
UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO

Dipartimento di Scienze Pure e Applicate
Corso di Laurea in Informatica Applicata

---

Tesi di Laurea

# IL FRAMEWORK BLAZOR

Relatore:
Chiar.mo Prof. Emanuele Lattanzi

Candidato:
Francesco Belacca

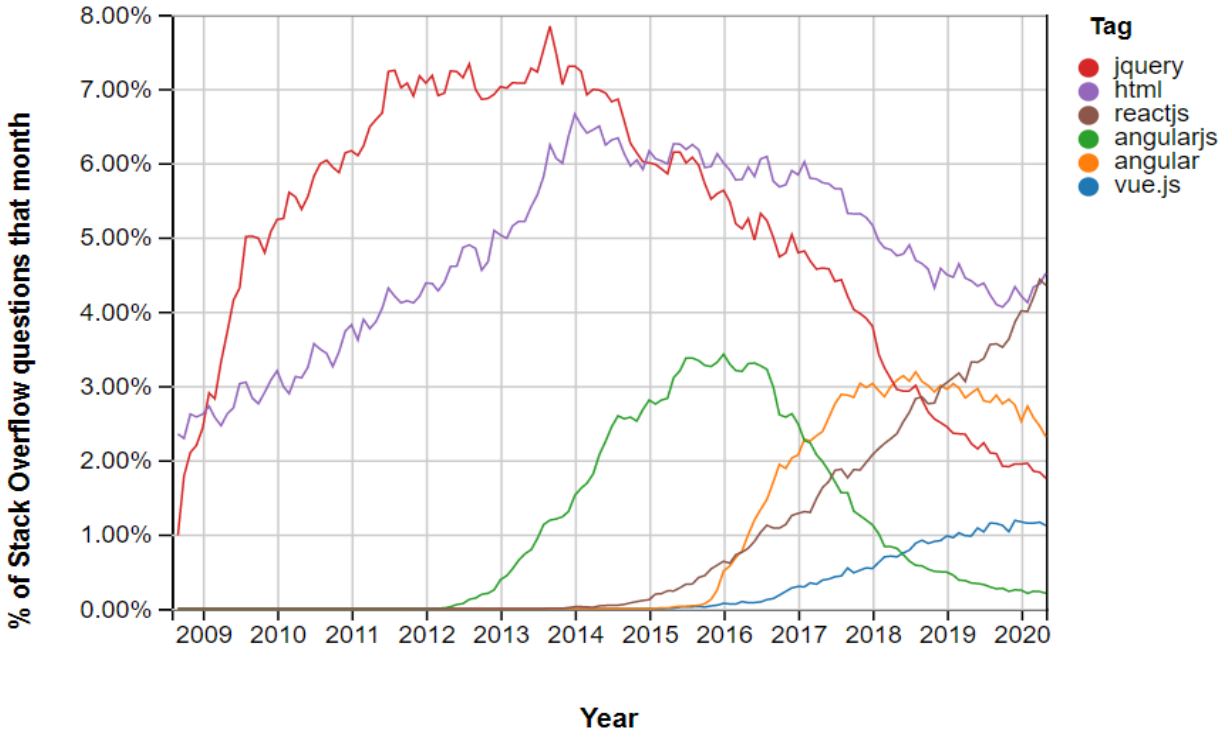
---

Anno Accademico 2019-2020

# Indice

# Contesto

I framework e le tecnologie
nel FrontEnd



