



.NET Architectural Components

.NET CORE

A .NET app runs in one or more implementations of .NET (.NET Framework, .NET Core, or Mono). There is an API specification common to all implementations of .NET.

*It's called the **.NET Standard**.*

[HTTPS://DOCS.MICROSOFT.COM/EN-US/DOTNET/STANDARD/COMPONENTS#APPLICABLE-STANDARDS](https://docs.microsoft.com/en-us/dotnet/standard/components#applicable-standards)

What is .NET?

<https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet>

.NET is a free, open source developer platform for building many types of applications for multiple different programming platforms.

With .NET, you can use multiple languages, editors, and libraries to build for web, mobile, desktop, games, and IoT.



<https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-implementation-support>

1. Find the row with the .NET implementation you want to run.
2. Find the column in that row with the version you will code to.
3. The column header indicates the **.NET Standard** version that your target supports. Higher **.NET Standard** versions will support your implementation.
4. If you have more than one target platform, pick the smaller version. So if you want to run on .NET Framework 4.5 and .NET Core 1.0, the highest **.NET Standard** version you can use *is* **.NET Standard 1.1**.

[illegible]

.NET Standard - Facts

<https://docs.microsoft.com/en-us/dotnet/standard/net-standard>

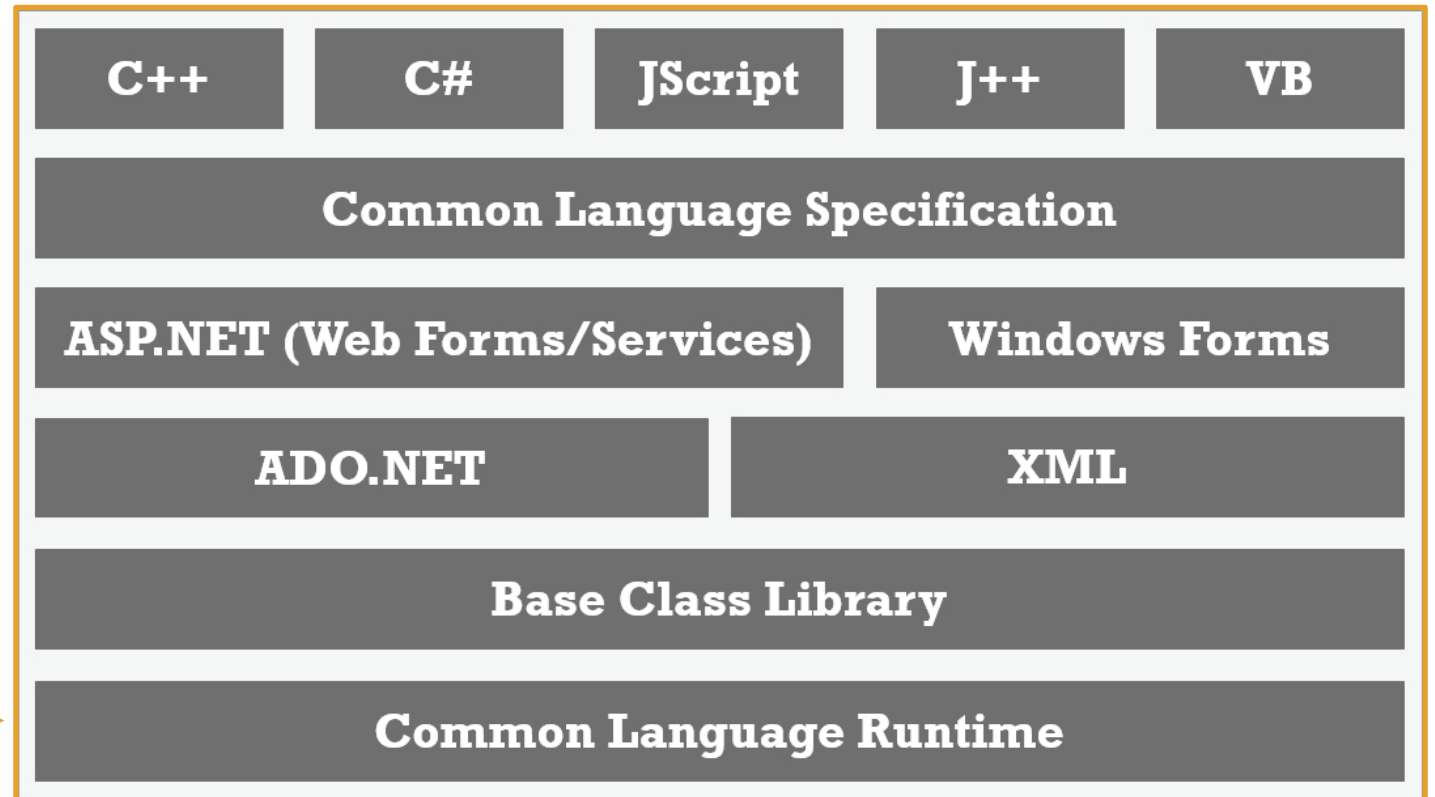
<https://en.wikipedia.org/wiki/NuGet>

- .NET Standard versions are additive. Higher versions incorporate all APIs from previous versions.
 - No 'breaking' changes between versions.
 - The higher the .NET Standard version, the more APIs are available to you.
 - The lower the version, the more platforms implement it.
- Versions are Immutable: Once finalized, .NET Standard versions are frozen.
- .NET Standard reference assemblies are distributed with [NuGet](#) packages.
- The ***NETStandard.Library*** metapackage references the complete set of ***NuGet*** packages that define .NET Standard.
 - The most common way to target ***netstandard*** is by referencing this metapackage.
 - It describes and provides access to the ~40 .NET libraries and associated APIs that define .NET Standard.
- .NET Standard is the replacement for Portable Class Libraries (PCL).

.NET Framework (traditional)

<http://thedotnetproject.blogspot.com/2015/11/the-net-framework.html>

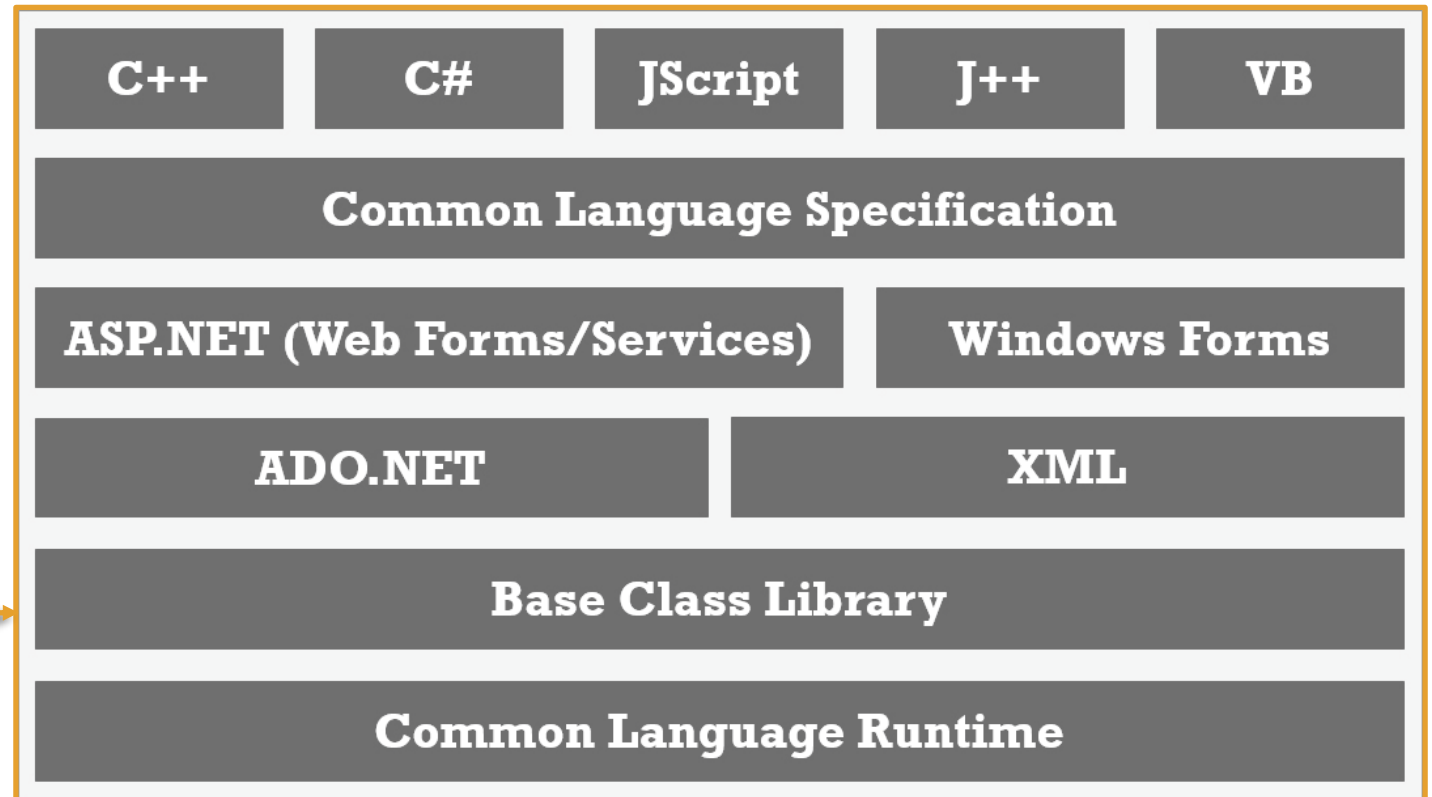
At the bottom of the .Net Framework structure is **Common Language Runtime** (CLR) which handles code execution.



