C# で SPA を作る
Blazor WebAssembly の進化

そしてその先へ

OSC2022 Online/Hokkaido – E 10:00AM

おことわり

- 本セッションの録画を後日公開の予定です。
- 本セッションでとりあげる内容は、正確であるように努めたく思いますが、とりわけ自分があまり明るくない分野の内容については、当方の認識誤りなど多々あるかもしれません。
- もしお気づきの点などございましたら Twitter のハッシュタグ #osc22do_e10 でお知らせ頂く等していただけますと幸いです。
- 同じ .NET 上で稼働する言語・処理系として、Visual Basic および F# もある一方、本セッションでは時間の都合上、明示的に言及していません。”C#”の箇所を適宜 F# や VB に読み替えていただければと思います。


TWITTER HASH TAG
#osc22do
#osc22do_e10

セッション概要

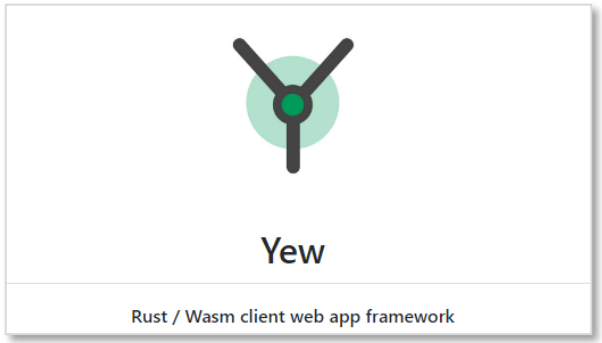
- JavaScript 以外の言語で実装する
  クライアント Web
- C# で SPA を書く Blazor が歩んできた道
- .NET6 から使える
  “Native Dependency“ とは
- Blazor WebAssembly を超えたその先

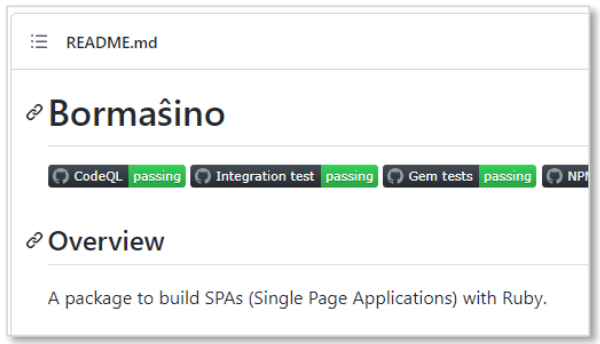
JAVASCRIPT 以外の言語で実装するクライアント WEB

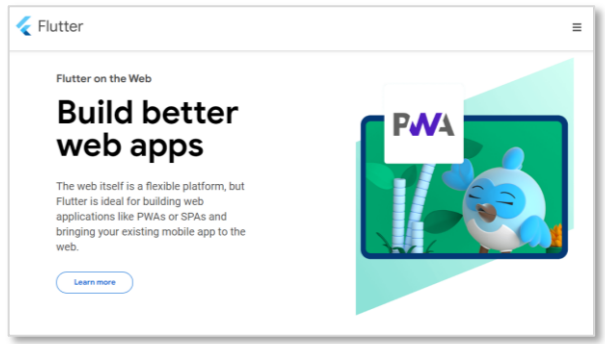
JAVASCRIPT 以外の言語で実装するクライアント WEB

- 今日、JavaScript 以外の言語で実装したプログラムが、ブラウザで動くのは**珍しくない**
  - インタープリター / WebAssembly にコンパイル / JavaScript にコンパイル
- 必ずしも React/Vue/Angular のような SPA ライブラリと同じ構造ではない
  - コンポーネント指向とは限らない
- ブラウザ上ではなく**サーバー上で動作**し、ブラウザとはリアルタイム通信することで、イベントの処理と DOM の更新を行なう方式もある
  - Blazor でいうところの Blazor Server に相当?