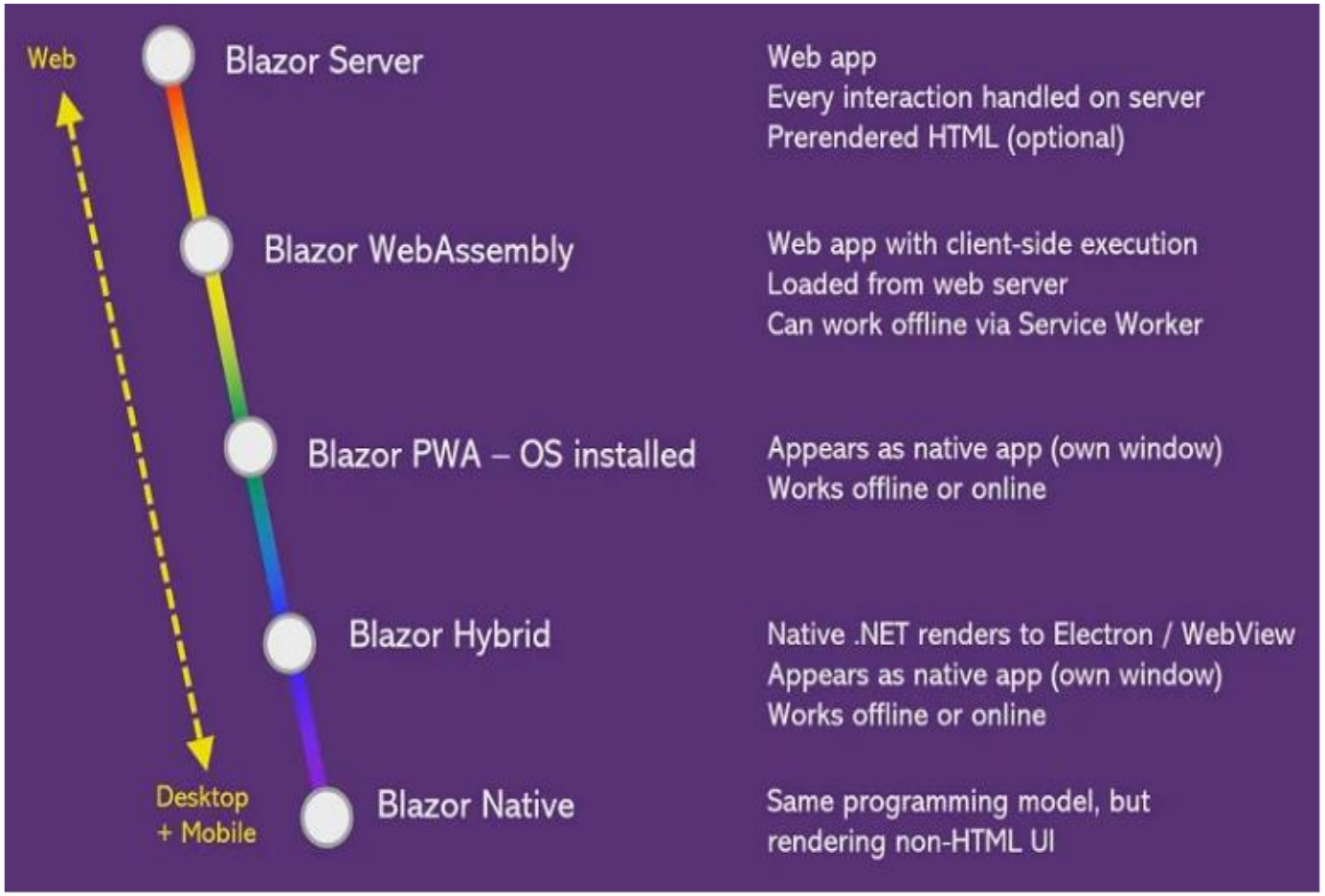

Si adattano ai trend, promettendo:

1. Produttività
2. Riutilizzo di codice

Costanti nel tempo:
HTML, CSS, Javascript(TypeScript,CoffeScript,Dart,Scala, …), Component Model

Riutilizzare skills e tecnologie diventa sempre più imporante

# Il Framework Blazor




- Razor + Browser
- C# per eventi client-side
- Nuget Components
- JavaScript Interoperability

