Blazor Conf 2022

Blazor ♥ JavaScript

```
{
    "name"      : "Andrea Dottor",
    "mail"      : "andrea@dottor.net",
    "twitter"   : "@dottor"
}
```

Un grazie agli sponsor

E alle community che ci hanno supportato

"*Interactive web UI with C# instead of JavaScript*"

fonte: https://dotnet.microsoft.com/en-us/apps/aspnet/web-apps/blazor

'Death to JavaScript!' Blazor, for .NET Web Apps Using WebAssembly, Goes Alpha

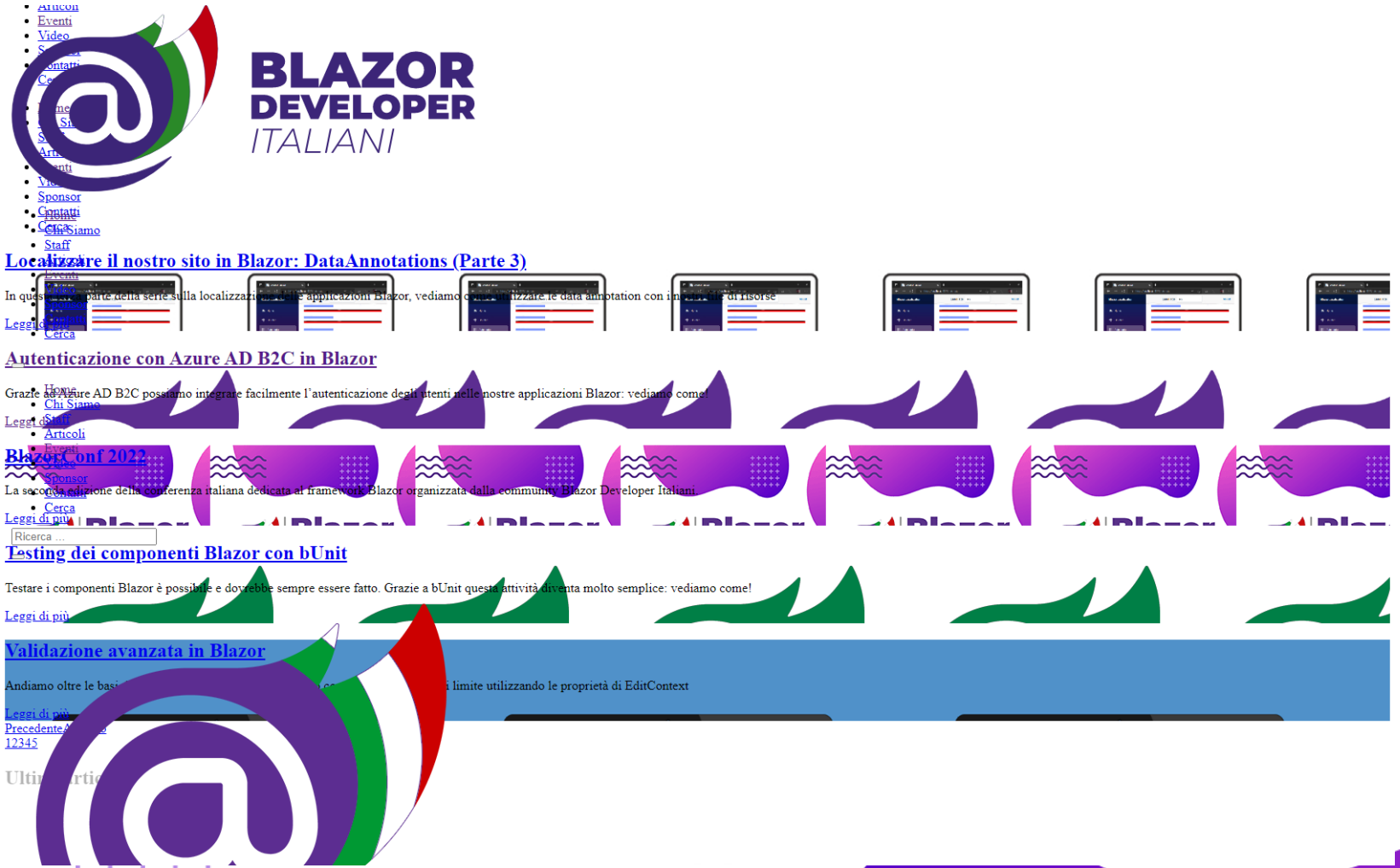
Blazor è un innovativo framework di Microsoft che permette di creare interfacce utenti semplici e coinvolgenti senza più utilizzare JavaScript, ma programmando in C#. Permette quindi a uno sviluppatore di riutilizzare nelle proprie web app logiche di business già scritte in quell'ambiente di sviluppo.