ブラウザ上で動作する例







Rust / Yew

Rust 版 React

コンポーネント指向

JSX 風に Rust コード内にマークアップを記述

Rust は他にも多数のフロントエンドライブラリ/フレームワークが存在する

“Rust web framework comparison”

Python / PyScript

Python 版 jQuery?

Python で DOM を操作するコードを書く

SPA をゴリゴリ書くというよりも、豊富なパッケージ資産でデータ可視化をサクッと行なう感じ?

Ruby / Bormaŝino

ブラウザ上で動く Sinatra?

ERB でビューを書きコントローラを実装して POST / GET / PUT / DELETE を処理する

フォーム送信がサーバーに送られずブラウザ内のコントローラで処理して次のビューを描画する

Dart / Flutter on the Web

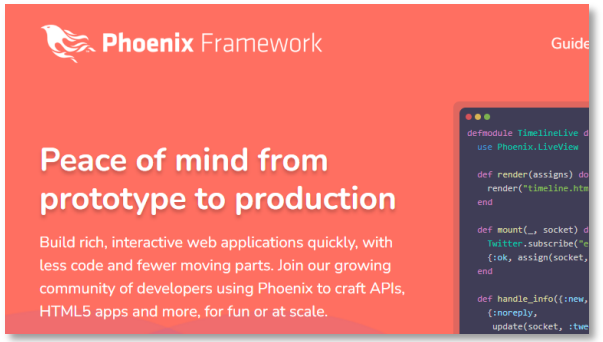
Dart コードは JavaScript にコンパイルされる

Canvas に全部描画する感じ?(HTML に展開する場合もある模様)

サーバー側で動作する例

PHP / Laravel / Livewire

Ruby / Rails / Hotwire



Elixir / Phoenix / LiveView

C# で SPA を書く BLAZOR が歩んできた道

BLAZOR が歩んできた道 - 2017年

2017

BLAZOR が歩んできた道 - 2017年

2017年6月12-16日

NDC Oslo で Steve Sanderson 氏が Blazor をデモ

2017

2018

BLAZOR が歩んできた道 - 2018年

- コンポーネントモデル
- ルーティング
- レイアウト
- フォーム検証
- 依存性注入(DI)
- JavaScript相互運用
- サーバー側レンダリング
- デバッガ
- インテリセンス
- スコープ化された CSS
- ホットリロード
- WebAssembly AOT