# Blazor Server

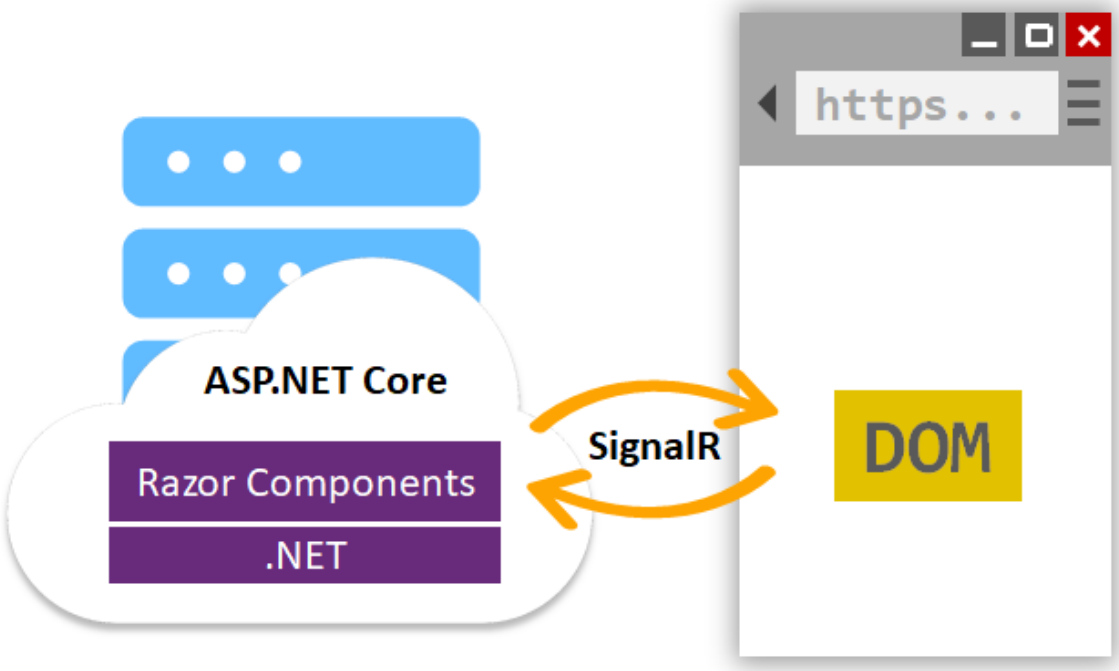
Componenti sul Server

User Interface in memoria per ogni sessione

Update Server → Client del solo delta

Peso iniziale indipendente dalla Single Page Application

Gestione eventi(click, deag, …) tramite una connessione SinglaR



# WebAssembly

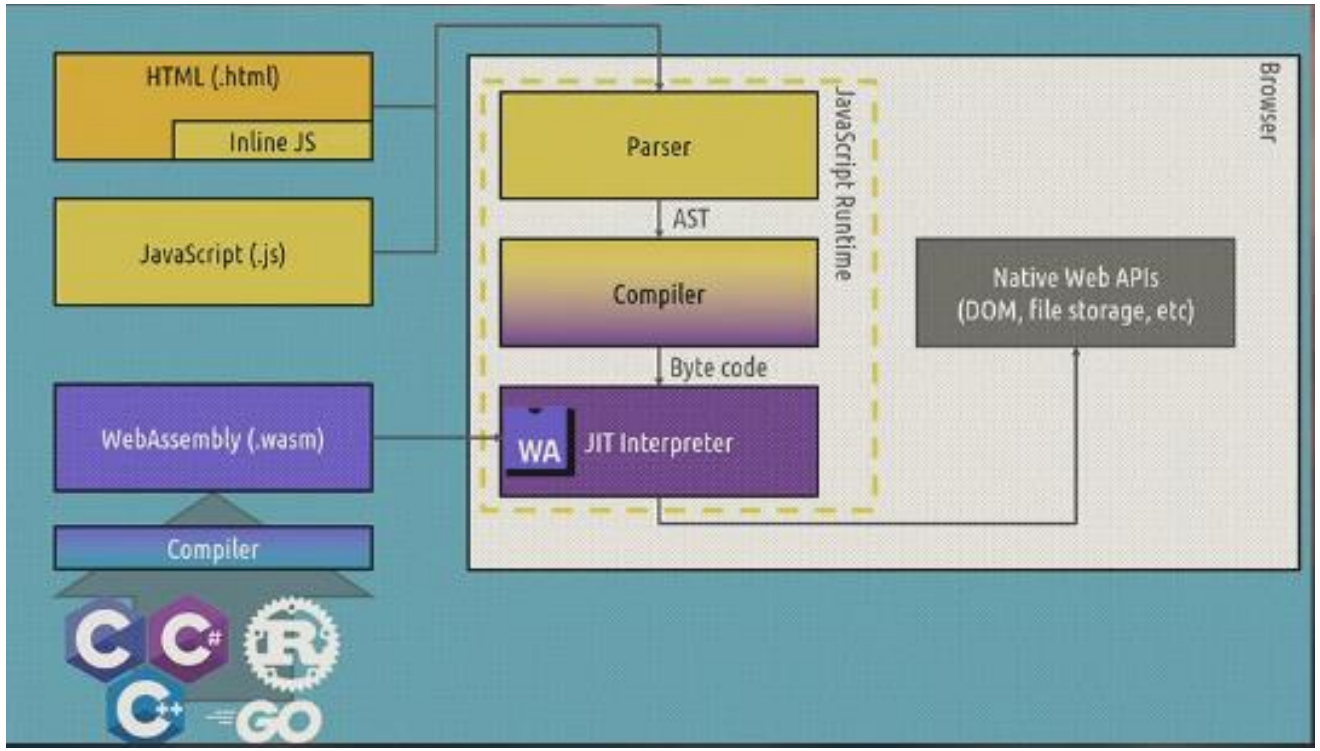
Potenzialmente più veloce di JavaScript perché compilato Ahead Of Time

