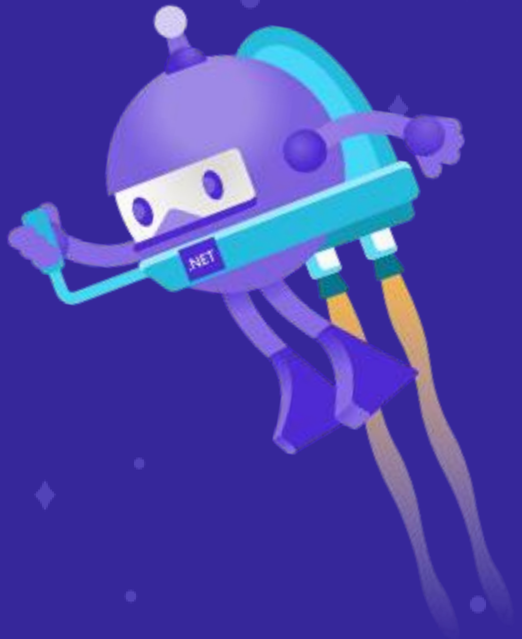
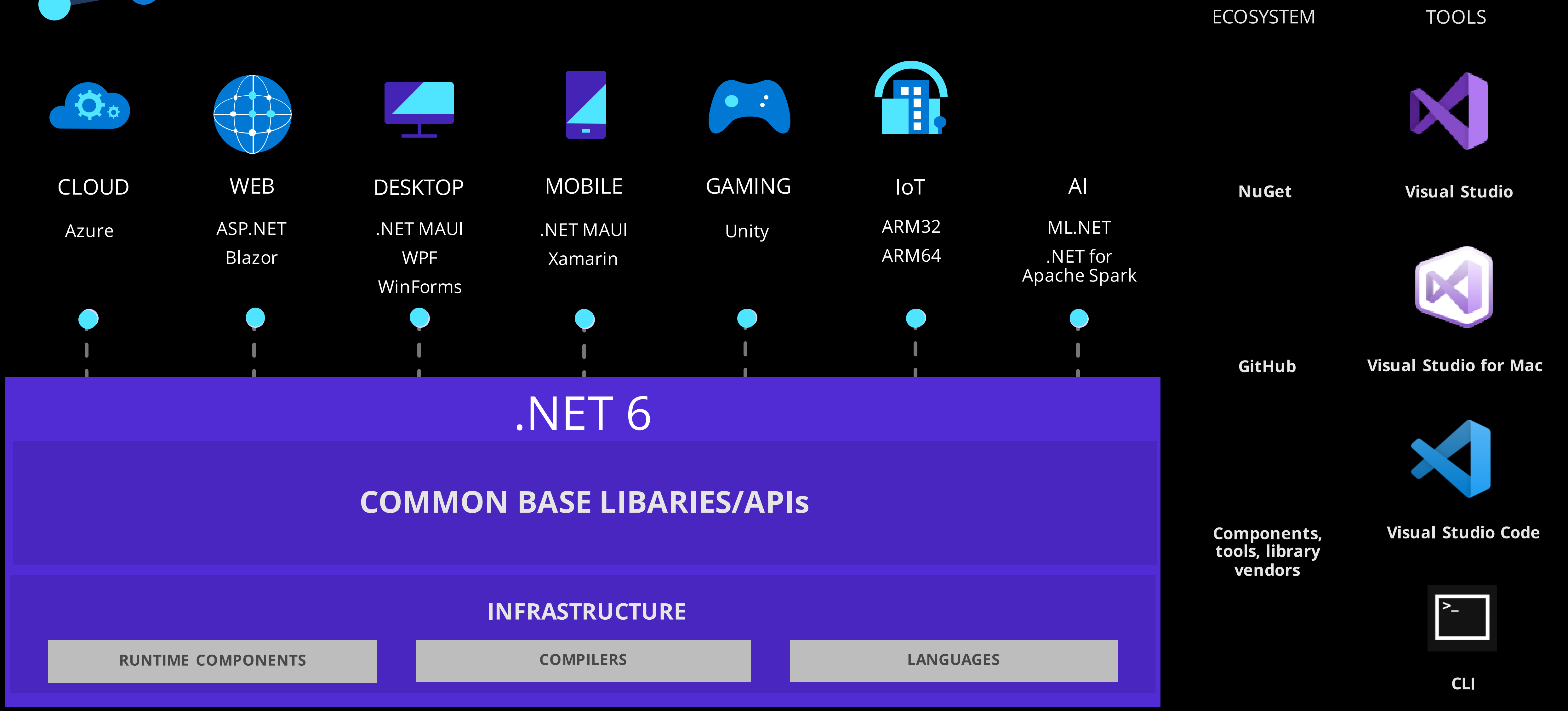
What's new in C# 10 and .NET 6