2017

2017年6月12-16日

NDC Oslo で Steve Sanderson 氏が Blazor をデモ

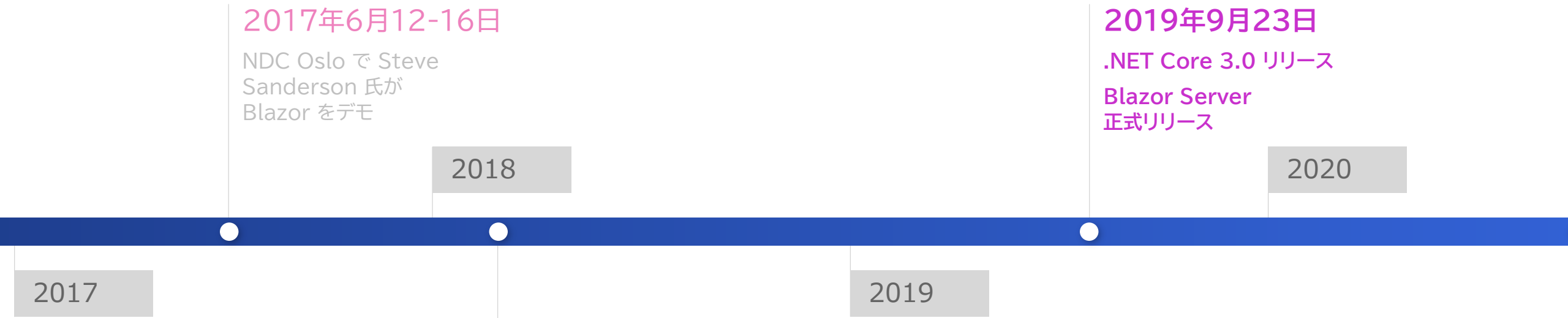
2018

2018年2月6日

.NET 公式ブログにて、ASP.NET の実験プロジェクトとして公式に紹介

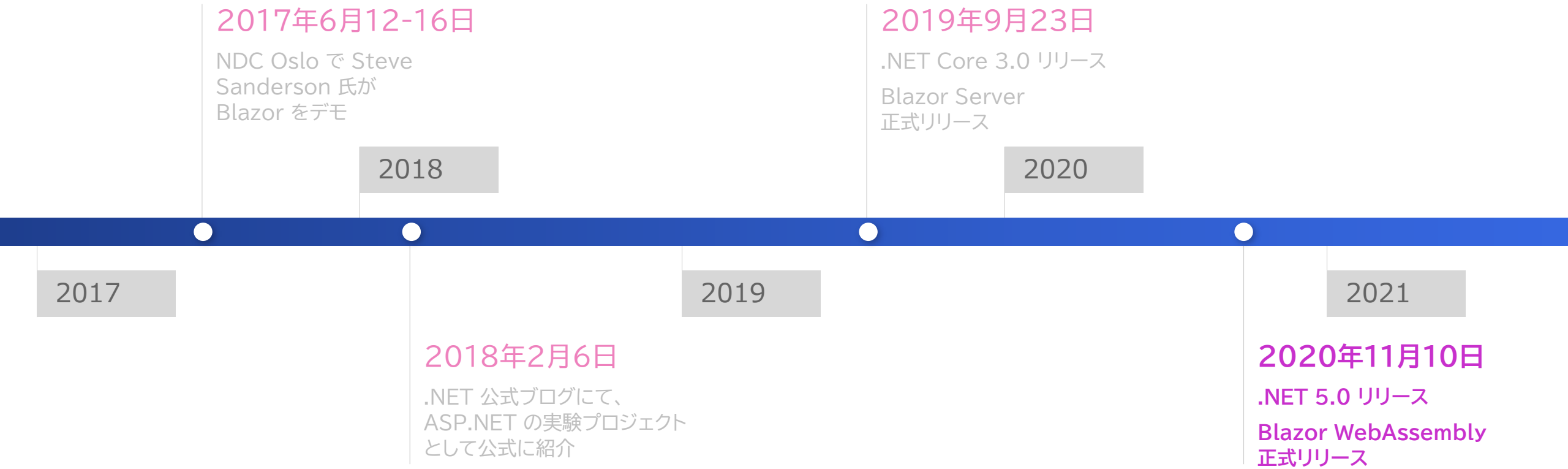
BLAZOR が歩んできた道 - 2019年

- コンポーネントモデル
- ルーティング
- レイアウト
- フォーム検証
- 依存性注入(DI)
- JavaScript相互運用
- サーバー側レンダリング
- デバッガ
- インテリセンス
- スコープ化された CSS
- ホットリロード
- WebAssembly AOT