Creato come target di compilazione di altri sorgenti

Byte code del web

Stessa Sandbox di JavaScript – il Browser

Supportato dal 91.66% dei browser installati



# Blazor
# WebAssembly

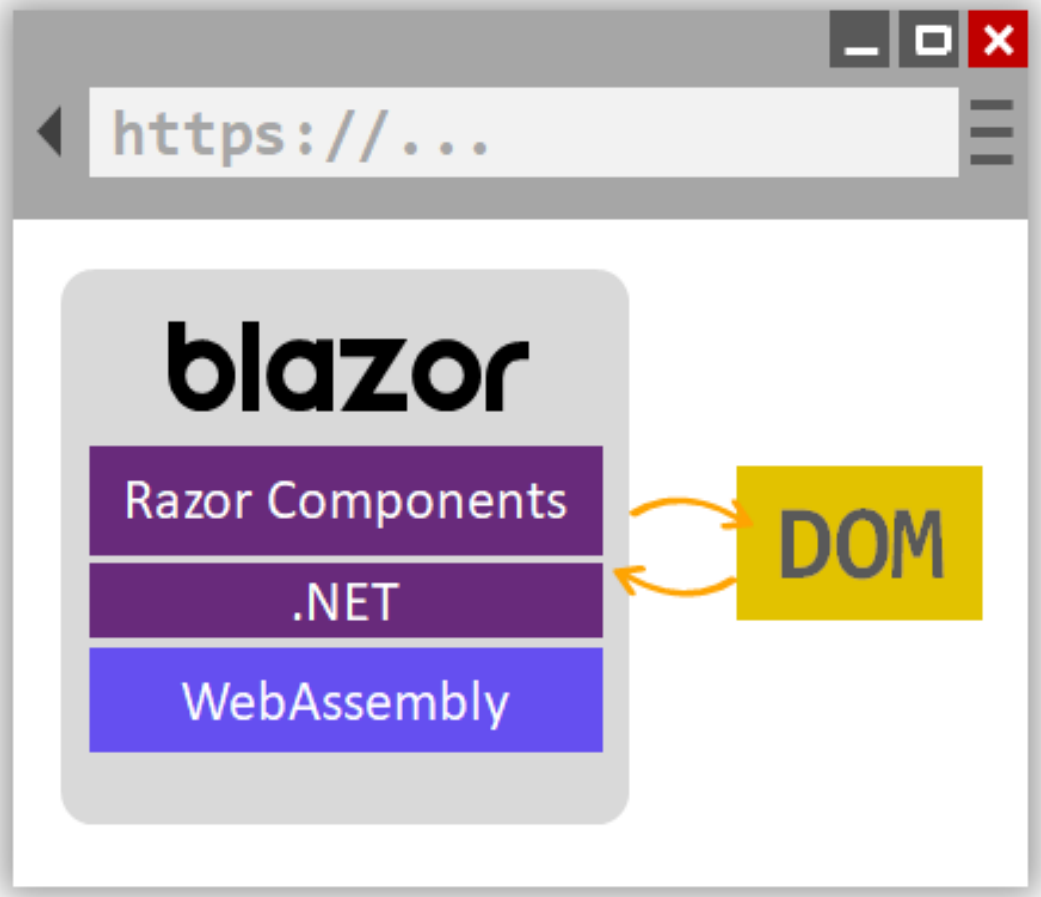
Componenti scaricati lato Client

Mono (Common Language Runtime) nel file dotnet.wasm

Dynamic Link Libraries interpretate ed eseguite lato Client

Funzionamento simile a Angular, React, Vue

Possibilità di funzionamento offline e Progressive Web App



# Sviluppare con Blazor

```razor
<div>
    <FancySpan IsBold="@(currentCount % 2 == 0)">So exciting</FancySpan>
    <p>Current count: @currentCount</p>
</div>

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

@code {
    private int currentCount = 0;

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

Razor Compiler

