UPGRADING YOUR C# SKILLS TO .NET 7

with Tim Corey

WORKSHOP OVERVIEW

- Understanding the .NET Framework through .NET 7
- Project Types Overview
- Changes with .NET 6+
- Understanding ASP.NET Core
- Understanding Blazor
- Azure Functions
- Docker
- Working with SQL
- Minimal APIs
- Best Practices
- Tips & Tricks

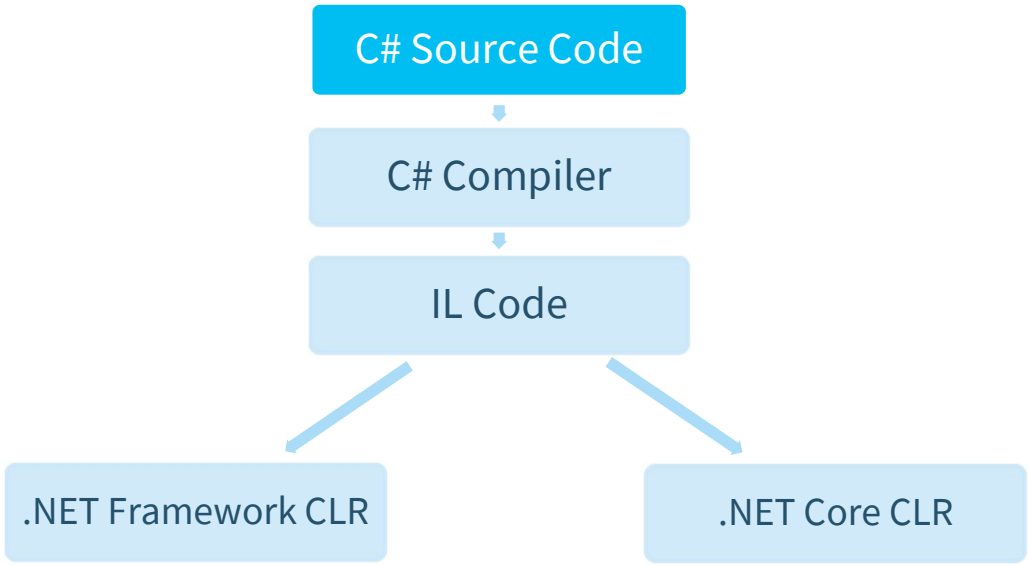
.NET VERSIONS EXPLAINED

.NET, .NET Core, .NET Framework, .NET Standard, and more

WHY DID .NET CORE NEED TO EXIST?

1. The .NET Framework was full of patches and workarounds that slowed it down.
2. The .NET Framework was tied directly into Windows.
3. The .NET Framework was designed to house everything you needed in one package.
4. The .NET Framework was designed originally over 20 years ago when the Internet was a very different place.

THE COMMON LANGUAGE RUNTIME (CLR)