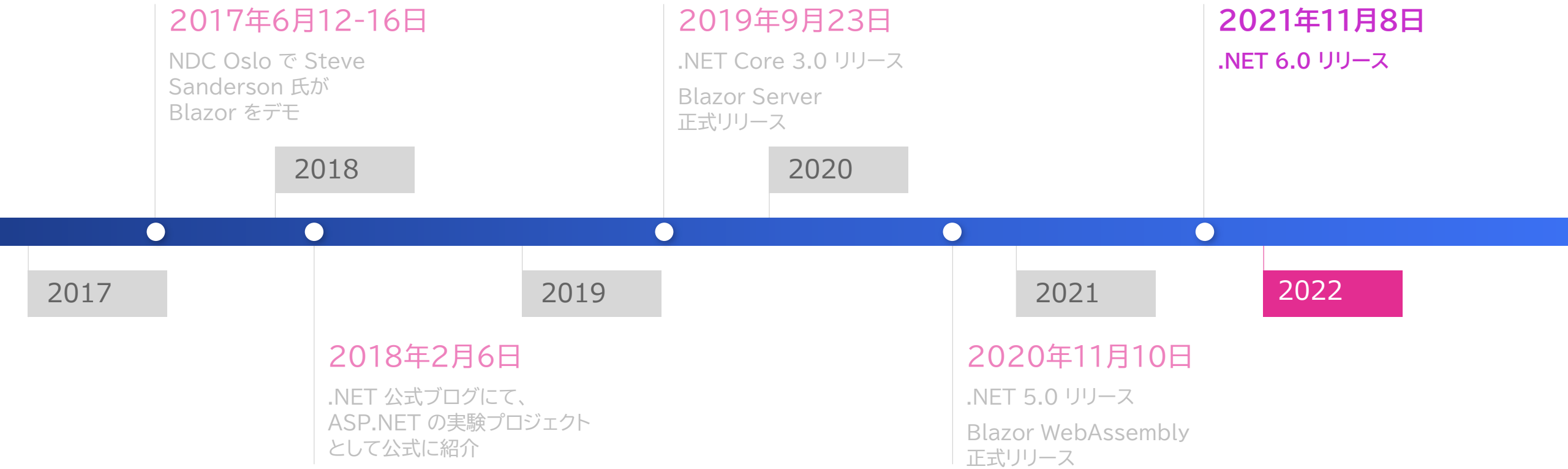
BLAZOR が歩んできた道 - 2020年

- コンポーネントモデル
- ルーティング
- レイアウト
- フォーム検証
- 依存性注入(DI)
- JavaScript相互運用
- サーバー側レンダリング
- デバッガ
- インテリセンス
- スコープ化された CSS
- ホットリロード
- WebAssembly AOT



BLAZOR が歩んできた道 - 2021年

- コンポーネントモデル
- ルーティング
- レイアウト
- フォーム検証
- 依存性注入(DI)
- JavaScript相互運用
- サーバー側レンダリング
- デバッガ
- インテリセンス
- スコープ化された CSS
- ホットリロード
- WebAssembly AOT



BLAZOR が歩んできた道

- NDC での発表から**既に5年**、Blazor は発展途上ではなく、**成熟期に入った**と言えるのではないだろうか
  - 潤沢なパッケージ資産とエコシステム
  - コード補完と分析、デバッガ、ホットリロードが提供する、**リッチな開発者体験**
- とはいえ、これからも Blazor は進化を続けていく
  - 今回は取り上げませんでしたが、.NET MAUI とのハイブリッド構成で iOS や Android デバイス向けモバイルアプリ開発にも Blazor を利用可能
  - **.NET 7** は今年 2022年 11月リリース予定!

