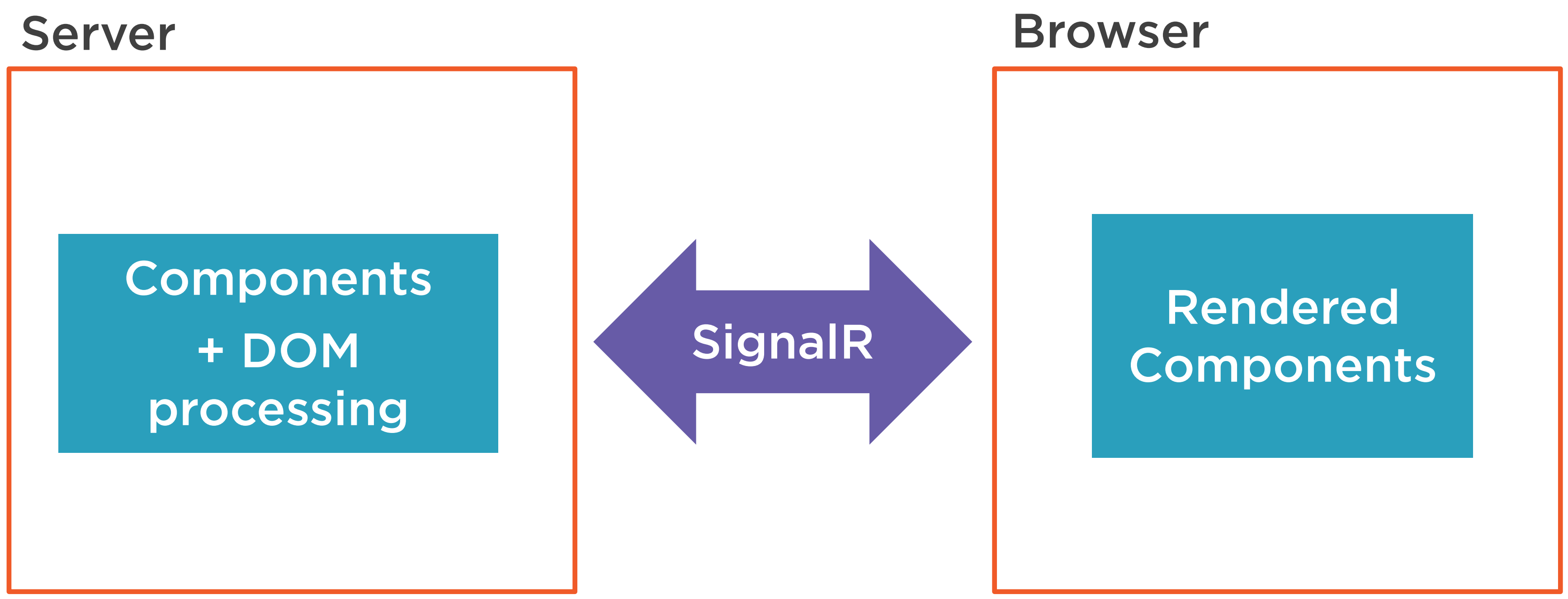
The diff mechanism

Child content

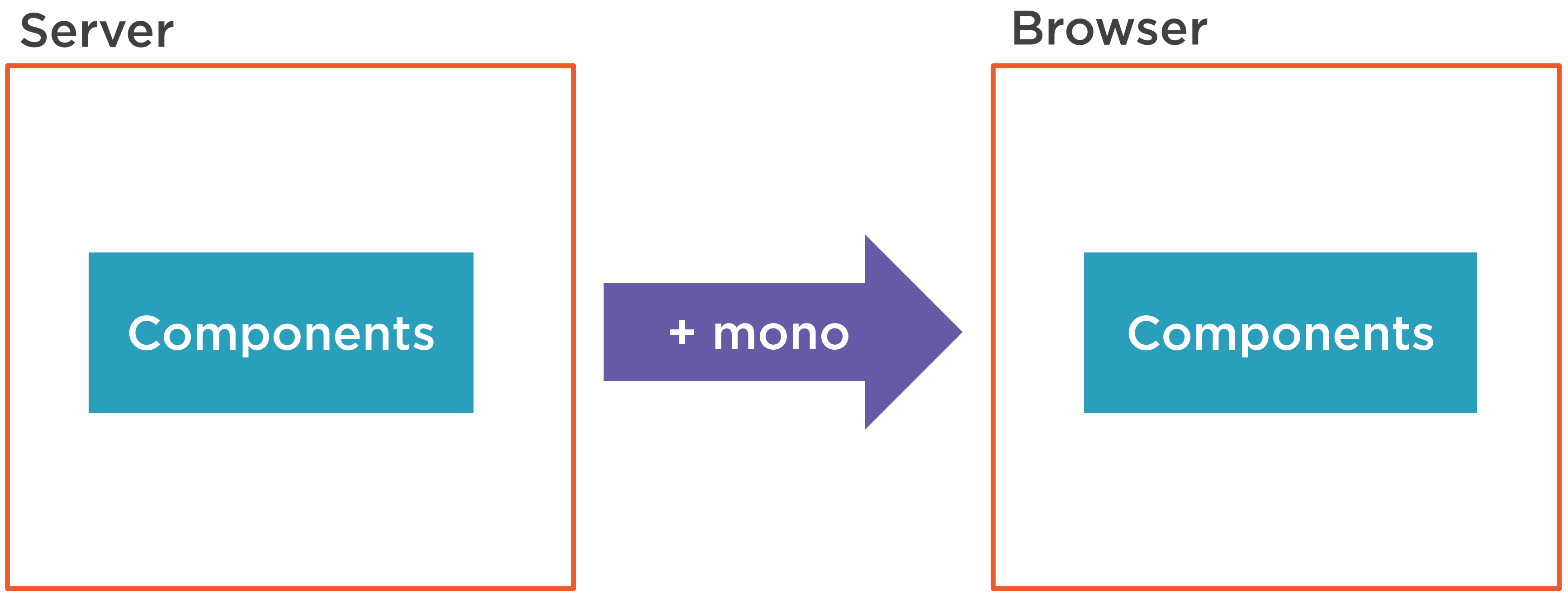
Components use Razor syntax

Components in Components

Hosting Model #1: Blazor Server



Hosting Model #2: Blazor Web Assembly

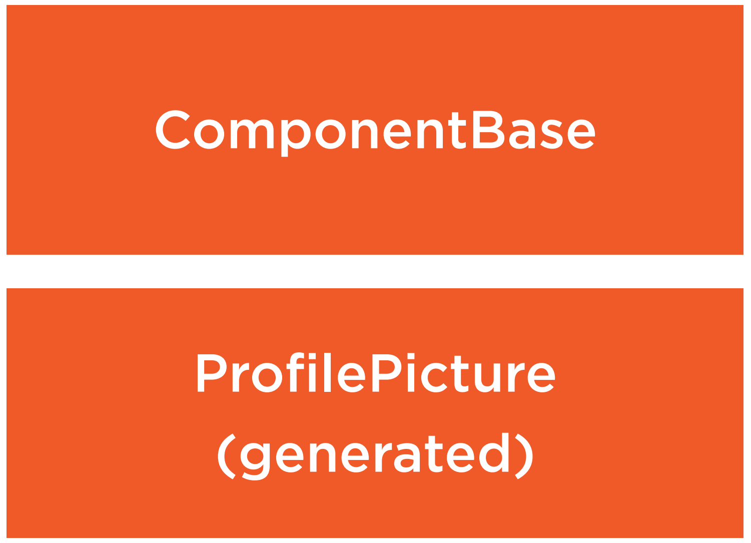


Server and Web Assembly Components

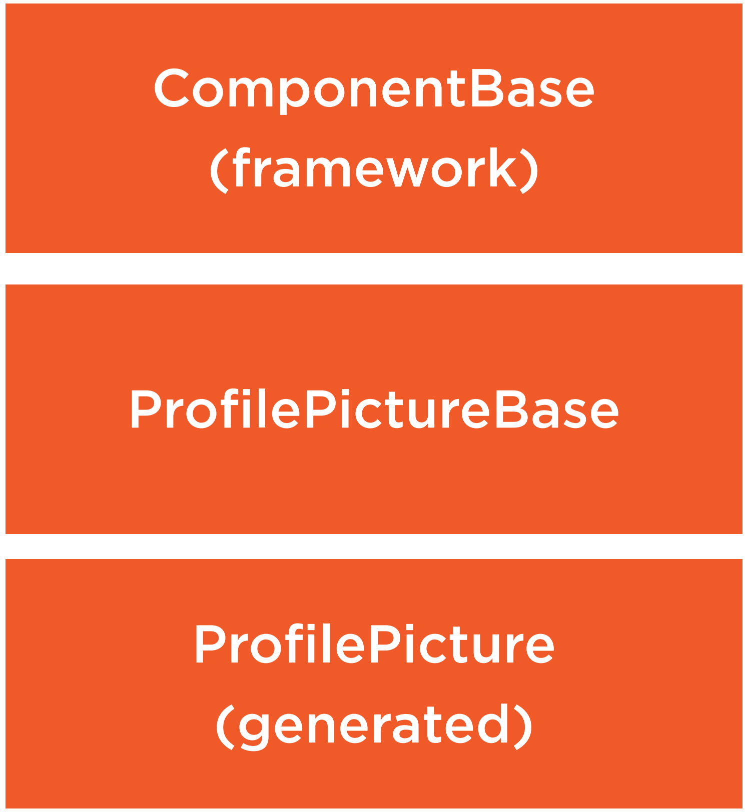
What you can do differs

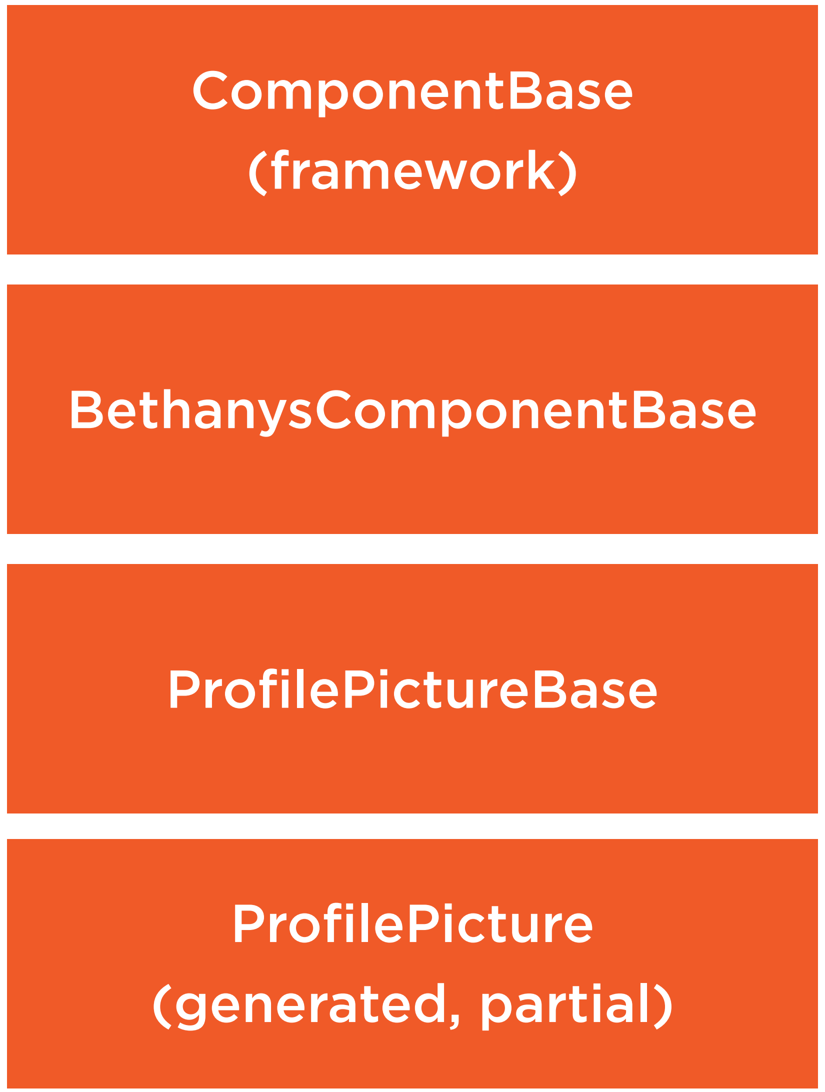
Structure and features identical

Partial Component Class Hierarchy Without Code-behind



Partial Component Class Hierarchy with Code-behind




ComponentBase
(framework)
BethanysComponentBase
ProfilePictureBase
ProfilePicture
(generated, partial)

Handling Events

```
<img .. @onclick="ProfileClick"/>
```

```
<h2>@message</h2>

@for (var i = 1; i < 4; i++)
{
    var buttonNumber = i;

    <button class="btn btn-primary"
            @onclick="@(e => UpdateHeading(e, buttonNumber))">
        Button #@i
    </button>
}

@code {
    private string message = "Select a button to learn its position.";

    private void UpdateHeading(MouseEventArgs e, int buttonNumber)
    {
        message = $"You selected Button #{buttonNumber} at " +
            $"mouse position: {e.ClientX} X {e.ClientY}.";
    }
}
```

The Diff Mechanism

Old

```
<div>
   <img class = "circle"/>
</div>
```

New

```
<div>
   <img class = ""/>
</div>
```

Update

```
<div>
   <img class = ""/>
</div>
```

Summary

Components are reusable pieces of UI

Anatomy of a component

Handling events

Data binding

Child content