

Integrating JavaScript Components



Daniel Villamizar

Senior Cloud Solutions Architect - MVP

@danielvillamizara – <https://www.linkedin.com/in/danielvillamizara/>

Module overview



Invoking JavaScript from Blazor

Wrapping components in a Razor Class Library

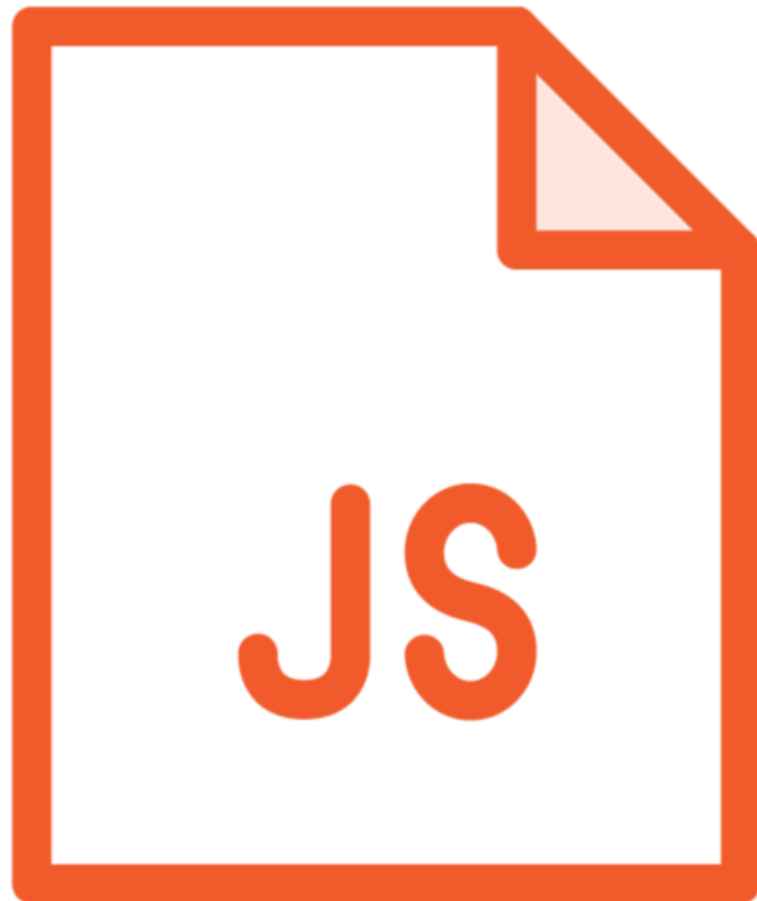
Lazy loading libraries

Invoking JavaScript from Blazor



In a Blazor WebAssembly app,
we're “just” running
a web page on the client.

So, JavaScript will work fine
here too.



Not everything is possible via just .NET

JavaScript interop

- Call into JavaScript from Blazor code

Runs on the client

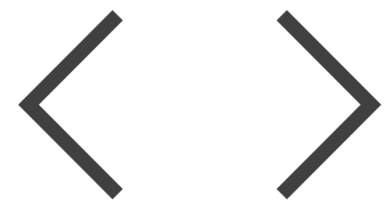
- Loaded via surrounding page

Blazor and JavaScript Interop

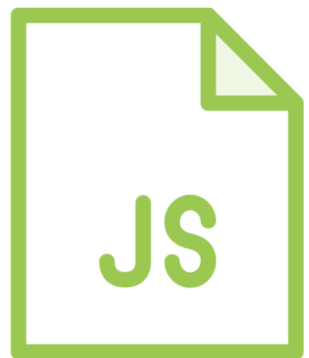
.NET calls JavaScript

JavaScript calls .NET

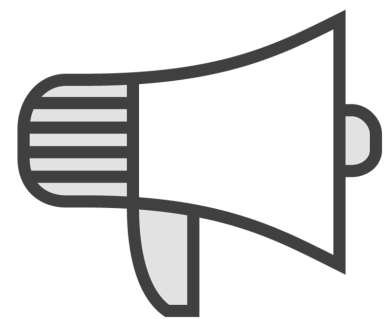
Steps to Invoke JavaScript from .NET



Register script in surrounding page



Use IJSRuntime via Dependency Injection



Call script from .NET

```
<script>  
  window.DoSomething = () => {  
    //do some interesting task here  
  }  
</script>
```