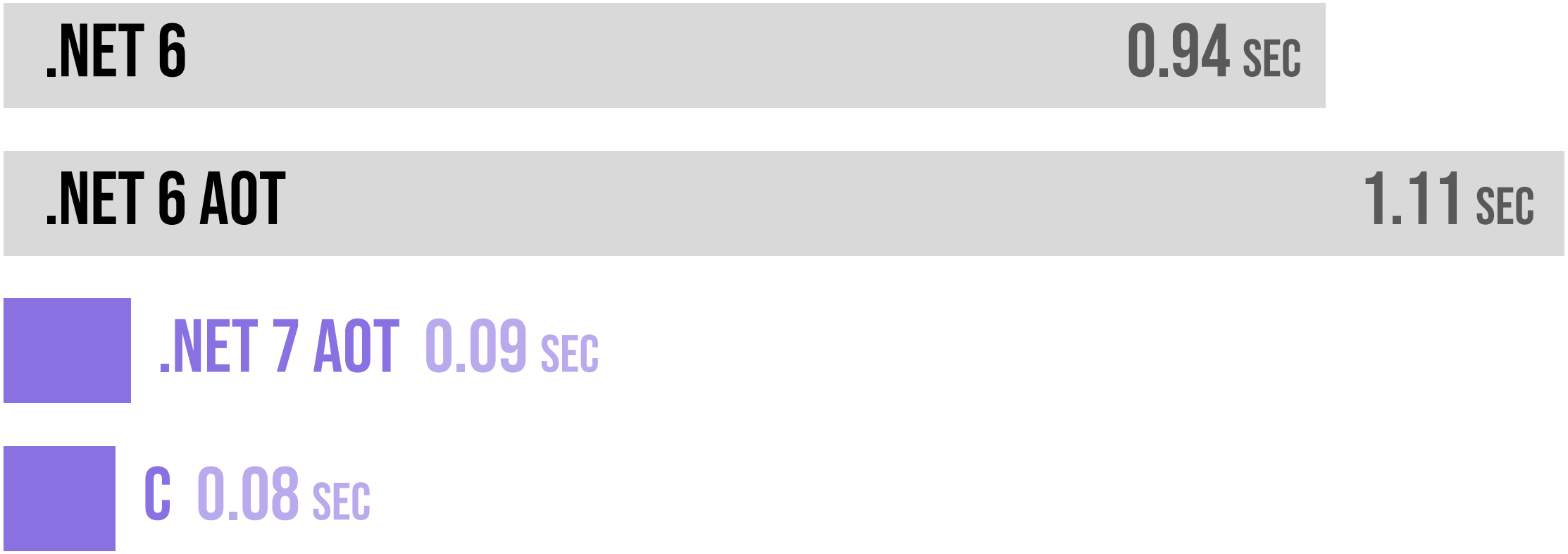
.NET 6 から使える “NATIVE DEPENDENCY” とは


DEMONSTRATION

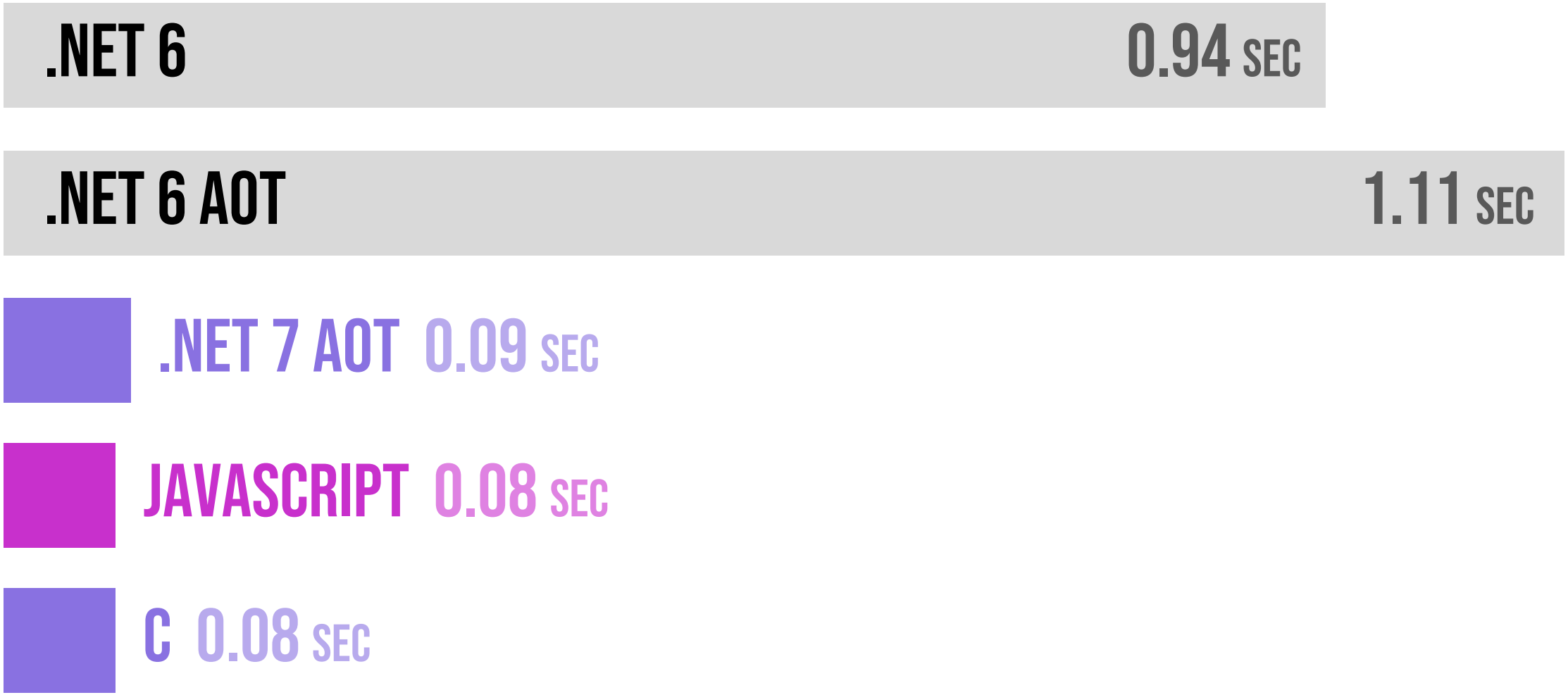
.NET と WEBASSEMBLY NATIVE との速度比較

.NET と WEBASSEMBLY NATIVE との速度比較

.NET と WEBASSEMBLY NATIVE との速度比較



.NET と WEBASSEMBLY NATIVE との速度比較



BLAZOR WEBASSEMBLY の NATIVE DEPENDENCY

- 任意の言語・処理系から生成された **WebAssembly バイナリ内の関数**を、Blazor WebAssembly から容易に呼び出せる
