UNIVERSITY OF WESTMINSTER

Enterprise Application Development

Lecture 5 – Web application development with Blazor

UNIVERSITY OF WESTMINSTER

Web development

- Web Development process
  - Define the problem you are solving
    - Requirements
    - Use cases
  - Build a wireframe
  - Technology choice
    - Frontend
    - Backend
    - Database
- Development environment
- Testing
  - Unit, integration and end-to-end
- Hosting and Deployment
  - Dedicated or shared hosting
  - Manual deployment or CI/CD

UNIVERSITY OF WESTMINSTER

Web site vs Web application

- Web site
  - Presentation
  - Static output
- Web application
  - Takes user input (CRUD commands)
  - Performs computation or persists data
  - Dynamic output

Technology choice

- Frontend

Structure

Look

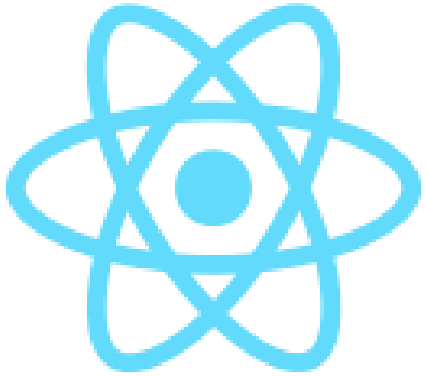
Behavior

UNIVERSITY OF WESTMINSTER

Technology choice

- Frontend frameworks

Angular



React

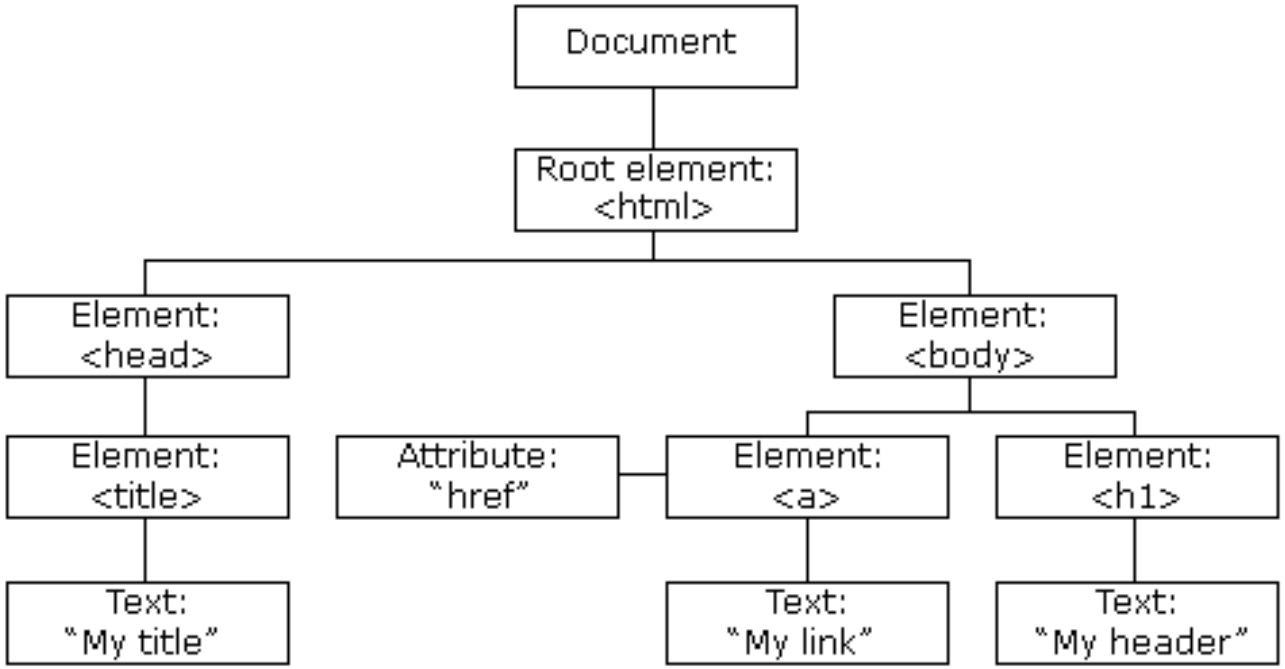
Vue

Blazor

UNIVERSITY OF WESTMINSTER

Frontend

- DOM (Document Object Model)