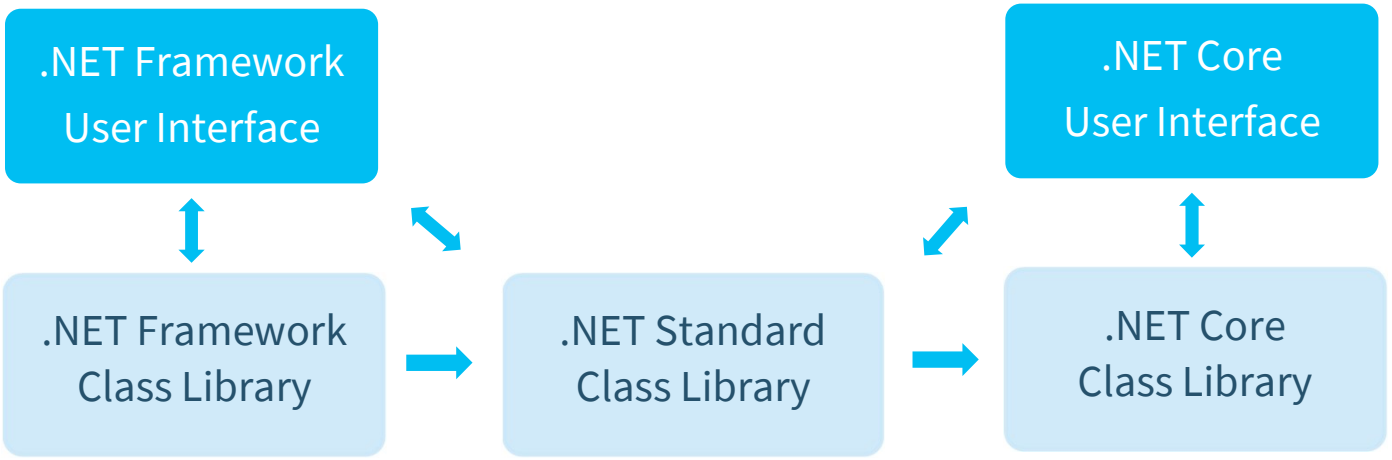
BENEFITS OF .NET CORE OVER THE .NET FRAMEWORK

- Works on most platforms – Windows, Mac, Linux*, Android, iOS, Xbox, IoT
- Faster
- Modular
- More features
- Follows best practices
- Open-source
- Modern
- Compliant with industry standards
- Easy to upgrade

* Linux desktop available using third-party, open-source tools like Uno and Avalonia