  - さらに NuGet パッケージ化されていれば、これを利用する Blazor WebAssembly アプリ開発者は、何もせずに**C＃構文**のままに**透過的に利用可能**
- **Rust** によるクライアント Web 開発の盛り上がりの恩恵にあずかれるかも!?
  - Rust で書かれた計算中心のライブラリを、Blazor WebAssembly プログラミングから利用できる可能性

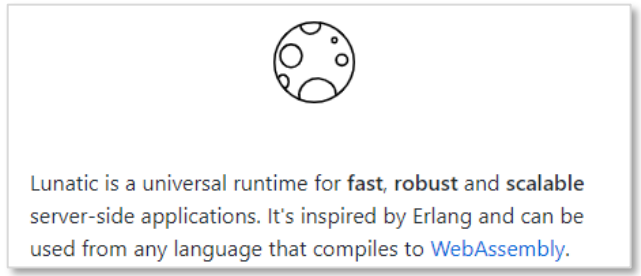
Blazor WebAssembly を超えたその先

ブラウザの外へ活躍の場を広げる WEBASSEMBLY

- **WASI** - WebAssembly コードを、**ブラウザ以外の環境で実行**できるようにするための、システム資源にアクセスする API の**仕様**
  - *“The WebAssembly System Interface”*
- WASI をサポートした **WebAssembly 実行環境の実装**がいくつかある