Oleksii Nikiforov
@nikiforovall



.NET – A unified development platform



ECOSYSTEM

TOOLS

CLOUD

Azure

WEB

ASP.NET

Blazor

DESKTOP

.NET MAUI

WPF

WinForms

MOBILE

.NET MAUI

Xamarin

GAMING

Unity

IoT

ARM32

ARM64

AI

ML.NET

.NET for Apache Spark

NuGet

GitHub

Components, tools, library vendors

Visual Studio

Visual Studio for Mac

Visual Studio Code

CLI

.NET 6

COMMON BASE LIBARIES/APIs

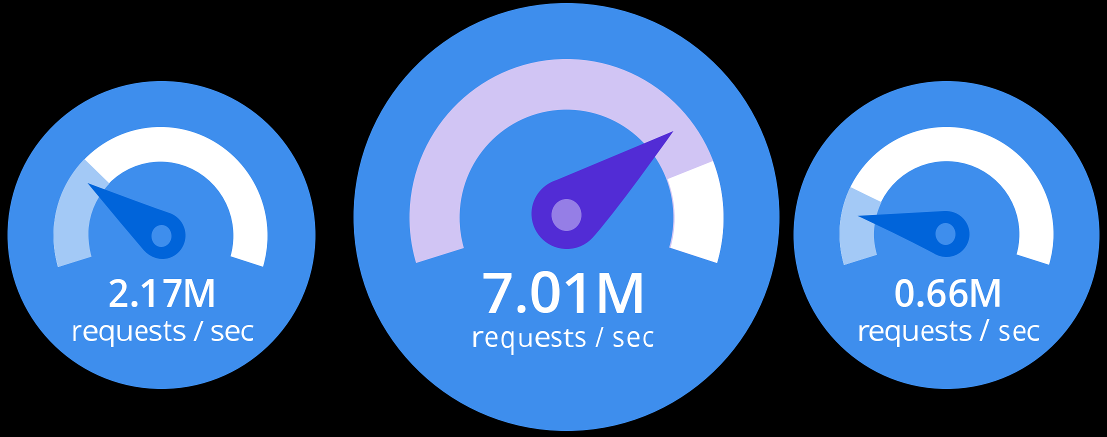
INFRASTRUCTURE

RUNTIME COMPONENTS

COMPILERS

LANGUAGES

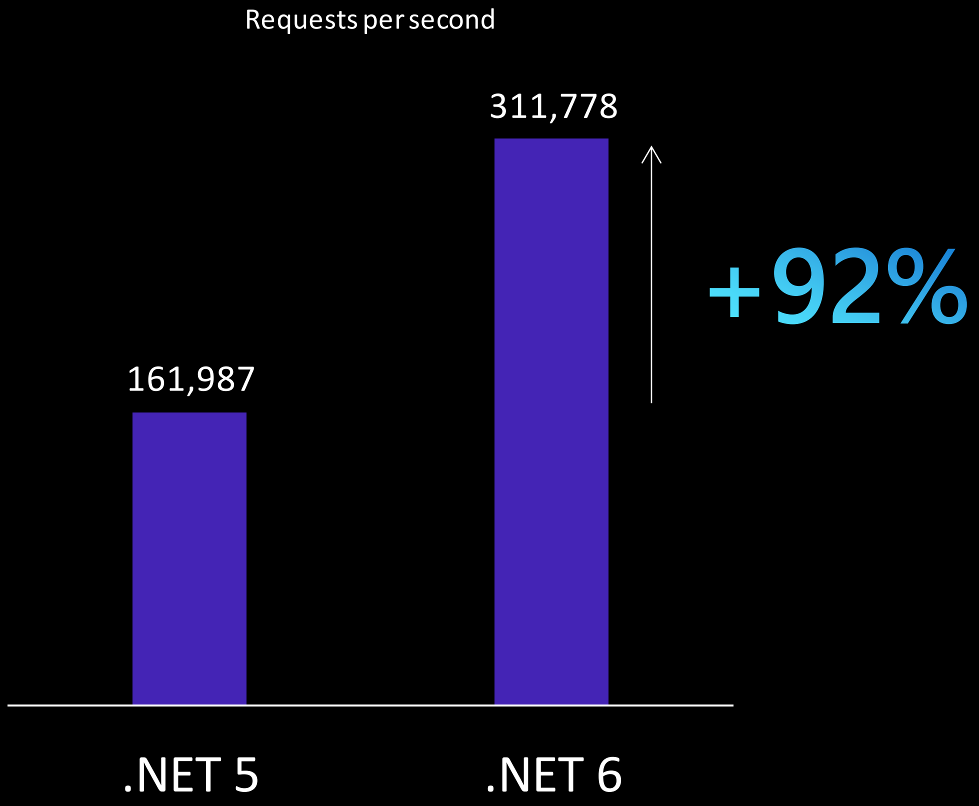
.NET 6 Performance



Java Servlet .NET Node.js

>10X faster than Node.js

ASP.NET Core web framework



Entity Framework Core Performance

Sources:
https://www.techempower.com/benchmarks/#section=data-r20&hw=ph&test=plaintext
https://www.techempower.com/benchmarks/#section=test&runid=3fc99e53-f60d-428e-9937-e809880d3da2&hw=ph&test=fortune&a=2&o=e

RELEASED

.NET 6

- Unified common base libraries & SDK
- Industry leading performance
- Simplified development, easier to get started
- New C# 10, F# 6 releases
- Apple Silicon (Arm64) support
- Long-Term Support Release

get.dot.net/6

Global Usings

```
global using Model;  // Global usings apply to entire project
// using System;     // Implicit usings for each project type

global using static System.Console;
```

```
<PropertyGroup>
    <ImplicitUsings>true</ImplicitUsings> // MSBuild feature
</PropertyGroup>
```

File-scoped namespaces

```
using Model;  // One namespace per file

* dotnet tool install --global dotnet-format
```

csharp_style_namespace_declarations = file_scoped:error

```
* dotnet format style
```

```
(Extras) [CallerArgumentExpression("arg")]
(Extras) Constant interpolated strings
(Extras) Parameterless constructors in structs
(Extras) Extended property patterns
```

Record structs

```
// Records can be structs as well as classes

public record struct Point(double X, double Y, double Z);

public record class Point(double X, double Y, double Z);
```

(Extras) Expression 'with' in structs
(Extras) Expression 'with' in anonymous types
(Extras) Assignment and declaration in the same deconstruction

Minimal APIs for cloud native apps

```
var app = WebApplication.Create(args);

app.MapGet("/", () => "Hello World!");

app.Run();
```

Lightweight, single-file, cloud native APIs

Low ceremony, top-level C# programs

Easy to get started

Path to MVC

(Extra) System.Numerics.BitOperations
(Extra) Explicit return type in lambdas
(Extra) Lambdas with attributes