Will Blazor replace JavaScript?

Blazor Web Apps - Goodbye JavaScript! I'm in love with C#

Blazor lets you build interactive web UIs using C# instead of JavaScript. Blazor apps are composed of reusable web UI components implemented using C#, HTML, and CSS. Both client and server code is written in C#, allowing you to share code and libraries.

…come scrivere HTML senza CSS



Vantaggi nell'utilizzare JavaScript con Blazor

Uso di librerie JavaScript esistenti

Utilizzo di API del browser non direttamente supportate in Blazor

Riuso di codice JavaScript (da app già scritte)

Ultimo (ma non l'ultimo) dei vantaggi

Riduzione del numero di componenti di terze parti di cui si ha realmente bisogno

Da dove iniziare

Chiamare funzioni JavaScript da Blazor

Tramite dependency injection possiamo farci passare **IJSRuntime** ed utilizzare i metodi

- IJSRuntime.InvokeAsync
- JSRuntimeExtensions.InvokeAsync
- JSRuntimeExtensions.InvokeVoidAsync

JavaScript isolation, IJSObjectReference

Da .NET 5 possiamo sfruttare la **JavaScript Isolation** per evitare di avere tutte le funzioni JavaScript visibili a livello globale

- JavaScript modules (ECMAScript specification).
- Ogni componente può fare l'import di un proprio file JavaScript e richiamarne le funzioni

```
protected override async Task OnAfterRenderAsync(bool firstRender)
{
    if (firstRender)
    {
        module = await JS.InvokeAsync<IJSObjectReference>("import", "./_content/Dottor.Umarell.UI/js/SBAdmin.js");
        await module.InvokeVoidAsync("appInit");
    }
}
```

ElementReference

Con **ElementReference** si ha la possibilità di passare ad una funzione JavaScript un riferimento ad un elemento del DOM.

- Questi elementi è consigliato siano vuoti ed il contenuto venga modificato solo da JavaScript (e non anche da codice C#)

```
<canvas @ref="_chartElement"></canvas>
```

```
private ElementReference _chartElement;
```

```
await _mapModule.InvokeVoidAsync("draw", _chartElement, data, labels);
```

Demo

- Getting started
- Chart.js
- Bootstrap

Chiamare funzioni .NET da JavaScript

Metodi statici: si utilizza **DotNet.invokeMethod** o **DotNet.invokeMethodAsync**:

- I metodi devono essere marcati come public e static ed essere decorate con l'attributo [JSInvokable].
- DotNet.invokeMethodAsync('{ASSEMBLY NAME}', '{.NET METHOD ID}', {ARGUMENTS});

Metodi di istanza:

- Passare alla funzione JavaScript l'istanza dell'oggetto utilizzando **DotNetObjectReference**.
- Invocare il metodo .NET da JavaScript utilizzando invokeMethod o invokeMethodAsync dall'oggetto DotNetObjectReference passato alla funzione.
- Ricordarsi di fare il dispose di DotNetObjectReference.

Demo

- Google Maps

Passare uno stream a JavaScript

Utilizzando **DotNetStreamReference** è possibile passare direttamente uno stram da .NET a JavaScript

- Possibilità di mantenerlo aperto valorizzando leaveOpen a true.

```
@code {
    private async Task DownloadFileFromStream()
    {
        var fileStream = GetFileStream();
        var fileName = "log.bin";

        using var streamRef = new DotNetStreamReference(stream: fileStream);

        await JS.InvokeVoidAsync("downloadFileFromStream", fileName, streamRef);
    }
}
```

Demo

QuestPDF + download

Performance

IJSInProcessRuntime

Permette di invocare metodi JavaScript in modo sincrono.

- 4 volte più veloce rispetto ad InvokeAsync **

Utilizzabile solo da Blazor WASM

IJSInProcessObjectReference rappresenta il riferimento da utilizzare (dopo un import) per invocare i metodi in modo sincrono.