.NET Framework (traditional)

<http://thedotnetproject.blogspot.com/2015/11/the-net-framework.html>

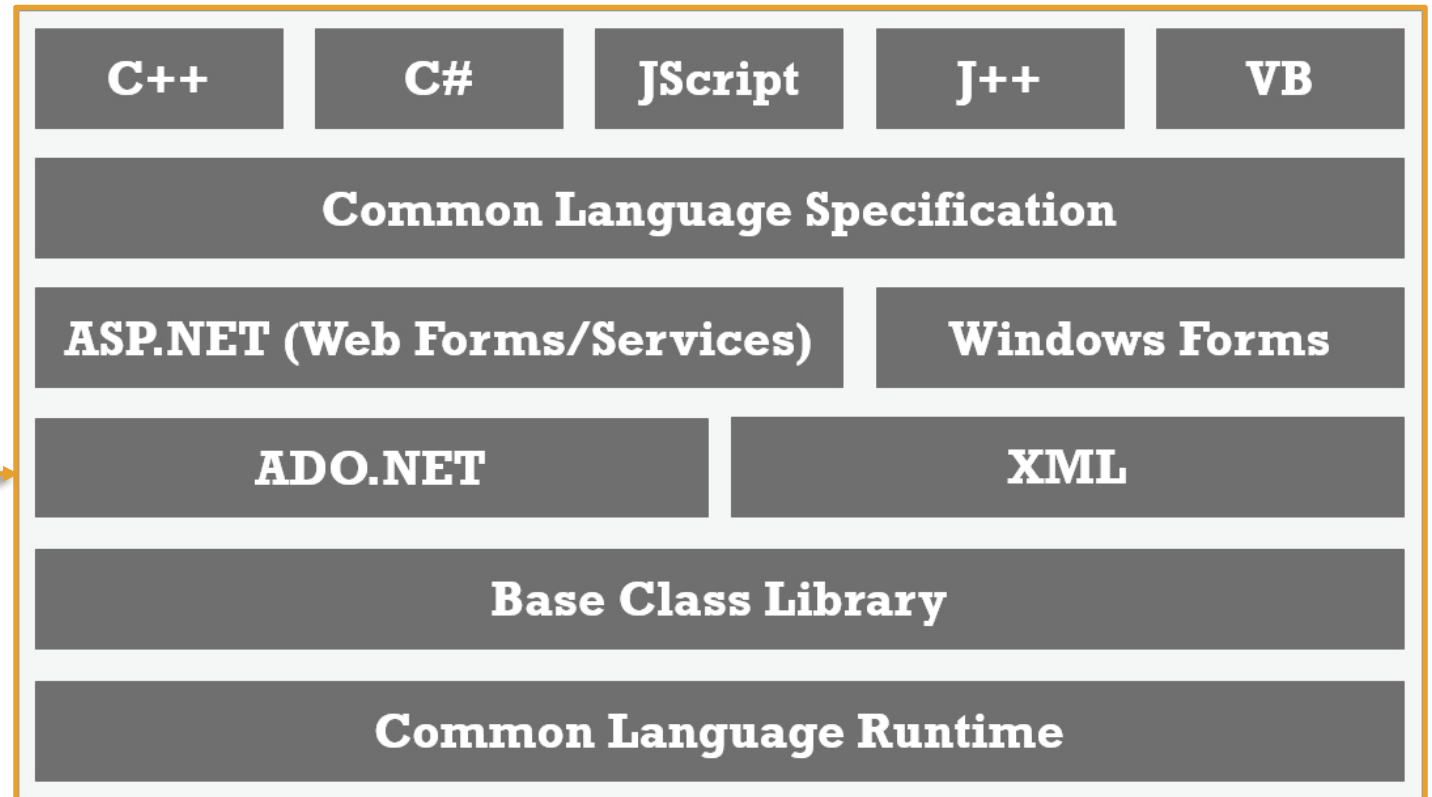
Above the CLR layer is the **.NET Base Class Library** which contains 'runtime libraries' that support common functions like file reading and writing, XML document manipulation, exception handling, application globalization, network communication, threading and reflection, which makes the programmer's job easier.