LINQ Improvements

```
// Chunking
// Splits the elements of a sequence into chunks of size at most size.
Enumerable.Chunk<TSource>(IEnumerable<TSource> source, int size)

// Index Support for ElementAt
Enumerable.ElementAt<TSource>(IEnumerable<TSource> source, Index index)

// Range Support for Take
Enumerable.Take<TSource>(IEnumerable<TSource> source, Range range)

// Default Parameters for Common Methods
Enumerable.FirstOrDefault<TSource>(IEnumerable<TSource> source, TSource defaultValue)

// MaxBy and MinBy
Enumerable.MaxBy<TSource, TKey>(IEnumerable<TSource> source, Func<TSource,TKey> keySelector)
```

System.Threading.Tasks.Parallel.ForEachAsync

```
// Executes a for-each operation on an IEnumerable<T> in which iterations may run in parallel.
Task ForEachAsync<TSource>(IEnumerable<TSource> source, Func<TSource,CancellationToken,ValueTask> body);

// Executes a for-each operation on an IAsyncEnumerable<T> in which iterations may run in parallel.
Task ForEachAsync<TSource>(IAsyncEnumerable<TSource> source, Func<TSource,CancellationToken,ValueTask> body);
```

(Extras) Random.Shared
(Extras) Task.WaitAsync

System.Threading.PeriodicTimer

```
using var timer = new PeriodicTimer(TimeSpan.FromSeconds(1));

while (await timer.WaitForNextTickAsync(CancellationToken.None))
{
    Process();
}
```

(Extras) ArgumentNullException.ThrowIfNull

System.Collections.Generic.PriorityQueue

```
PriorityQueue<string, int> priorityQueue = new();
priorityQueue.Enqueue("Second", 2);
priorityQueue.Enqueue("First", 1);

while (priorityQueue.Count > 0)
{
    var item = priorityQueue.Dequeue();
}
```

System.DateOnly and System.TimeOnly

```
// public DateOnly(int year, int month, int day)
// public DateOnly(int year, int month, int day, Calendar calendar)
DateOnly dateOnly = new(2021, 11, 25);
Console.WriteLine(dateOnly);
// Output: 25-Nov-21

// public TimeOnly(int hour, int minute)
// public TimeOnly(int hour, int minute, int second)
// public TimeOnly(int hour, int minute, int second, int millisecond)
// public TimeOnly(long ticks)
TimeOnly timeOnly = new(16, 0, 0);
Console.WriteLine(timeOnly);
// Output: 16:00 PM
```

System.Text.Json

```
var order = new JsonObject
{
    ["OrderId"] = 1,
    ["Discounts"] = new JsonArray(
        new JsonObject
        {
            ["DiscountId"] = 1,
            ["Value"] = .05,
        },
    ),
};
```

```
Serialization Notification

Property Ordering

IAsyncEnumerable support

Streams support

Working with JSON DOM
```

System.Text.Json Version="6.0.0"

Upgrade

```
* dotnet tool install --global upgrade-assistant

* dotnet tool install --global dotnet-outdated-tool
```

- Reduce time and difficulty modernizing older .NET codebases
- Guided, step-by-step experience
- Multiple project types supported
- C# & VB.NET languages
- Supports .NET 6

aka.ms/dotnet-upgrade-assistant

Q&A

- Source code

  https://github.com/NikiforovAll/whats-new-in-dotnet6
- Blog post

  https://dev.to/nikiforovall/whats-new-in-net-6-and-c-10-everything-you-wanted-to-know-n2p
- Coding Story

  https://bit.ly/2YKKYGU
- .NET 5 & C#

  https://github.com/NikiforovAll/csharp_workshop
- .NET Conf 2021 Recordings

  https://bit.ly/3DcMZe1
- Find more at:

  https://www.theurlist.com/whats-new-in-dotnet6-and-csharp10