Unmarshalled JavaScript interop

Utilizzando **IJSUnmarshalledRuntime** o **IJSUnmarshalledObjectReference** è possibile invocare una funzione JavaScript evitando la fase di serializzazione e deserializzazione dei dati

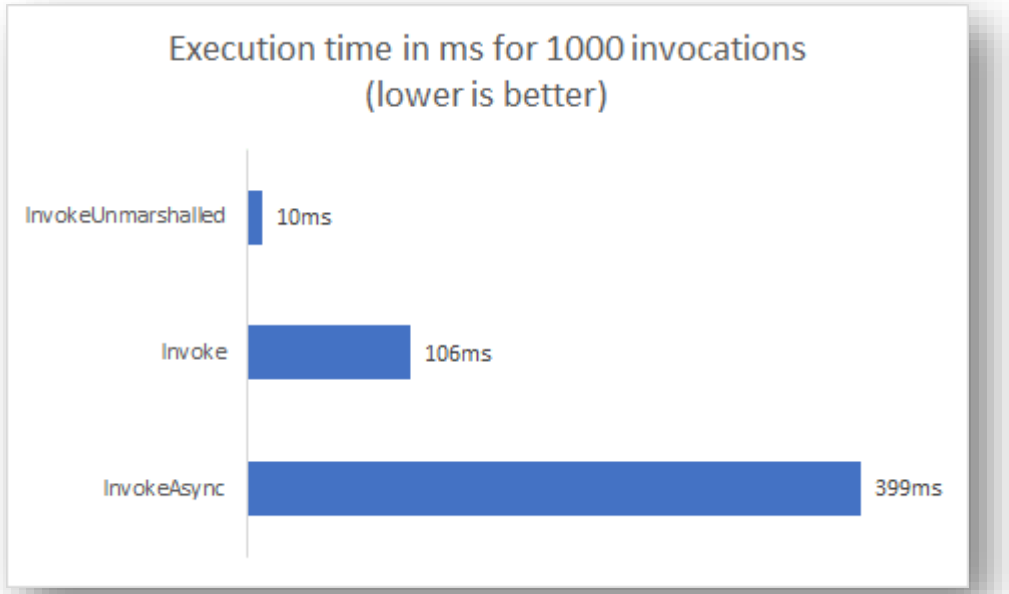
Utilizzabile solo da Blazor WASM

IJSUnmarshalledObjectReference

```
private void CallJSUnmarshalledForString()
{
    var unmarshalledRuntime = (IJSUnmarshalledRuntime)JS;

    var jsUnmarshalledReference = unmarshalledRuntime
        .InvokeUnmarshalled<IJSUnmarshalledObjectReference>(
            "returnObjectReference");

    callResultForString =
        jsUnmarshalledReference.InvokeUnmarshalled<InteropStruct, string>(
            "unmarshalledFunctionReturnString", GetStruct());
}

private InteropStruct GetStruct()
{
    return new InteropStruct
    {
        Name = "Brigadier Alistair Gordon Lethbridge-Stewart",
        Year = 1968,
    };
}

[StructLayout(LayoutKind.Explicit)]
public struct InteropStruct
{
    [FieldOffset(0)]
    public string Name;

    [FieldOffset(8)]
    public int Year;
}
```

```
<script>
  window.returnObjectReference = () => {
    return {
      unmarshalledFunctionReturnBoolean: function (fields) {
        const name = Blazor.platform.readStringField(fields, 0);
        const year = Blazor.platform.readInt32Field(fields, 8);

        return name === "Brigadier Alistair Gordon Lethbridge-Stewart" &&
            year === 1968;
      },
      unmarshalledFunctionReturnString: function (fields) {
        const name = Blazor.platform.readStringField(fields, 0);
        const year = Blazor.platform.readInt32Field(fields, 8);

        return BINDING.js_string_to_mono_string(`Hello, ${name} (${year})!`);
      }
    };
  }
</script>
```

Performance



fonte: https://www.meziantou.net/optimizing-js-interop-in-a-blazor-webassembly-application.htm

Domande?

Grazie!

Chi sono:

Andrea Dottor

Senior Developer & Consultant

andrea dottor
consulenza e sviluppo software

www.dottor.net

andrea@dottor.net

@dottor

.NET IN PILLOLE
il podcast

MVP Microsoft® Most Valuable Professional