UNIVERSITY OF WESTMINSTER

Technology choice

- Backend

Laravel

Express.js

Django

ASP.NET

UNIVERSITY OF WESTMINSTER

Client-Server architecture

- Server contains the resources (files and data)
- Clients connect to the server to get these resources
- The communication between the client and server is typically done over HTTP
- HTTP Methods
  - GET, POST, PUT, DELETE

UNIVERSITY OF WESTMINSTER

Web development environment

- SDK
- IDE
  - Visual Studio
  - Visual Studio Code
  - JetBrains IDEs
- Tools
  - Git
    - Bitbucket, GitHub
  - Project management
    - Trello, Jira
  - Postman or cURL (test APIs)
  - Azure DevOps

Blazor

- Blazor is an open-source SPA framework for building interactive client-side web applications with .NET and C#
- Blazor currently has two hosting models, server-side Blazor and Web Assembly
- It is based on component architecture
- A component is a group of UI controls which can be reused in other pages or components

SPA - Single Page Application

- Single Page Application (SPA) is a web application that rewrites the current page with new data based on user interactions rather than loading a new page

- With this transitions between pages are faster, we only update a part of the page or just the data on the page

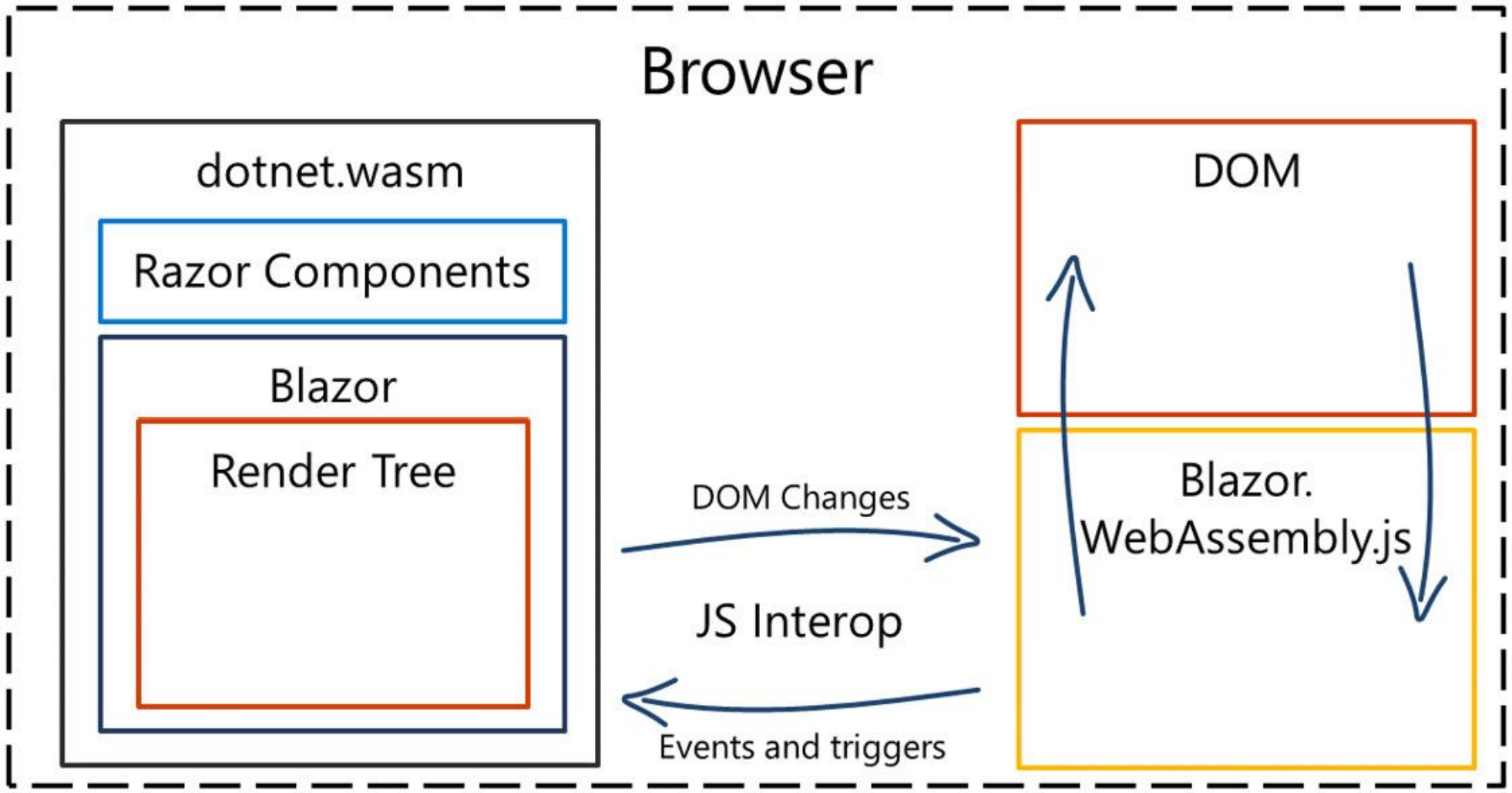
- It feels more like a native app

