

# Developing Cross-Platform Web Apps With Blazor

Wael Kdouh - @waelkdouh

Senior Customer Engineer

v1.0

## Conditions and Terms of Use

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

## Copyright and Trademarks

© 2013 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at

<http://www.microsoft.com/about/legal/permissions/>

Active Directory, Azure, IntelliSense, Internet Explorer, Microsoft, Microsoft Corporate Logo, Silverlight, SharePoint, SQL Server, Visual Basic, Visual Studio, Windows, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

# How to View This Presentation

- To switch to **Notes Page** view:
  - On the ribbon, click the **View** tab, and then click **Notes Page**
- To navigate through notes, use the Page Up and Page Down keys
  - Zoom in or zoom out, if required
- In the **Notes Page** view, you can:
  - Read any supporting text—now or after the delivery
  - Add notes to your copy of the presentation, if required
- Take the presentation files home with you

# Module 5: Routing

## Module Overview

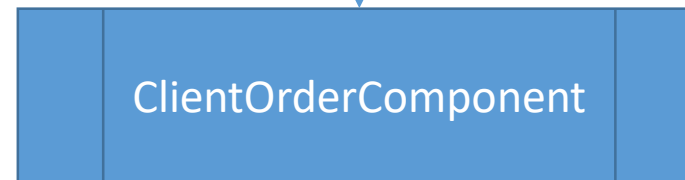
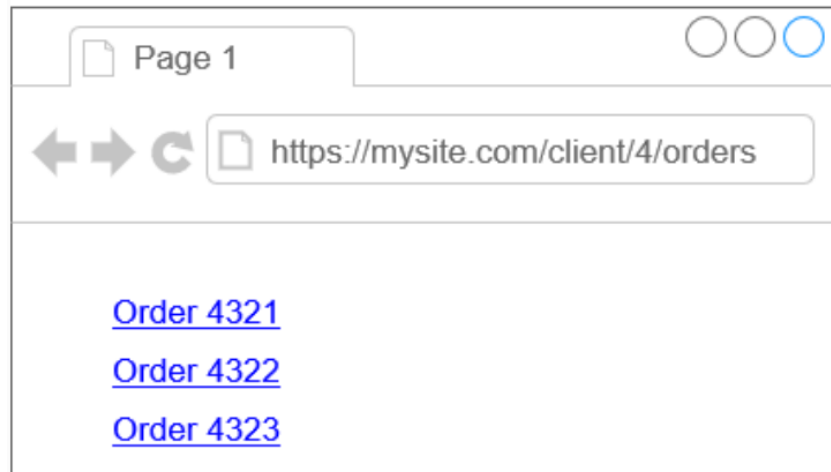
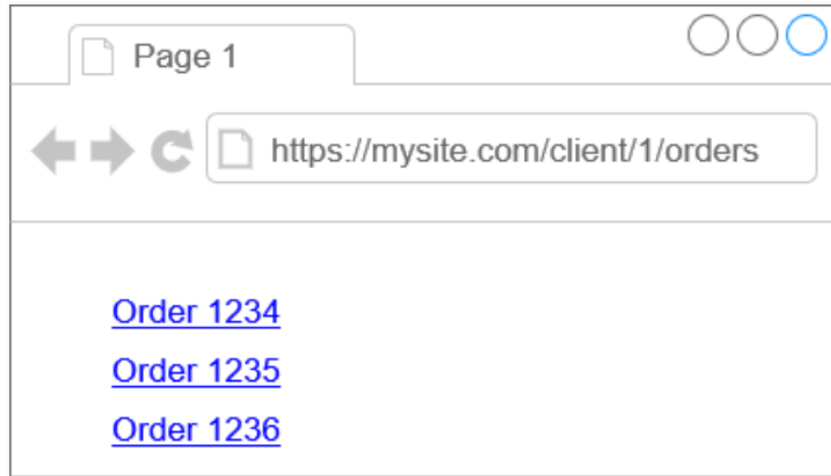
Module 5: Routing

Section 1: Routing

Lesson: Overview

# Routing

- As with a standard ASP.NET MVC, Blazor **routing** is a technique for inspecting the browser's **URL** and **matching it up to a page to render**