Adding a Script


```
[Inject]  
public IJSRuntime JsRuntime { get; set; }
```

Bring in IJSRuntime

```
var result = await JsRuntime.InvokeAsync<object>("DoSomething", "");
```

Invoking a JavaScript Function

Available Methods

InvokeVoidAsync()

InvokeAsync()



Working with JavaScript Interop

Working with JavaScript can only be
done when the component is done
rendering!

Use `OnAfterRenderAsync()` for this

Demo

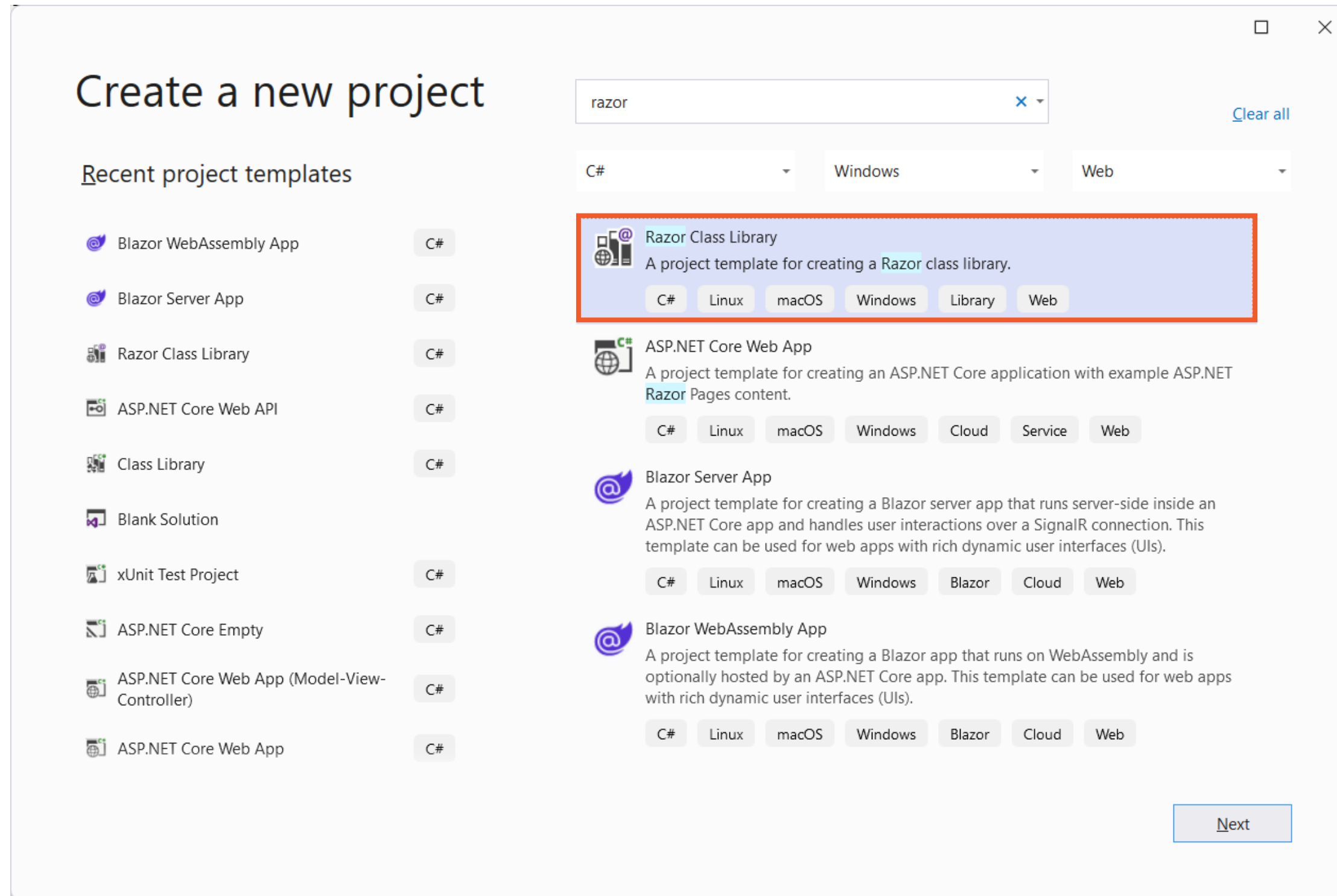


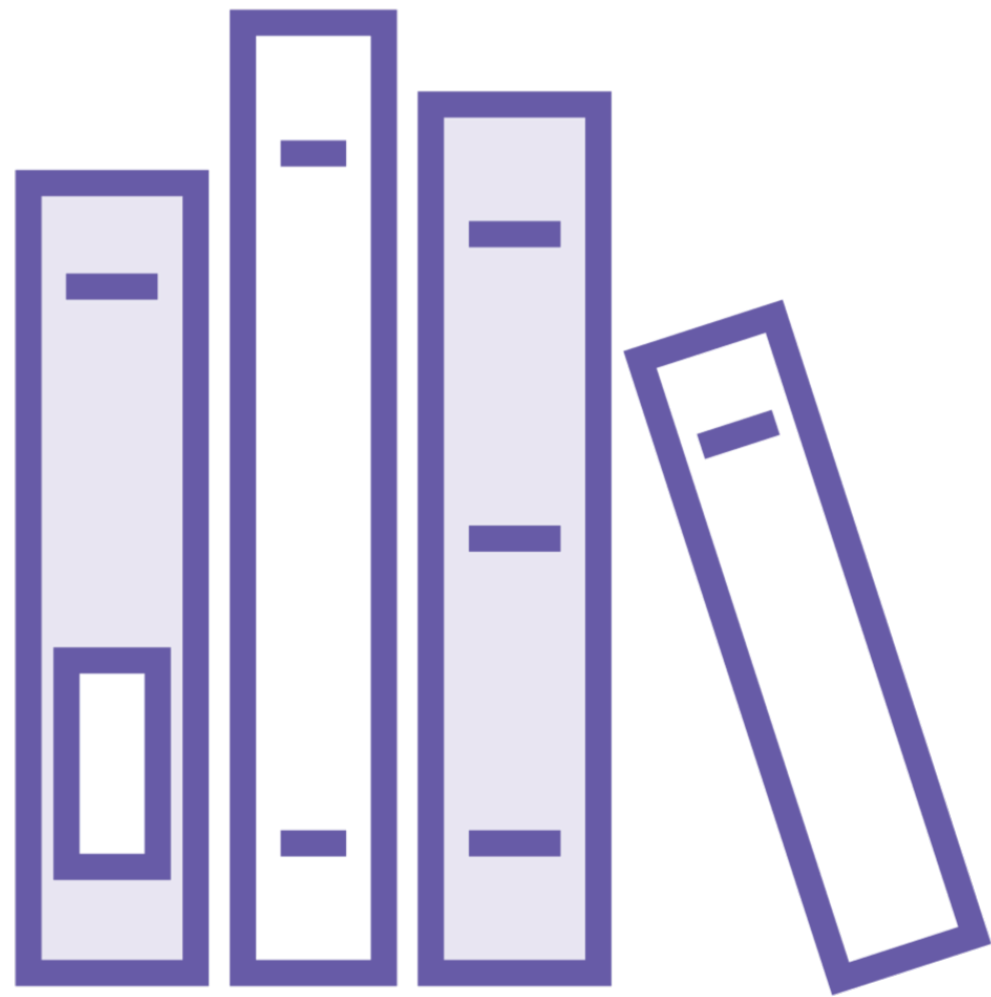
Showing a map on the details page

Map component:
<https://aka.ms/blazorworkshop>

Wrapping Components in a Razor Class Library

Introducing Razor Class Libraries





Razor Class Library project

- Can contain components and static assets
- Reusable
- NuGet package
- Referenced from main project

```
@using BethanysPieShopHRM.App.Components
```

```
<Map> . . . </Map>
```