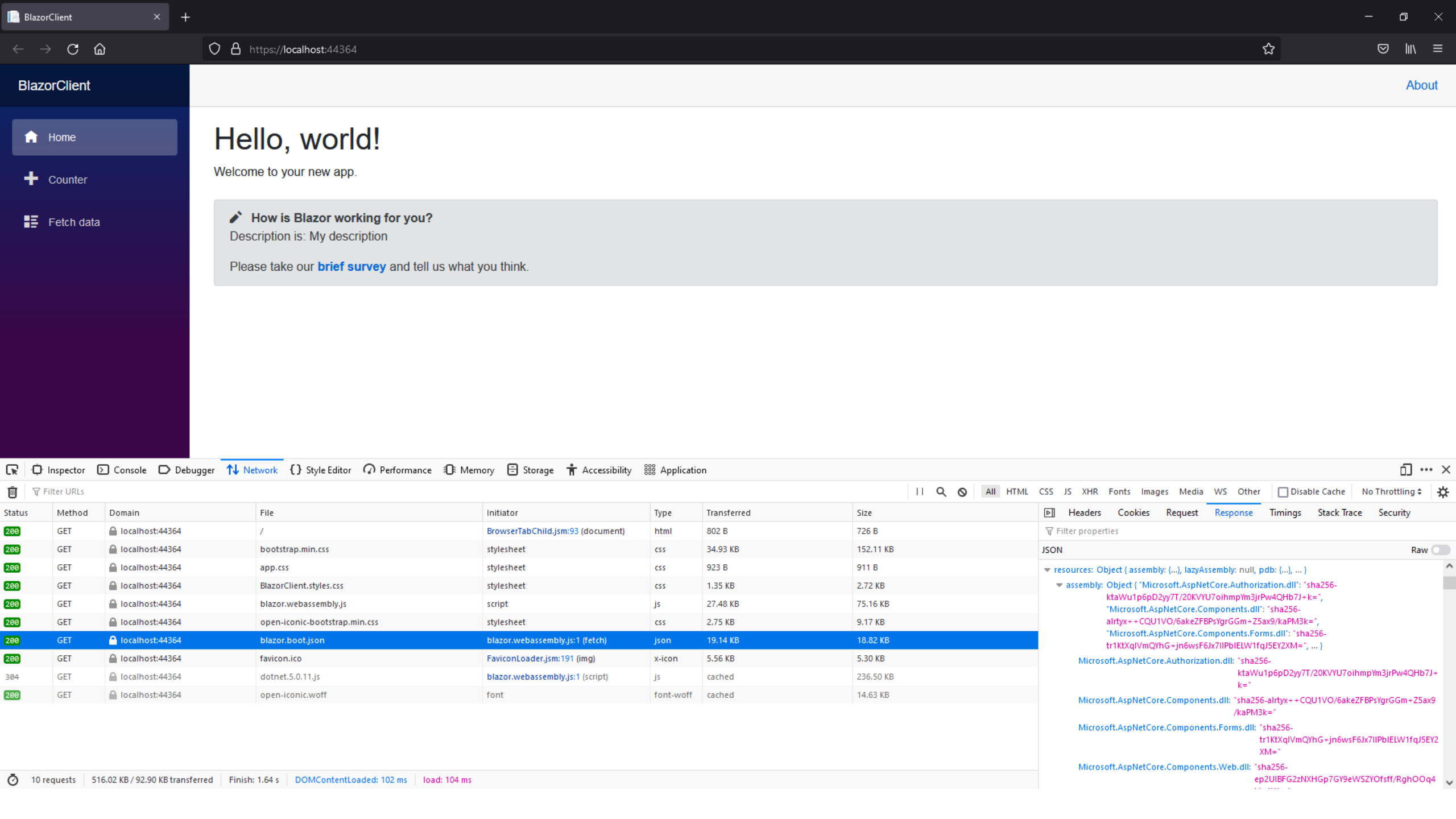
UNIVERSITY OF WESTMINSTER

PWA - Progressive Web Application

- PWA is a type of application delivered through the web and built with common web technologies (HTML, CSS & JS)
- Benefits of PWAs is that the client can choose to install the app onto their device and run then even when they have no internet access.
- Works in chromium-based browsers and Safari
  - Does not work in Firefox
- Example: https://app.starbucks.com/

Blazor Client (WebAssembly)




BlazorClient
https://localhost:44364
BlazorClient
About
Home
Counter
Fetch data
Hello, world!
Welcome to your new app.
How is Blazor working for you?
Description is: My description
Please take our brief survey and tell us what you think.
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter URLs
All HTML CSS JS XHR Fonts Images Media WS Other
Disable Cache
No Throttling
Status Method Domain File Initiator Type Transferred Size
200 GET localhost:44364 / BrowserTabChild.jsm:93 (document) html 802 B 726 B
200 GET localhost:44364 bootstrap.min.css stylesheet css 34.93 KB 152.11 KB
200 GET localhost:44364 app.css stylesheet css 923 B 911 B
200 GET localhost:44364 BlazorClient.styles.css stylesheet css 1.35 KB 2.72 KB
200 GET localhost:44364 blazor.webassembly.js script js 27.48 KB 75.16 KB
200 GET localhost:44364 open-iconic-bootstrap.min.css stylesheet css 2.75 KB 9.17 KB