Routing is more flexible than simply matching a URL to a page. It allows us to match based on patterns of text so that, for example, both URLs in the preceding image will map to the same component and pass in an ID for context (either a 1 or a 4 in this example)

# Simulated Navigation

- Navigating to a new URL within the same app doesn't navigate in the traditional WWW sense
- No request is sent to the server requesting the content for the new page. Instead, Blazor rewrites the browser's URL and then renders the relevant content
- When a navigation is made to a new URL that resolves to the same type of component as the current page, the component will not be destroyed before navigation. The `OnInitialized` lifecycle methods will not be executed. The navigation is simply seen as a change to the component's parameters

# Defining Routes

- To define a route add a @page declaration at the top of the component

```
@page "/"
```

```
<h1>Hello, world!</h1>
```

```
Welcome to your new app.
```

```
<SurveyPrompt Title="How is Blazor working for you?" />
```



# Defining Routes

- If you open the **generated source** code for index.razor (in **\obj\Debug\netstandard2.1\Razor\Pages**) you will see the @page directive compiled to the following code:

```
[Microsoft.AspNetCore.Components.RouteAttribute("/")]
```

```
public partial class Index : Microsoft.AspNetCore.Components.ComponentBase
{
    #pragma warning disable 1998
    protected override void BuildRenderTree(Microsoft.AspNetCore.Components.Rendering.RenderTreeBuilder __builder)
    {
        __builder.AddMarkupContent(0, "<h1>Hello, world!</h1>\r\n\r\nWelcome to your new app.\r\n\r\n");
        __builder.OpenComponent<Module5.Shared.SurveyPrompt>(1);
        __builder.AddAttribute(2, "Title", "How is Blazor working for you?");
        __builder.CloseComponent();
    }
    #pragma warning restore 1998
}
```

# Defining Routes

- During start-up, Blazor scans for classes decorated with RouteAttribute and builds its route definitions accordingly

```
[Microsoft.AspNetCore.Components.RouteAttribute("/")]
```

```
public partial class Index : Microsoft.AspNetCore.Components.ComponentBase
{
    #pragma warning disable 1998
    protected override void BuildRenderTree(Microsoft.AspNetCore.Components.Rendering.RenderTreeBuilder __builder)
    {
        __builder.AddMarkupContent(0, "<h1>Hello, world!</h1>\r\n\r\nWelcome to your new app.\r\n\r\n");
        __builder.OpenComponent<Module5.Shared.SurveyPrompt>(1);
        __builder.AddAttribute(2, "Title", "How is Blazor working for you?");
        __builder.CloseComponent();
    }
    #pragma warning restore 1998
}
```

# Route Discovery

- Route discovery is performed automatically by Blazor in its default project template
- If you look inside the App.razor file you will see the use of a Router component:

```
<Router AppAssembly="@typeof(Program).Assembly">
  <Found>...</Found>
  <NotFound>
    <LayoutView>...</LayoutView>
  </NotFound>
</Router>
```

# Route Discovery

The Router component scans all classes within the specified assembly that implement IComponent, it then reflects over the class to see if it is decorated with any RouteAttribute attributes

```
<Router AppAssembly="@typeof(Program).Assembly">
  <Found>...</Found>
  <NotFound>
    <LayoutView>...</LayoutView>
  </NotFound>
</Router>
```

For each RouteAttribute it finds, it parses its URL template string and adds a relationship from the URL to the component into its internal route table

# Route Discovery

- A single component may be decorated with zero, one, or many RouteAttribute attributes (@page declarations)
- A component with zero attributes cannot be reached via a URL, whereas a component with multiple attributes can be reached via any of the URL templates it specifies

```
Counter.razor  + X
1  @page "/"counter"
2  @page "/"counter/{CurrentCount:int}"
3
4  <h1>Counter</h1>
5
6  <p>Current count: @CurrentCount</p>
7
8  <button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
9
```

Module 5: Routing

Section 1: Routing

Lesson: Route Parameters

# Route Parameters

- If you want the same component to render different views based on information in the URL (such as a customer ID) then you would need to use route parameters

```
@page "/counter"
@page "/counter/{CurrentCount:int}"

<h1>Counter</h1>

<p>Current count: @CurrentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

<ul>
  <li><a href="/counter/42">Navigate to /counter/42</a></li>
  <li><a href="/counter/123">Navigate to /counter/123</a></li>
  <li><a href="/counter/">Navigate to /counter</a></li>
</ul>

@code {
    [Parameter]
    public int? CurrentCount { get; set; }
```