.NET STANDARD EXPLAINED

A common interface of APIs for any .NET implementation

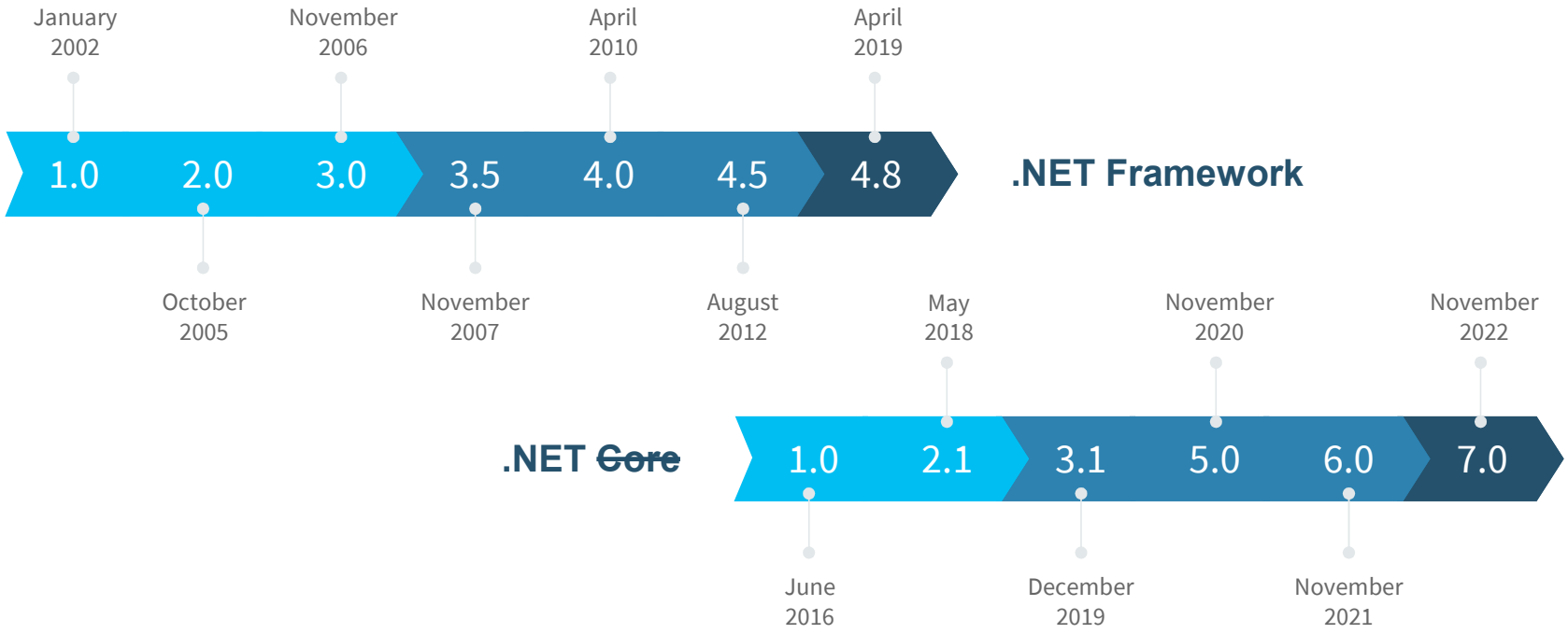


.NET STANDARD VERSIONS

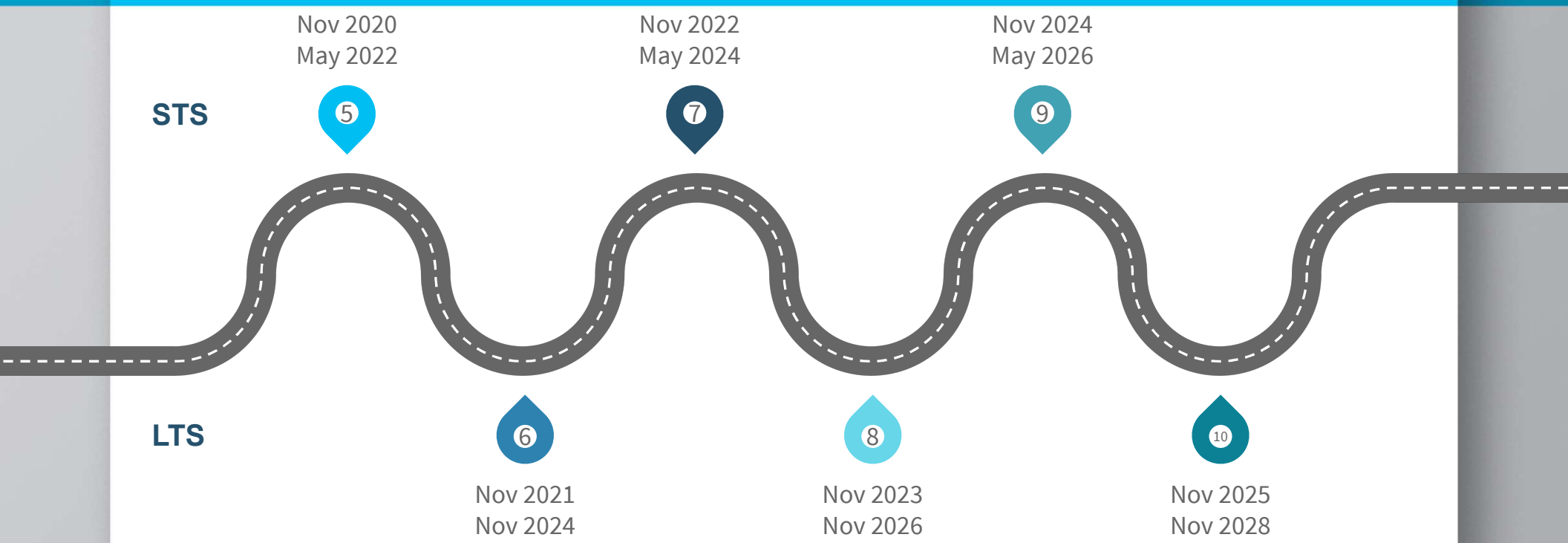
- V1.0 – 7,949 out of 37,118 APIs
  - .NET Framework 4.5 & higher
  - .NET Core 1 & higher
- V2.0 – 32,638 out of 37,118 APIs
  - .NET Framework 4.7.2 & higher
  - .NET Core 2 & higher
- V2.1 – 37,118 out of 37,118 APIs
  - No .NET Framework support
  - .NET Core 3.0 & higher

* .NET Standard also supports Xamarin, Mono, UWP, Unity, and more

.NET TIMELINES



UPGRADE AND SUPPORT CYCLE



WHEN TO USE EACH

.NET Framework

- Additions to existing projects
- When forced to use old operating systems

.NET

- New projects
- When upgrading projects
- When you have a choice

WHAT ABOUT ASP.NET AND ASP.NET CORE?