200 GET localhost:44364 blazor.boot.json blazor.webassembly.js:1 (fetch) json 19.14 KB 18.82 KB
200 GET localhost:44364 favicon.ico FaviconLoader.jsm:191 (img) x-icon 5.56 KB 5.30 KB
304 GET localhost:44364 dotnet.5.0.11.js blazor.webassembly.js:1 (script) js cached 236.50 KB
200 GET localhost:44364 open-iconic.woff font font-woff cached 14.63 KB
Headers Cookies Request Response Timings Stack Trace Security
Filter properties
JSON Raw
resources: Object { assembly: {…}, lazyAssembly: null, pdb: {…}, … }
assembly: Object { "Microsoft.AspNetCore.Authorization.dll": "sha256-ktaWu1p6pD2yy7T/20KVYU7oihmpYm3jrPw4QHb7J+k=", "Microsoft.AspNetCore.Components.dll": "sha256-alrtyx++CQU1VO/6akeZFBPsYgrGGm+Z5ax9/kaPM3k=", "Microsoft.AspNetCore.Components.Forms.dll": "sha256-tr1KtXqlVmQYhG+jn6wsF6Jx7IlPbIELW1fqJ5EY2XM=", … }
Microsoft.AspNetCore.Authorization.dll: "sha256-ktaWu1p6pD2yy7T/20KVYU7oihmpYm3jrPw4QHb7J+k="
Microsoft.AspNetCore.Components.dll: "sha256-alrtyx++CQU1VO/6akeZFBPsYgrGGm+Z5ax9/kaPM3k="
Microsoft.AspNetCore.Components.Forms.dll: "sha256-tr1KtXqlVmQYhG+jn6wsF6Jx7IlPbIELW1fqJ5EY2XM="
Microsoft.AspNetCore.Components.Web.dll: "sha256-ep2UIBFG2zNXHGp7GY9eWSZYOfsff/RghOOq4
10 requests
516.02 KB / 92.90 KB transferred
Finish: 1.64 s
DOMContentLoaded: 102 ms
load: 104 ms

Blazor Client (WebAssembly)