A route parameter is defined in the URL by wrapping its name in a pair of { braces } when adding a component's @page declaration

Capturing the value of a parameter is as simple as adding a property with the same name and decorating it with a [Parameter] attribute

# Route Parameters

- If you want the same component to render different views based on information in the URL (such as a customer ID) then you would need to use route parameters

```
@page "/counter"
```

```
@page "/counter/{CurrentCount:int}"
```

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

```
<p>Current count: @CurrentCount</p>
```

```
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
```

```
<ul>  
  <li><a href="/counter/42">Navigate to /counter/42</a></li>  
  <li><a href="/counter/123">Navigate to /counter/123</a></li>  
  <li><a href="/counter/">Navigate to /counter</a></li>  
</ul>
```

```
@code {  
    [Parameter]  
    public int? CurrentCount { get; set; }  
}
```

When a navigation is made to a new URL that resolves to the same type of component as the current page, the component will not be destroyed before navigation and the `OnInitialized` lifecycle methods will not be executed. The navigation is simply seen as a change to the component's parameters



# Constraining Route Parameters

Will only match a URL to a component if the value of CurrentCount is an integer

```
@page "/counter"
```

```
@page "/counter/{CurrentCount:int}"
```

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

```
<p>Current count: @CurrentCount</p>
```

```
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
```

```
<ul>
  <li><a href="/counter/42">Navigate to /counter/42</a></li>
  <li><a href="/counter/123">Navigate to /counter/123</a></li>
  <li><a href="/counter/">Navigate to /counter</a></li>
</ul>
```

```
@code {
    [Parameter]
    public int? CurrentCount { get; set; }
```

# Constraining Route Parameters

Constraint	.NET type	Valid	Invalid
bool	System.Boolean	<ul style="list-style-type: none"><li>•true</li><li>•false</li></ul>	<ul style="list-style-type: none"><li>•1</li><li>•Hello</li></ul>
:datetime	System.DateTime	<ul style="list-style-type: none"><li>•2001-01-01</li><li>•02-29-2000</li></ul>	<ul style="list-style-type: none"><li>•29-02-2000</li></ul>
:decimal	System.Decimal	<ul style="list-style-type: none"><li>•2.34</li><li>•0.234</li></ul>	<ul style="list-style-type: none"><li>•<b>2,34</b></li></ul>
:double	System.Double	<ul style="list-style-type: none"><li>•2.34</li><li>•0.234</li></ul>	<ul style="list-style-type: none"><li>•<b>2,34</b></li></ul>
:float	System.Single	<ul style="list-style-type: none"><li>•2.34</li><li>•0.234</li></ul>	<ul style="list-style-type: none"><li>•<b>2,34</b></li></ul>
:guid	System.Guid	<ul style="list-style-type: none"><li>•99303dc9-8c76-42d9-9430-de3ee1ac25d0</li></ul>	<ul style="list-style-type: none"><li>•{99303dc9-8c76-42d9-9430-de3ee1ac25d0}</li></ul>
:int	System.Int32	<ul style="list-style-type: none"><li>•-1</li><li>•42</li></ul>	<ul style="list-style-type: none"><li>•<b>12.34</b></li></ul>
:long	System.Int64	<ul style="list-style-type: none"><li>•-1</li><li>•42</li></ul>	<ul style="list-style-type: none"><li>•<b>12.34</b></li></ul>

# Localization

- Blazor constraints do not currently support localization
  - Numeric digits are only considered valid if they are in the form 0..9, and not from a non-English language such as ૦..૯ (Gujarati)
  - Dates are only valid in the form MM-dd-yyyy, MM-dd-yy, or in ISO format yyyy-MM-dd
  - Boolean values must be true or false