.NET Framework (traditional)

<http://thedotnetproject.blogspot.com/2015/11/the-net-framework.html>

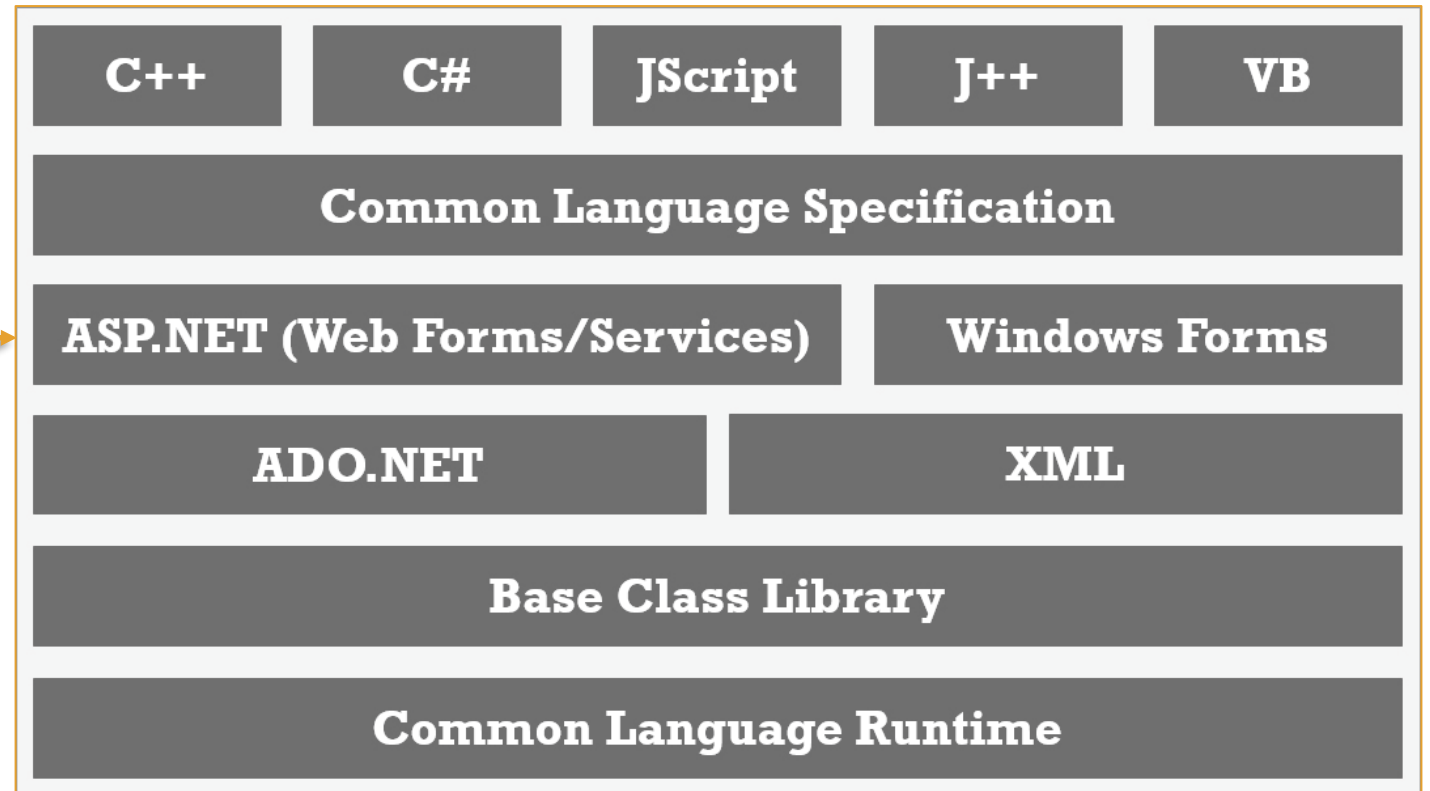
ADO.NET (*library*) and **XML** support tasks related to data access, parsing, manipulation and generating XML.