```
__builder.OpenElement(1, "div");
__builder.AddMarkupContent(2, "\r\n    ");
__builder.OpenComponent<code_snippets.Shared.FancySpan>(3);
__builder.AddAttribute(4, "IsBold", currentCount % 2 == 0);
__builder.AddAttribute(5, "ChildContent", (Microsoft.AspNetCore.Components.RenderFragment)((__builder2) => {
    __builder2.AddContent(6, "So exciting");
}
));
__builder.CloseComponent();
__builder.AddMarkupContent(7, "\r\n    ");
__builder.OpenElement(8, "p");
__builder.AddContent(9, "Current count: ");
__builder.AddContent(10, currentCount);
__builder.CloseElement();
__builder.AddMarkupContent(11, "\r\n");
__builder.CloseElement();
```

- HTML CSS e C#
- Il carattere di escape è @
- Estensione .razor
- Possibilità di codebehind

Roslyn

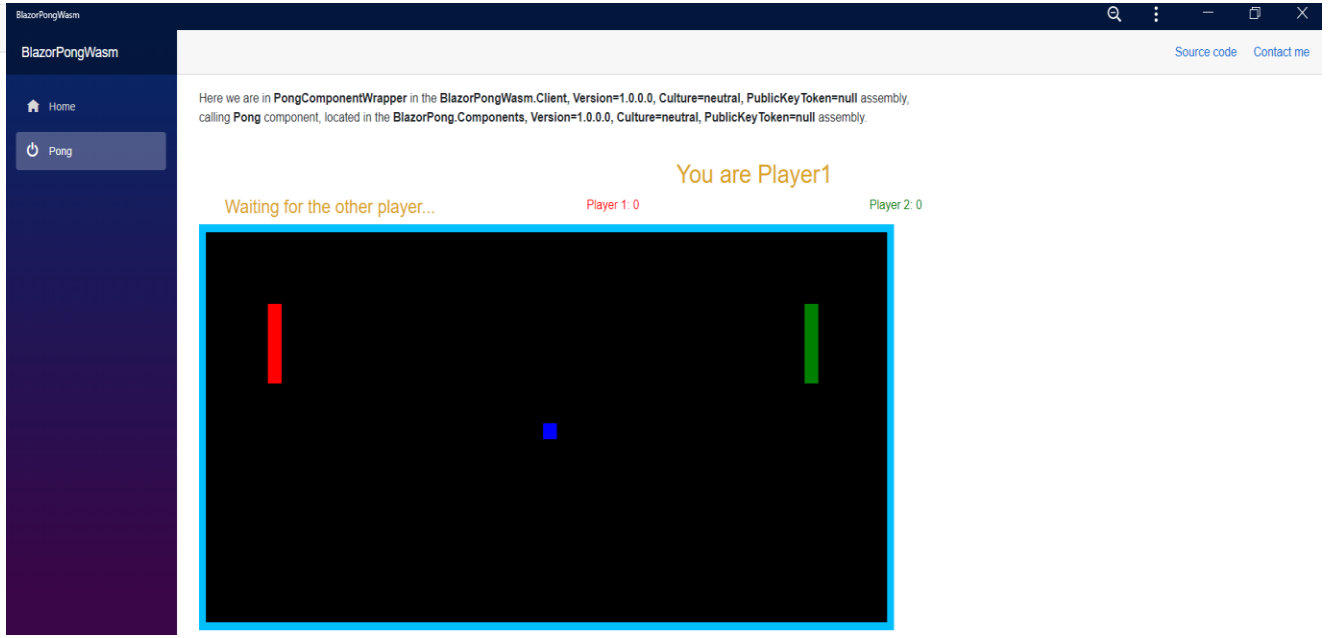

- C# valido
- Ottimizzato per il calcolo di differenze
- Estensione .cs
- Componenti ridotti a primitive

# Primitive dei componenti


- Div
- File .razor
- Attributo HTML o Parametro
- Testo
- Event handlers e bindings(2-way)

# BlazorPong - Shared

- Background worker lato Server per pallina e punteggio
- SignalR Hub lato Server per lo spostamento delle racchette
- Gestione delle sessioni lato Server: P_1, P_2, Spectator_1…Spectator_n
- Component Pong → Necessario per giocare in entrambi i modelli

# Confronto stato attuale

## Server

- Prestazioni uguali tra Clients
- Avvio immediato
- Indipendente dal peso effettivo
- Il client non riceve la Business Logic
- 175KB
- UI Rendering lato Server
- No offline, ping-dependent

## WASM

- UI Rendering lato Client
- Progressive Web Aapp
- Ideale per siti statici
- Molto pesante al primo avvio
- Download DLLs + dotnet.wasm
- Prestazioni differenti tra Clients
- 20MB

# Conclusioni

- Per BlazorPong il modello migliore è Blazor Server
- Diversi modelli per esigenze diverse
- Framework non ancora maturo, mancanze e possibilità di Breaking Changes
- Blazor WebAssembly incompleto(Ahead Of Time del solo framework)
- JavaScript ancora necessario(anche solo per BlazorPong)
- Librerie JavaScript utilizzabili(Interop), ma versione .NET spesso Mancante
- Produttività aumenta
- Riutilizzo di codice **estremo**(JS Interop, Nuget, codice Client-Server)

# Inspired by:

# Grazie mille per l’attenzione