# Unsupported Constraint Types

- Blazor constraints do not support the following constraint types:
  - **Greedy parameters**
    - In ASP.NET Core MVC it is possible to provide a parameter name that starts with an asterisk and catches a chunk of the URL including forward slashes
    - `/articles/{Subject}/{*TheRestOfTheURL}`
  - **Regular expressions**
    - Blazor does not currently support the ability to constrain a parameter based on a regular expression
  - **Enums**
    - It's not currently possible to constrain a parameter to match a value of an enum
  - **Custom constraints**
    - It is not possible to define a custom class that determines whether or not a value passed to a parameter is valid

# Optional Route Parameters

- Optional route parameters aren't supported explicitly by Blazor, but the equivalent can be easily achieved by adding more than one @page declaration on a component

---

```
@page "/counter"
```

```
@page "/counter/{CurrentCount:int}"
```

# Specifying a Default Value For Optional Parameters

```
@page "/counter"
@page "/counter/{CurrentCount:int}"

<h1>Counter</h1>

<p>Current count: @CurrentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

<ul>
    <li><a href="/counter/42">Navigate to /counter/42</a></li>
    <li><a href="/counter/123">Navigate to /counter/123</a></li>
    <li><a href="/counter/">Navigate to /counter</a></li>
</ul>

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

    public async override Task SetParametersAsync(ParameterView parameters)
    {
        await base.SetParametersAsync(parameters);
        CurrentCount = CurrentCount ?? 1;
        Console.WriteLine("SetParametersAsync lifecycle hook called");
    }

    private void IncrementCount()
    {
        CurrentCount++;
    }
}
```

SetParametersAsync is called whenever the parameters change and their values are pushed into the component's properties, such as during a navigation

null coalescing operator (??) ensures that a value is set for CurrentCount if /counter is entered into the url

# Demo: Route Parameters

## 404 – Not found

- When Blazor fails to match a URL to a component we might want to tell it what content to display
- The Router component has a parameter named **NotFound** which **is a RenderFragment**. Any Razor mark-up defined within this parameter of the Router component will be displayed when attempting to reach a URL the router cannot match to any component



Module 5: Routing

Section 1: Routing

Lesson: Navigation Options

# Navigating The App Via HTML

- The simplest way to link to a route within a Blazor component is to use an HTML hyperlink

```
@page "/counter"  
@page "/counter/{CurrentCount:int}"
```

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

```
<p>Current count: @CurrentCount</p>
```

```
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>
```

```
<ul>  
  <li><a href="/counter/42">Navigate to /counter/42</a></li>  
  <li><a href="/counter/123">Navigate to /counter/123</a></li>  
  <li><a href="/counter/">Navigate to /counter</a></li>  
</ul>
```

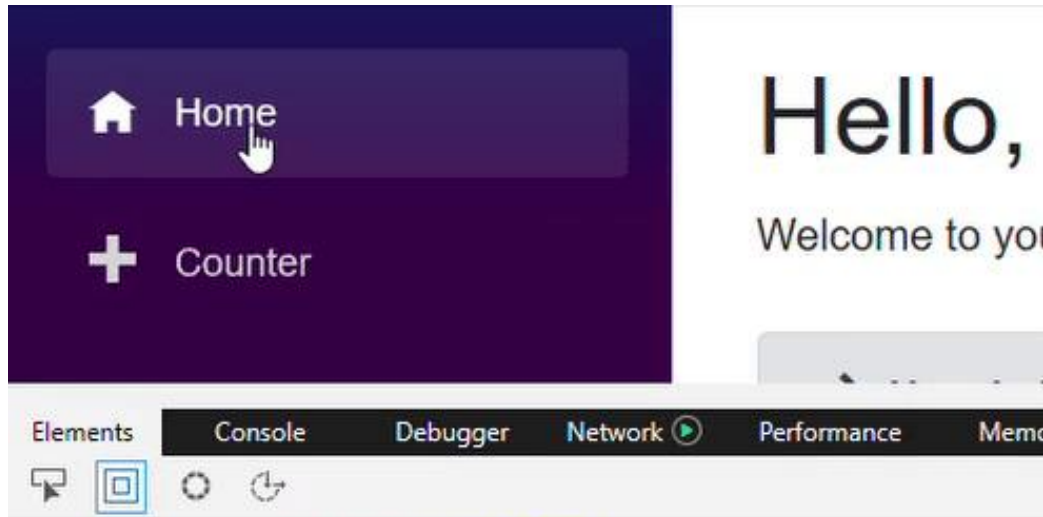
Hyperlinks in Blazor components are intercepted automatically. When a user clicks a hyperlink the browser will not send a request to the server, instead Blazor will update the URL in the browser and render whichever page is associated with the new address