.NET Framework (traditional)

<http://thedotnetproject.blogspot.com/2015/11/the-net-framework.html>

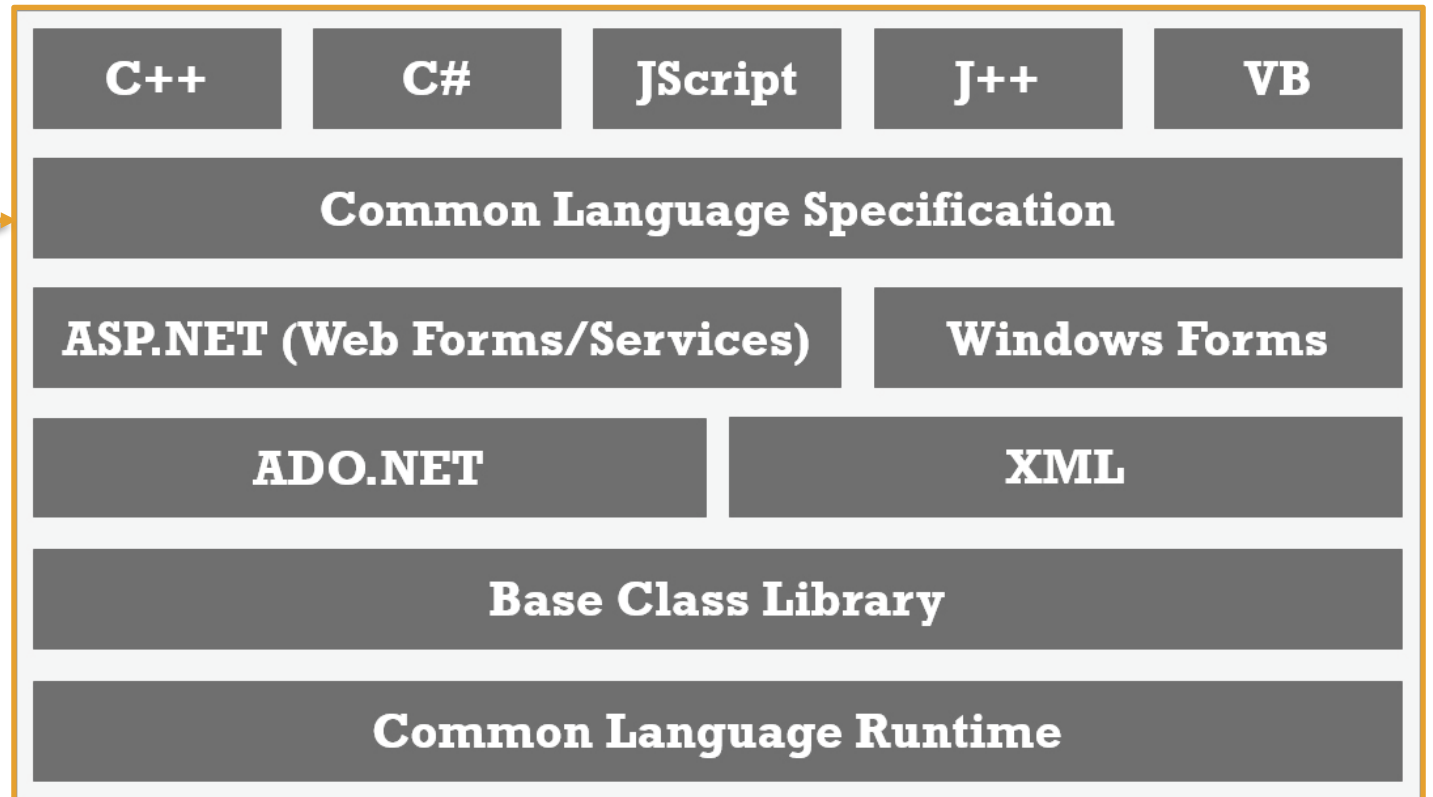
ASP.NET and **Windows Forms** build robust web applications and standard Windows applications as well as develop and consume web services.