Using a Component from an RCL

Full namespace also works

Typically @using will be placed in _Imports

```
<link
  href="_content/BethanysPieShopHRM.ComponentsLibrary/leaflet/leaflet.css"
  rel="stylesheet" />
```

Using Static Content from an RCL

Files must be placed in wwwroot folder

Demo

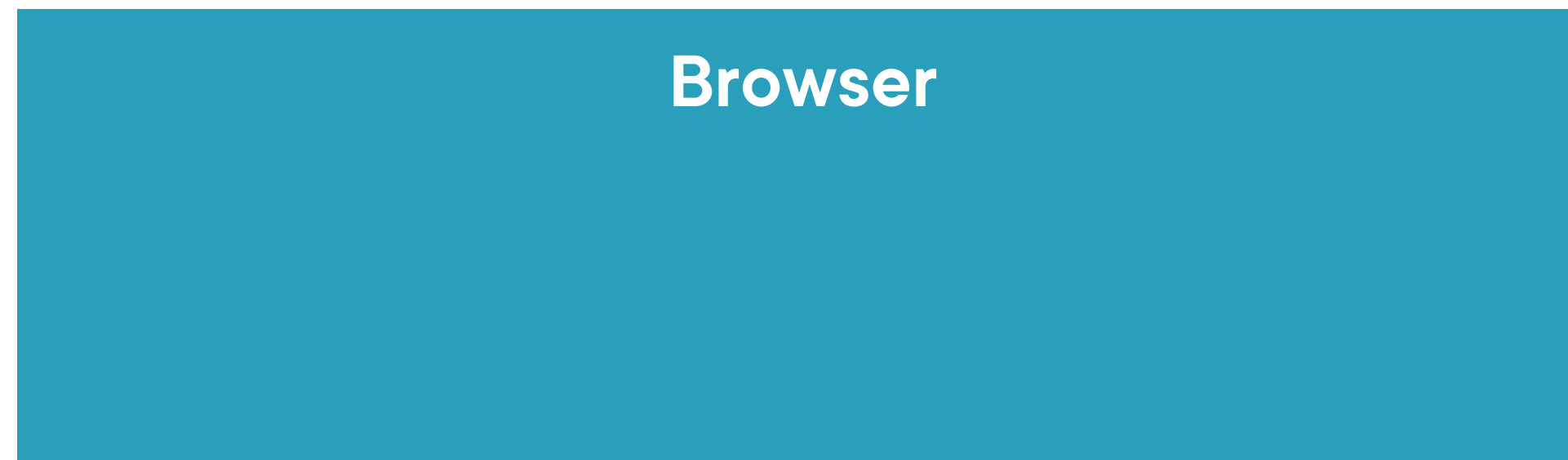
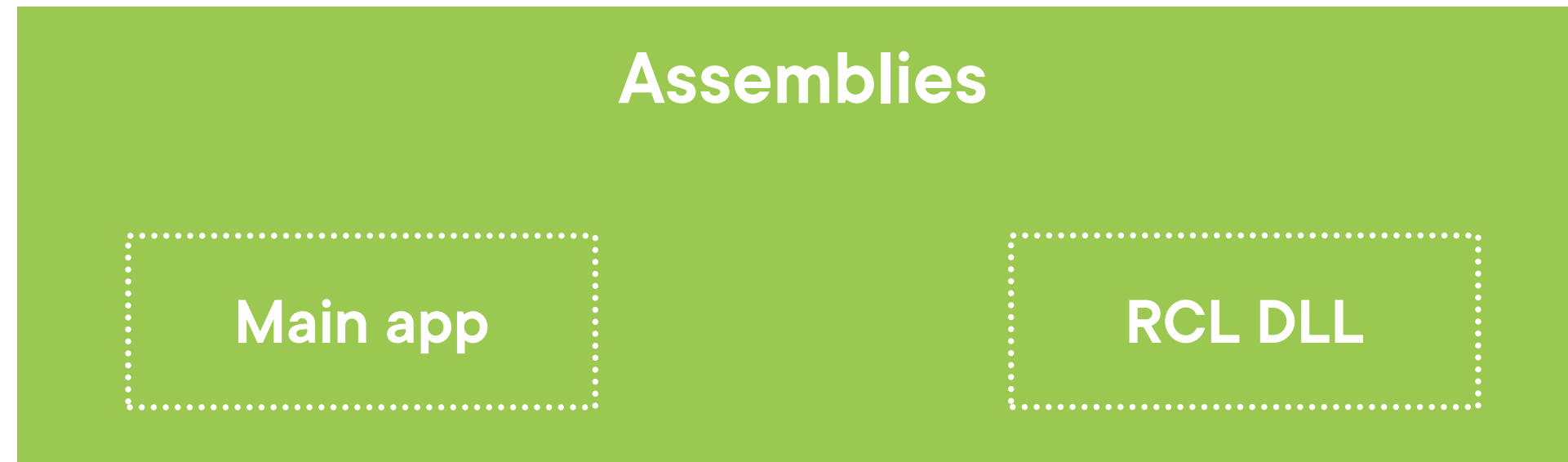