# Using the NavLink Component

- Blazor also includes a component for rendering hyperlinks with **additional support for changing the HTML element's CSS class when address matches the URL**
- If you look inside the /Shared/NavMenu.razor component in the default Blazor application you will see the following markup:

```
<div class="@NavMenuCssClass" @onclick="ToggleNavMenu">
  <ul class="nav flex-column">
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
        <span class="oi oi-home" aria-hidden="true"></span> Home
      </NavLink>
    </li>
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="counter">
        <span class="oi oi-plus" aria-hidden="true"></span> Counter
      </NavLink>
    </li>
  </ul>
</div>
```

# Using the NavLink Component



```
<li class="nav-item px-3">
  <!--!-->
  <!--!-->
  <a class="nav-link active" href="" match="All">
    <!--!-->
    <span class="oi oi-home" aria-hidden="true"></span>
    Home
  </a>
  <!--!-->
</li>
<!--!-->
<li class="nav-item px-3">
  <!--!-->
  <!--!-->
  <a class="nav-link" href="counter">...</a>
  <!--!-->
</li>
```

The ActiveClass parameter specifies which CSS class to apply to the rendered `<a>` element when the URL of the browser matches the URL of the href attribute. If not specified, Blazor will apply a CSS class named "active"

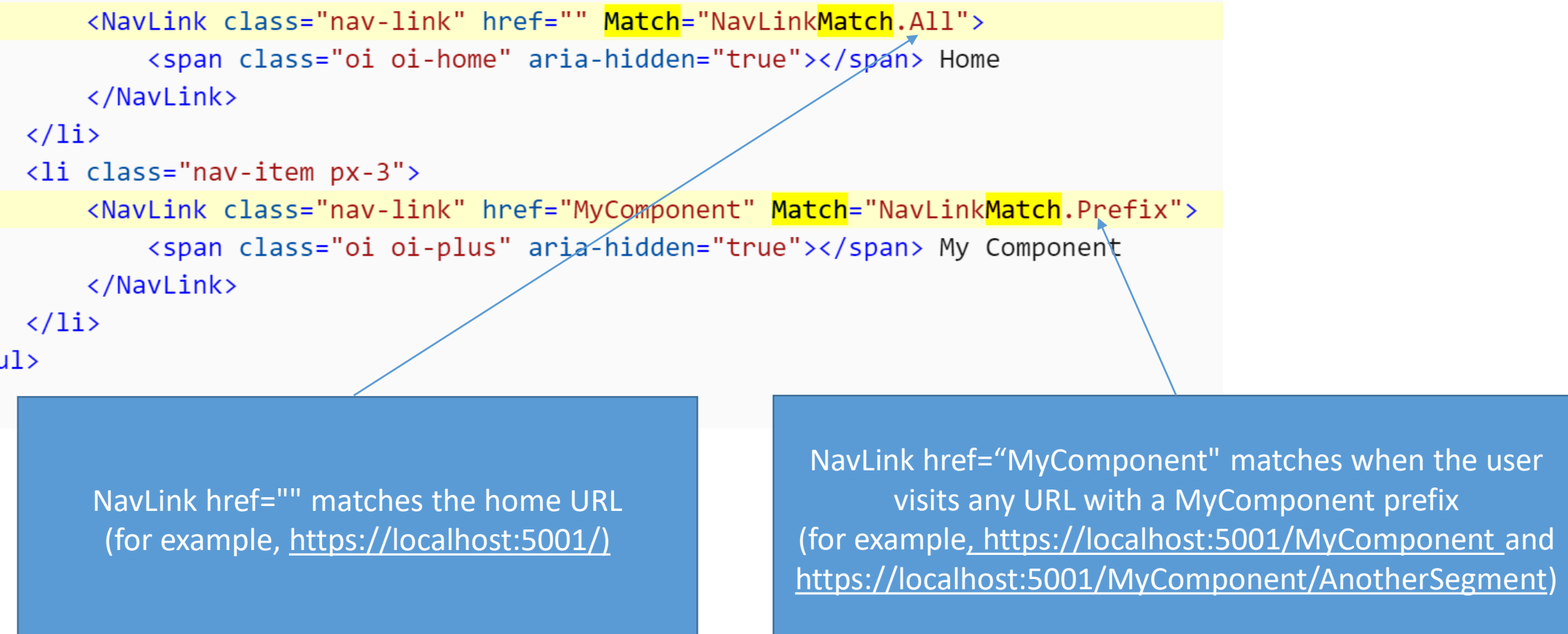
# Demo: ActiveClass

# NavLinkMatch

- The **Match parameter** of the NavLink component accepts a value of the type NavLinkMatch. This tells the NavLink component how you want the browser's URL compared with the href attribute of the <a> element it renders to determine whether they are the same or not
- There are two NavLinkMatch options that you can assign to the Match attribute of the <NavLink> element:
  - **NavLinkMatch.All** – The NavLink is active when it matches the entire current URL
  - **NavLinkMatch.Prefix** (default) – The NavLink is active when it matches any prefix of the current URL

# NavLinkMatch

```
<div class="@NavMenuCssClass" @onclick="@ToggleNavMenu">
  <ul class="nav flex-column">
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
        <span class="oi oi-home" aria-hidden="true"></span> Home
      </NavLink>
    </li>
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="MyComponent" Match="NavLinkMatch.Prefix">
        <span class="oi oi-plus" aria-hidden="true"></span> My Component
      </NavLink>
    </li>
  </ul>
</div>
```



NavLink href="" matches the home URL  
(for example, <https://localhost:5001/>)