ASP.NET

- WebForms
- MVC
- WebAPI

ASP.NET Core*

- Razor Pages
- MVC
- API
- Blazor Server
- Blazor WebAssembly

* ASP.NET Core will not drop the “Core” in order to not be confused with ASP.NET

PRACTICE

Create a **.NET Framework** Console App that asks for your name and then says “Hello <your name>”.

Upgrade the app to .NET 7

Bonus: Add the NuGet Package Dapper before upgrading. Verify it is still there at the end.

.NET PROJECTS OVERVIEW

Desktop, Web, Mobile, and Miscellaneous project types

DESKTOP PROJECT TYPES

1. Windows Forms (WinForms)
2. WPF
3. UWP
4. .NET MAUI
5. Blazor Hybrid
6. Console

WEB PROJECT TYPES

1. API (Web API)
2. Razor Pages
3. MVC
4. Blazor Server
5. Blazor WebAssembly
6. gRPC
7. Azure Functions

MOBILE PROJECT TYPES

1. .NET MAUI
2. Blazor Hybrid

* 3rd party tools like Uno and Avalonia are additional options

MISCELLANEOUS PROJECT TYPES

1. Worker Service
2. Class Library
3. Test Projects (xUnit, nUnit, & MSTest)

MORE INFORMATION

For more information about C# project types, how to get started in them, and which project type to pick for a given situation, go to the C# Projects site:

https://www.CSharpProjects.com

CHANGES WITH .NET 6+

Program.cs, no Startup.cs, Hot Reload, Tuples, Records, and more

PRACTICE

Create an ASP.NET Core web project type of your choice. Add a class library.

Create a method in the library that takes in a full name and splits it into first and last names. Return both names. Add a global using for the library.

Bonus: Add the class to dependency injection.

UNDERSTANDING ASP.NET CORE

The 5 types, Dependency Injection, Appsettings, and more

PRACTICE

Build a web app that displays a connection string on the main page, along with the time at which a custom class was instantiated.

The time should survive a page refresh.

Bonus: Put the connection string in a location that will not be included in source control.

UNDERSTANDING BLAZOR

The 3 types, components, sharing code, and more

PRACTICE

Create a Blazor project of your choice. Create a new component that counts up to 10 and then stops counting.

Display the component on the main page.

Bonus: Hide the counter on the main page once it is complete. Use this component on another type of Blazor project.

AZURE FUNCTIONS

What they are, how to build them, the choices, and more

DOCKER

Why it is useful, creating images, running containers, and more

PRACTICE

Create a simple website (just index.html) with some sample data.

Get it to run as a Docker container.

Bonus: Update the index.html page and redeploy it as a newer version of the container without removing the older image.

WORKING WITH SQL IN C#

Creating a SQL database, stored procedures, using Dapper, and more

PRACTICE

Build a simple SQL database using the SQL Server Data Project in Visual Studio. Deploy it to a LocalDB database.

Create a Console app that reads the data from the database. Hardcode the connection string, if needed.

Bonus: Use Docker to host the SQL Server instead of LocalDB.

MINIMAL APIS

Basic creation, Open API, Authentication, and more

PRACTICE

Add an additional endpoint to the API that gets just one employee based upon their ID.

Create the SQL changes necessary to fulfill these changes.

Bonus: Clean the API up and make it more manageable.

BEST PRACTICES

Project structures, nullability, and more

PRACTICE

Pick one of the best practices for APIs (rate limiting, caching, and versioning) and implement it in a new Minimal API.

Do the same task in a full API.

Bonus: Implement all three in one Minimal API project.

TIPS AND TRICKS

Snippets, shortcuts, automation, and more

PRACTICE

Build your own class that implements the IDisposable interface.

Instantiate that class and validate that the Dispose method gets called no matter what happens.

Bonus: Call the class from a Minimal API endpoint.

THANKS!

Any questions?

You can find me at:

- @IAmTimCorey
- YouTube.com/IAmTimCorey
- https://www.IAmTimCorey.com