Wasmtime

Wasmer

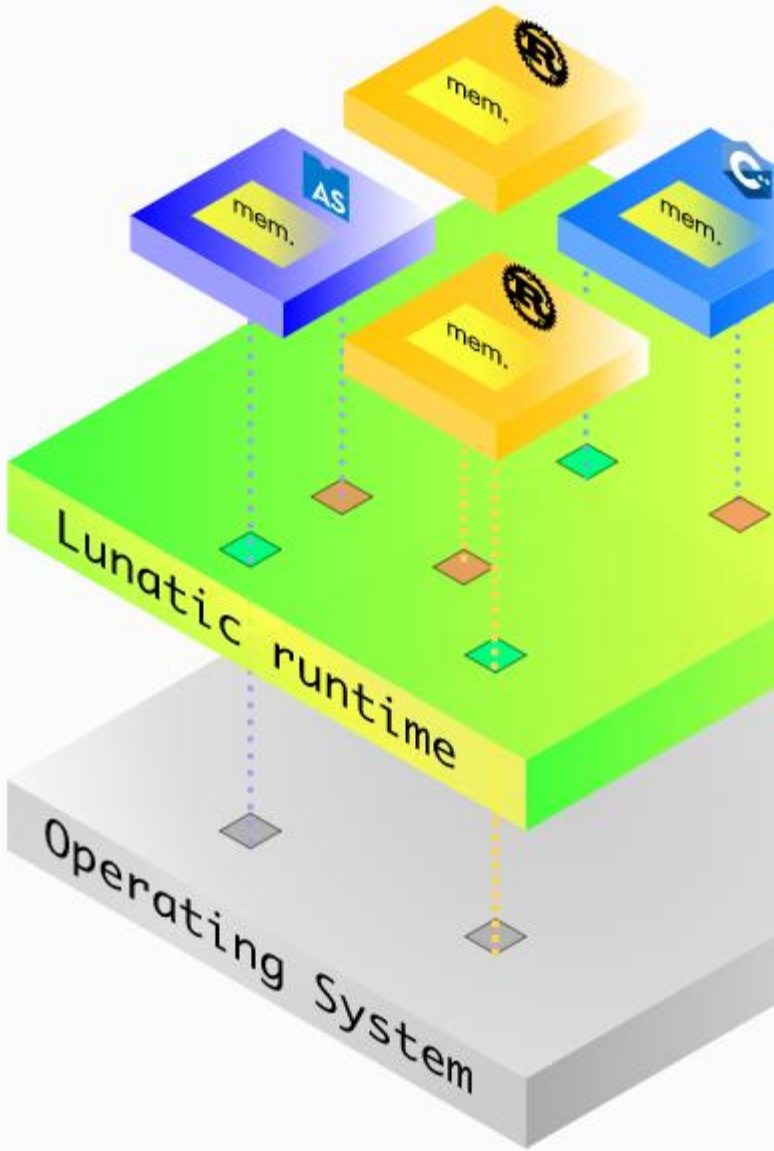


Lunatic


DEMONSTRATION

WEBASSEMBLY による新しいコンテナの形

- **瞬時に起動**する WebAssembly モジュール
  - 50マイクロ秒でモジュール起動
- 完全に分離・制御された実行空間
  - メモリやCPU使用量の割り当てを監視・制御可能
  - 必要な資源へのアクセスのみ許可
- 1要求を**処理するたび**にモジュールを**破棄**
  - **ガベージコレクションが不要**になる可能性!



おわりに

おわりに

- JavaScript 以外によるクライアント Web 開発は選択肢が増えつつある
- Blazor WebAssembly は**成熟期**に入りつつある
  - 今回はとくに、ネイティブな WebAssembly バイナリとの連携強化に着目
- WebAssembly の活躍の場が広がるのにあわせて、C#開発者にまたひとつ選択肢が加わる未来
  - Steve Sanderson 氏の実験プロジェクトながら、既に WASI ランタイム上で**フルセットの ASP.NET Core Webアプリが稼働**し、**デバッガまで機能**している
  - **新しいコンテナ、新しいマイクロサービスの形**になるのか


TWITTER HASH TAG
#osc22do
#osc22do_e10