- Advantages
  - No round trips to server which means faster UI update
  - It is not required to have a server at all
  - Write C# code on frontend (client side)
  - Since the code runs in the browser, it is easy to create Progressive Web Apps (PWA)

Blazor Client (WebAssembly)

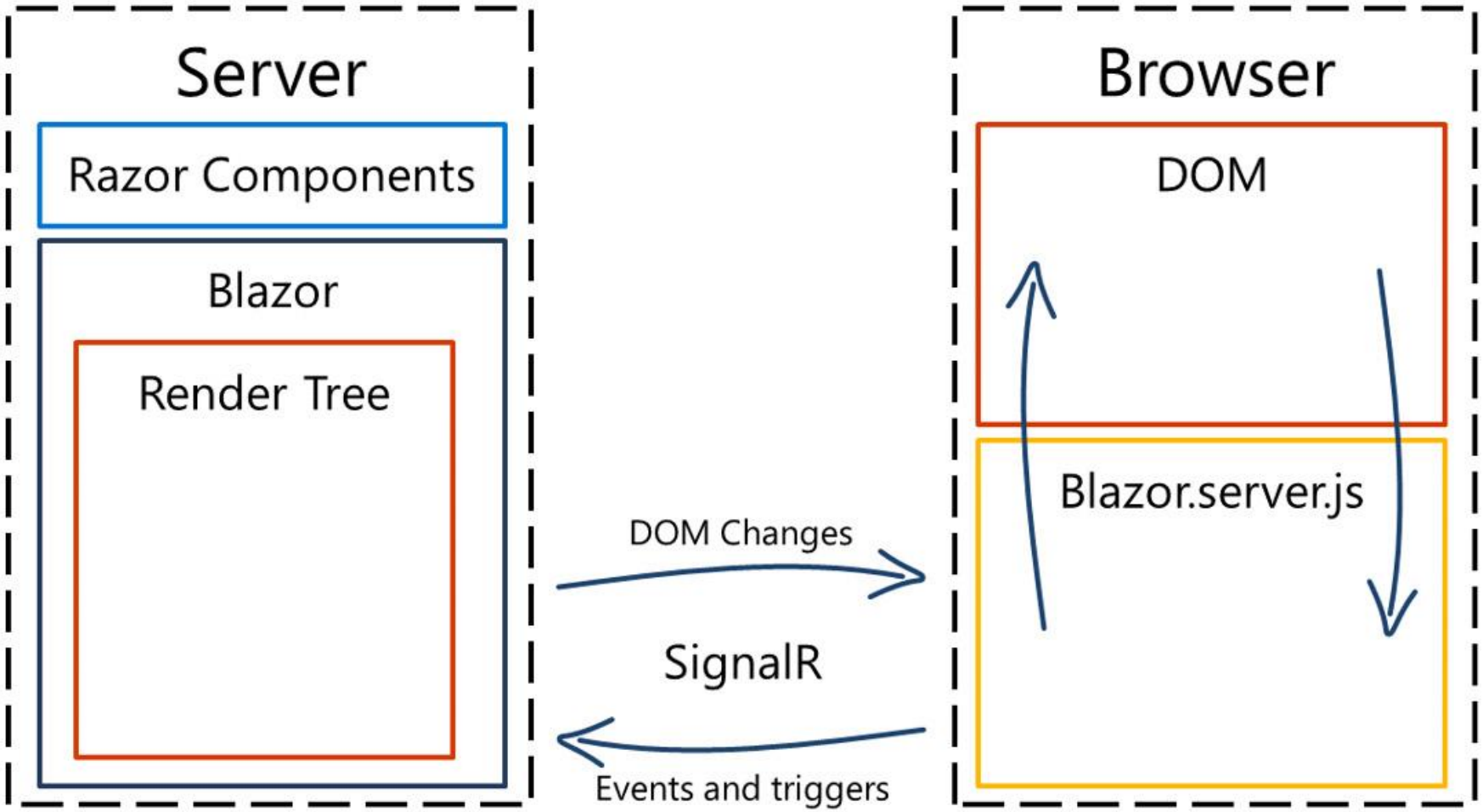
- Disadvantages
  - The footprint of a Blazor WebAssembly is large since everything is offloaded to the client
  - To access any resources, you will need an API since it is not possible to access the database directly
  - The code runs in the browser which means it can be decompiled and everyone can get access to your source code