**Moving the Map component to a Razor
Class Library**

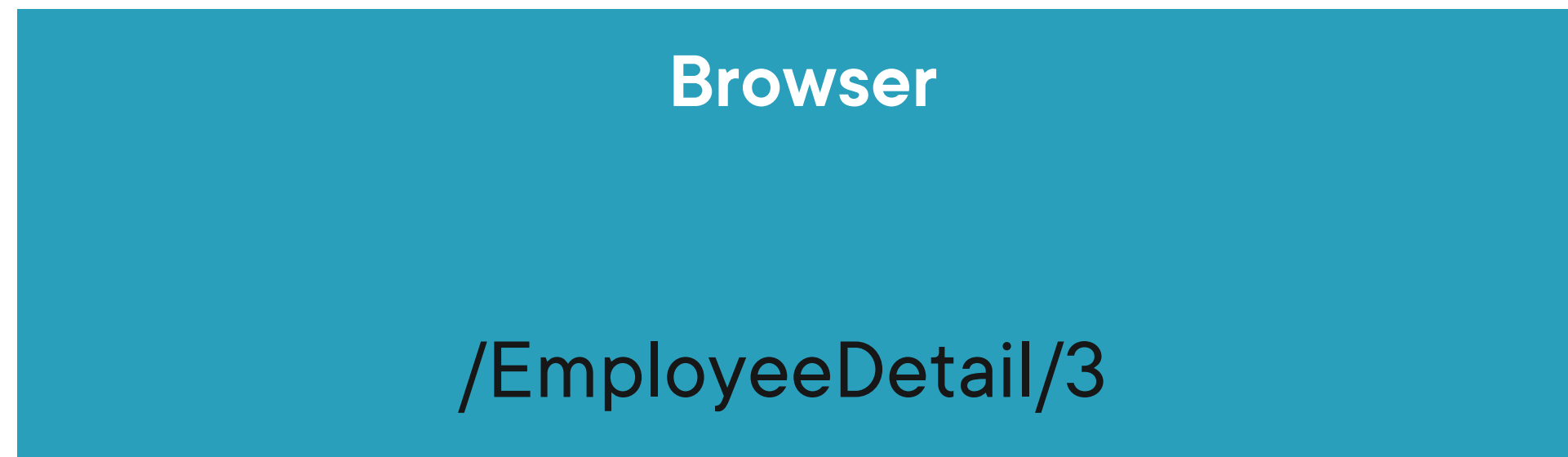
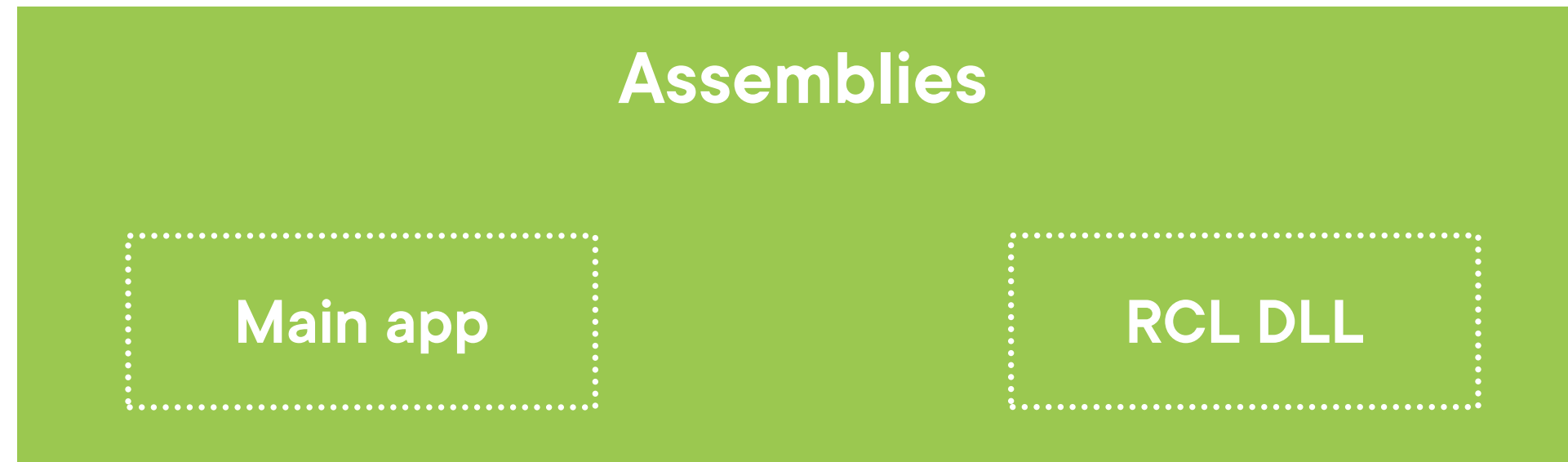
Changing the main project

Lazy Loading Libraries

Normal Loading



Introducing Lazy Loading



```
<ItemGroup>  
    <BlazorWebAssemblyLazyLoad Include="BethanysPieShopHRM.ComponentsLibrary.dll" />  
</ItemGroup>
```

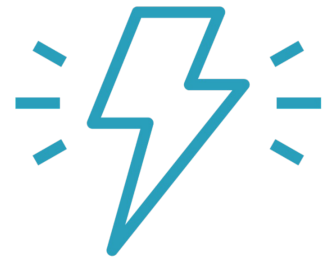
Changes to the Project File

Needs to be included for each assembly

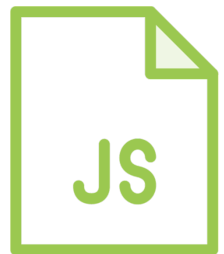
Changes to the Router



**App gets service instance injected to support lazy loading:
LazyAssemblyLoader**



Trigger is `OnNavigateAsync()`



Loads assembly via async JavaScript call



Assembly will be loaded into runtime



Can include routable components

Demo



Lazy loading the RCL

Summary



Missing functionality in Blazor can be added through JavaScript interop

Components can be placed in Razor Class Library

Lazy loading can speed up application start



Up next:
Authenticating in the application