NavLink href="MyComponent" matches when the user  
visits any URL with a MyComponent prefix  
(for example, <https://localhost:5001/MyComponent> and  
<https://localhost:5001/MyComponent/AnotherSegment>)

# Demo: URL Matching



Module 5: Routing

Section 1: Routing

Lesson: Navigating With Code

# Navigating App Via Code

- Access to browser navigation from Blazor is provided via the **NavigationManager service**. This can be injected into a Blazor component using `@inject` in a razor file, or the `[Inject]` attribute in a CS file (when using code behind)

# Navigating App Via Code

Member	Description
<code>Uri</code>	Gets the current absolute URI.
<code>BaseUri</code>	Gets the base URI (with a trailing slash) that can be prepended to relative URI paths to produce an absolute URI. Typically, <code>BaseUri</code> corresponds to the <code>href</code> attribute on the document's <code>&lt;base&gt;</code> element in <i>wwwroot/index.html</i> (Blazor WebAssembly) or <i>Pages/_Host.cshtml</i> (Blazor Server).
<code>NavigateTo</code>	<p>Navigates to the specified URI. If <code>forceLoad</code> is <code>true</code>:</p> <ul style="list-style-type: none"><li>• Client-side routing is bypassed.</li><li>• The browser is forced to load the new page from the server, whether or not the URI is normally handled by the client-side router.</li></ul>
<code>LocationChanged</code>	An event that fires when the navigation location has changed.
<code>ToAbsoluteUri</code>	Converts a relative URI into an absolute URI.
<code>ToBaseRelativePath</code>	Given a base URI (for example, a URI previously returned by <code>GetBaseUri</code> ), converts an absolute URI into a URI relative to the base URI prefix.

# Navigating App Via Code

- The following component navigates to the app's Counter component when the button is selected:

```
@page "/navigate"
@Inject NavigationManager NavigationManager

<h1>Navigate in Code Example</h1>

<button class="btn btn-primary" @onclick="NavigateToCounterComponent">
    Navigate to the Counter component
</button>

@code {
    private void NavigateToCounterComponent()
    {
        NavigationManager.NavigateTo("counter");
    }
}
```

# Demo: Navigating App Via Code

# Module Summary

- In this module, you learned about:
  - Defining Routes
  - Route Parameters
  - Constraining Route Parameters
  - Optional Route Parameters
  - Navigating Via HTML
  - Navigating Via Code
  - Detecting Navigation Events





# Lab 5: Routing



# References

- [Microsoft Docs](#)
- [Blazor University](#)