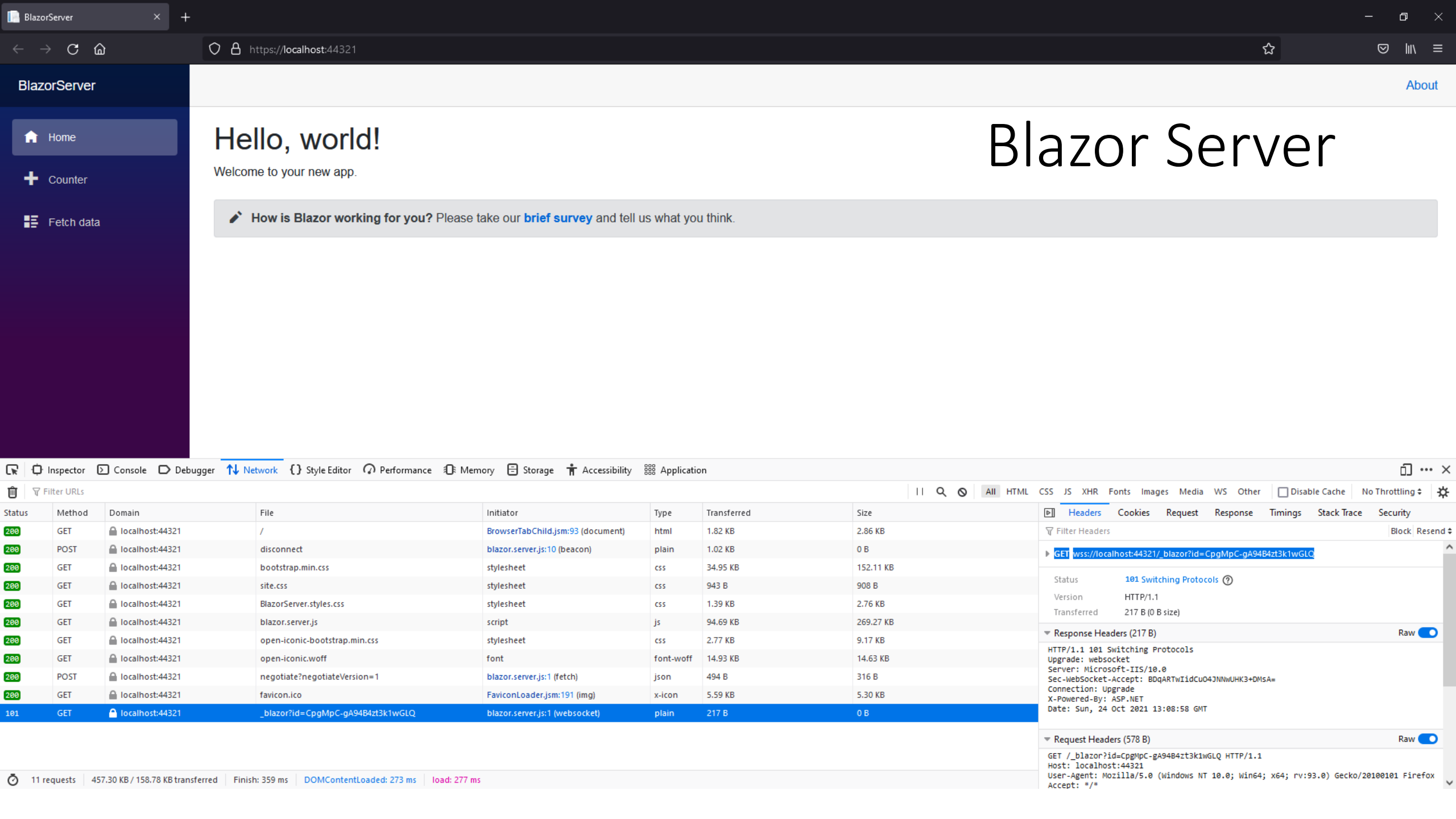
UNIVERSITY OF WESTMINSTER

Blazor Client (WebAssembly)

- The only way we can get data is by making external calls (to a server)
- We make HTTP requests using HttpClient which we get through Dependency Injection (DI)
- Code
  - @inject HttpClient
  - var forecasts = await Http.GetFromJsonAsync<WeatherForecast[]>(“WeatherForecast”);

Blazor Server




BlazorServer
https://localhost:44321
BlazorServer
About
Home
Counter
Fetch data
Hello, world!
Welcome to your new app.
Blazor Server
How is Blazor working for you? Please take our brief survey and tell us what you think.
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Filter URLs
All HTML CSS JS XHR Fonts Images Media WS Other
Disable Cache
No Throttling
Status Method Domain File Initiator Type Transferred Size
200 GET localhost:44321 / BrowserTabChild.jsm:93 (document) html 1.82 KB 2.86 KB
200 POST localhost:44321 disconnect blazor.server.js:10 (beacon) plain 1.02 KB 0 B
200 GET localhost:44321 bootstrap.min.css stylesheet css 34.95 KB 152.11 KB
200 GET localhost:44321 site.css stylesheet css 943 B 908 B
200 GET localhost:44321 BlazorServer.styles.css stylesheet css 1.39 KB 2.76 KB
200 GET localhost:44321 blazor.server.js script js 94.69 KB 269.27 KB
200 GET localhost:44321 open-iconic-bootstrap.min.css stylesheet css 2.77 KB 9.17 KB
200 GET localhost:44321 open-iconic.woff font font-woff 14.93 KB 14.63 KB
200 POST localhost:44321 negotiate?negotiateVersion=1 blazor.server.js:1 (fetch) json 494 B 316 B
200 GET localhost:44321 favicon.ico FaviconLoader.jsm:191 (img) x-icon 5.59 KB 5.30 KB
101 GET localhost:44321 _blazor?id=CpgMpC-gA94B4zt3k1wGLQ blazor.server.js:1 (websocket) plain 217 B 0 B
Headers Cookies Request Response Timings Stack Trace Security
Filter Headers
Block
Resend
GET wss://localhost:44321/_blazor?id=CpgMpC-gA94B4zt3k1wGLQ
Status 101 Switching Protocols
Version HTTP/1.1
Transferred 217 B (0 B size)
Response Headers (217 B)
Raw
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Server: Microsoft-IIS/10.0
Sec-WebSocket-Accept: BDqARTwIidCuO4JNNwUHK3+DMsA=
Connection: Upgrade
X-Powered-By: ASP.NET
Date: Sun, 24 Oct 2021 13:08:58 GMT
Request Headers (578 B)
Raw
GET /_blazor?id=CpgMpC-gA94B4zt3k1wGLQ HTTP/1.1
Host: localhost:44321
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox
Accept: */*
11 requests
457.30 KB / 158.78 KB transferred
Finish: 359 ms
DOMContentLoaded: 273 ms
load: 277 ms

Blazor Server

- Advantages
  - Faster development process as it is not necessary to build an API and it is possible to access the database directly from components.
  - Smaller footprint than Blazor Client (WebAssembly) as we do not have to download all the files
  - It will work on web browsers that do not support WebAssembly (older versions of IE)
  - Users have no access to the code (files and libraries) since it runs on the server

Blazor Server

- Disadvantages
  - **Stateful:** You need to always be connected to the server since the rendering is done on the server.
  - There is no offline or PWA mode since it needs to be connected all the time
  - Every click or page update needs to go to the server
  - Difficult to scale

Razor

- .NET HTML rendering engine
- To transition from HTML to C# code we use the @ symbol.
- These are several ways we can add code to our HTML file:
  - Razor code blocks
  - Implicit Razor expressions
  - Explicit Razor expressions

UNIVERSITY OF WESTMINSTER

Razor code blocks

```
@code {
    //your code here
}
```

- Or just

```
@{
    //your code here
}
```

UNIVERSITY OF WESTMINSTER

Implicit vs Explicit razor expressions

- Implicit
  - <td>@forecast.Summary</td>
- Explicit
  - <td>@(forecast.TemperatureC)</td>
- With implicit expressions we are not be able to call generic methods
  - <td>@(MyGenericMethod<string>())</td>

UNIVERSITY OF WESTMINSTER

Razor

- Directives
  - Adding an attribute
    - @attribute [Authorize]
  - Adding an interface
    - @implements Idisposable
  - Inheriting
    - @inherits TypeNameOfClassToInheritFrom
  - Generic components
    - @typeparam Titem
- Changing the layout
  - @layout AnotherLayoutFile
- Setting a namespace
  - @namespace Another.NameSpace
- Setting a route
  - @page "/theurl“
- Adding a using statement
  - @using System.IO

UNIVERSITY OF WESTMINSTER

Blazor – Structure code

- In the Razor file
- In a partial class (code-behind)
- Inheriting a class
- Only code

Blazor – Lifecycle events

- Most lifecycle events have two versions – synchronous and asynchronous
- OnInitialized and OnInitializedAsync
  - Called when the component is fully loaded but the UI is not rendered yet
- OnParametersSet and OnParametersSetAsync
  - Called when the parameters passed to the component change
- OnAfterRender and OnAfterRenderAsync
  - If we want to call any JavaScript code, we have to do that from these methods
  - We will get an error if we try to make a JavaScript interop from any of the other lifecycle event methods

UNIVERSITY OF WESTMINSTER

Blazor – Parameters

```
@code {
    [Parameter]
    public string MyParameter { get; set; }
}
```

- @page "/parameterdemo/{MyParameter}“
- <MyComponent MyParameter="@MyValue"></ MyComponent >

UNIVERSITY OF WESTMINSTER

Blazor – Data Binding

- With binding, UI automatically updates when the property changes
- One-way binding
  - The data moves in one direction
  - Example: Button -> Increment **(Variable update)** -> Label
- Two-way binding
  - The data moves both ways
  - Example: Button -> Increment **(Variable update)** -> Input field -> **Variable update**
  - Example: Pass a value from a parent to child component and then update the value in child component will make the parent component update as well

UNIVERSITY OF WESTMINSTER

Blazor – Forms & Input Validation

- Blazor provides UI controls that build on top of HTML and provide more functionalities
- Think of these as HTML wrappers with additional features
- Examples
  - InputText produces <input type="text">
  - InputRadio produces <input type="radio">
- With Blazor UI controls we get validation and formatting out-of-the-box

UNIVERSITY OF WESTMINSTER

Blazor – IdentityServer

- For authentication and authorization, we can use Identity Server which provides us with these functionalities
- We get a logic for login and registration that we can build on top and extend to support our use case
- When we want to restrict access on some pages we need to add @attribute [Authorize] and this will make sure the users are logged in and authorized before accessing the page

Blazor – Code Sharing

- When we use Visual Studio WebAssembly template for creating a Blazor application and tick the box for server ASP.NET project, Visual Studio will create another project called Shared

- In this shared project we can keep the classes that will be used in both client and server projects

- The classes that we will most likely share between client and server projects are models and with this approach we can reuse that code

UNIVERSITY OF WESTMINSTER

Blazor – JavaScript Interop

- Blazor uses JavaScript to update the Document Object Model (DOM), download files, and access things such as local storage on the client.

- Calling JavaScript from .NET can be done in two ways
  - Global JavaScript
  - JavaScript Isolation

- Calling .NET from JavaScript
  - Static .NET method call
  - Instance method call
  - Component instance method call

UNIVERSITY OF WESTMINSTER

Blazor – Managing State

- Storing data on the server side
  - We need an API that we can call to save/retrieve data into/from a database
- Passing data in URLs
  - Route parameters
    - @page "/post/{BlogPostId:int?}“
  - Using a query string
    - var query = new Uri(Navigation.Uri).Query;
- Implement browser storage

UNIVERSITY OF WESTMINSTER

Blazor – Deployment

- Hosting Blazor WebAssembly
  - Server application needs to be hosted on a server that supports .NET
  - Client application can be hosted on the same or separate server as the server application or it can be hosted on CDN (Content Delivery Network)

- Hosting Blazor Server (SignalR)
  - Same as the server application from Blazor WebAssembly
  - On top of that we might need Azure SignalR Service that will take care of all the connections

Resources

- Documentation
  - https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor
  - https://blazor-university.com/overview/what-is-blazor/
  - https://docs.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-3.1

- Books
  - Web Development with Blazor: A hand-on guide for .NET developers to build interactive UIs with C#, Jimmy Engstroem