.NET Framework (traditional)

<http://thedotnetproject.blogspot.com/2015/11/the-net-framework.html>

Common Language Specification (CLS) then facilitates interoperability between languages. It defines the reasonable subset of Common Type System (CTS), the shared data type that serves a great role in cross-language integration. The flexibility of CTS make many languages adapt to .NET platform.



CURRENT .NET Implementations

<https://docs.microsoft.com/en-us/dotnet/standard/components#universal-windows-platform-uwp>



There are four .NET implementations that Microsoft actively develops/maintains:

- .NET Core
- .NET Framework
- Mono
- Universal Windows Platform (UWP)

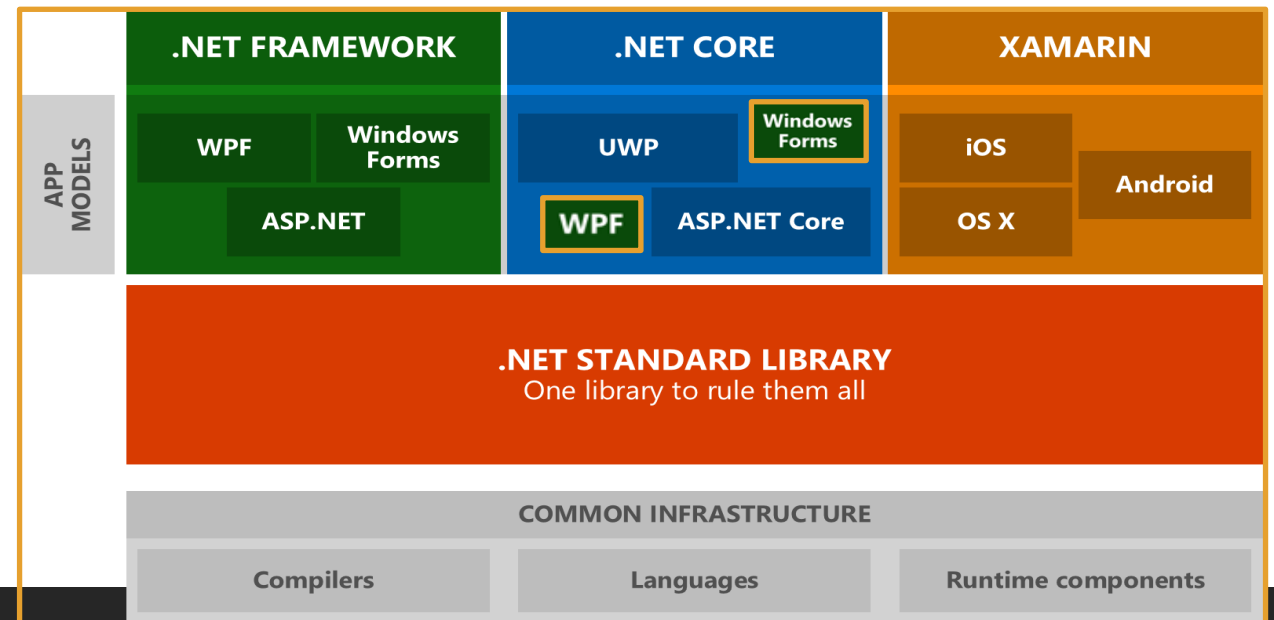
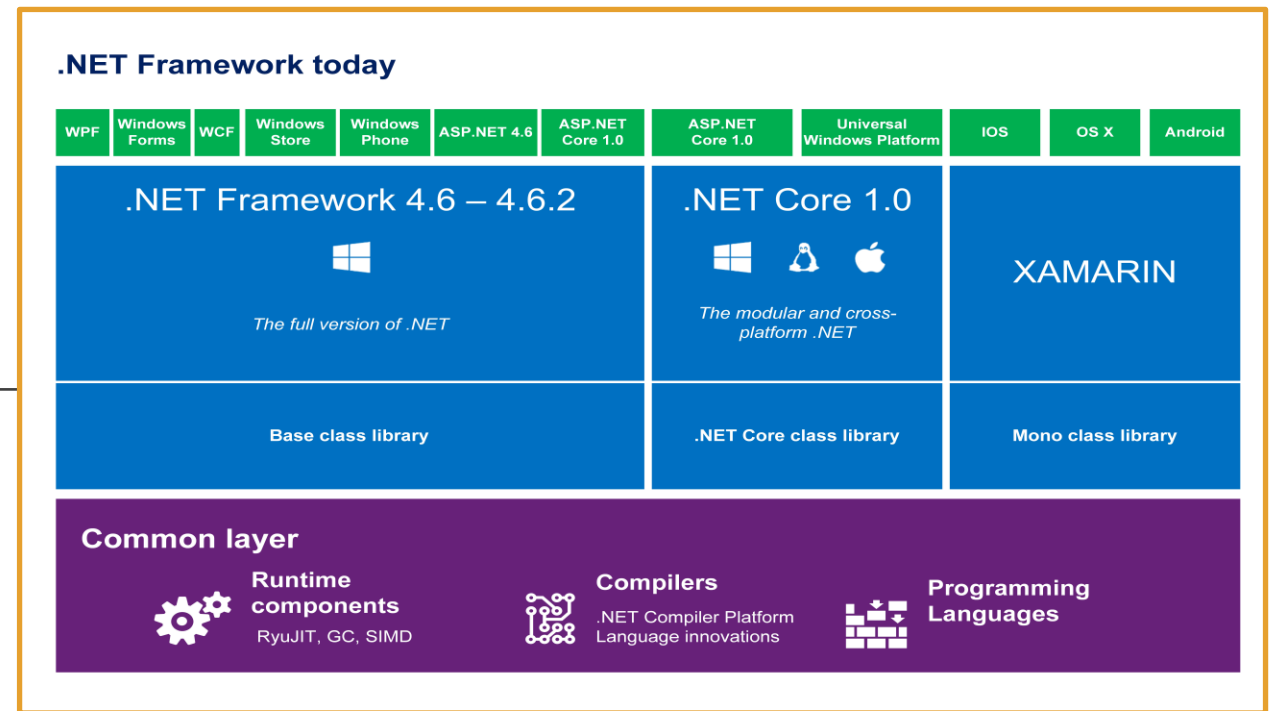
Each of the four implementations includes the following components:

1. One or more runtimes.
 - CLR for .NET Framework,
 - CoreCLR & CoreRT for .NET Core.
2. A Base ***Class Library*** implementing ***.NET Standard*** (possibly additional APIs).
3. One or more application frameworks.
 - .NET Framework and .NET Core have ASP.NET, Windows Forms, and Windows Presentation Foundation (WPF)
4. Development tools.

.NET Framework(newer)

<https://docs.microsoft.com/en-us/dotnet/standard/components#net-framework>

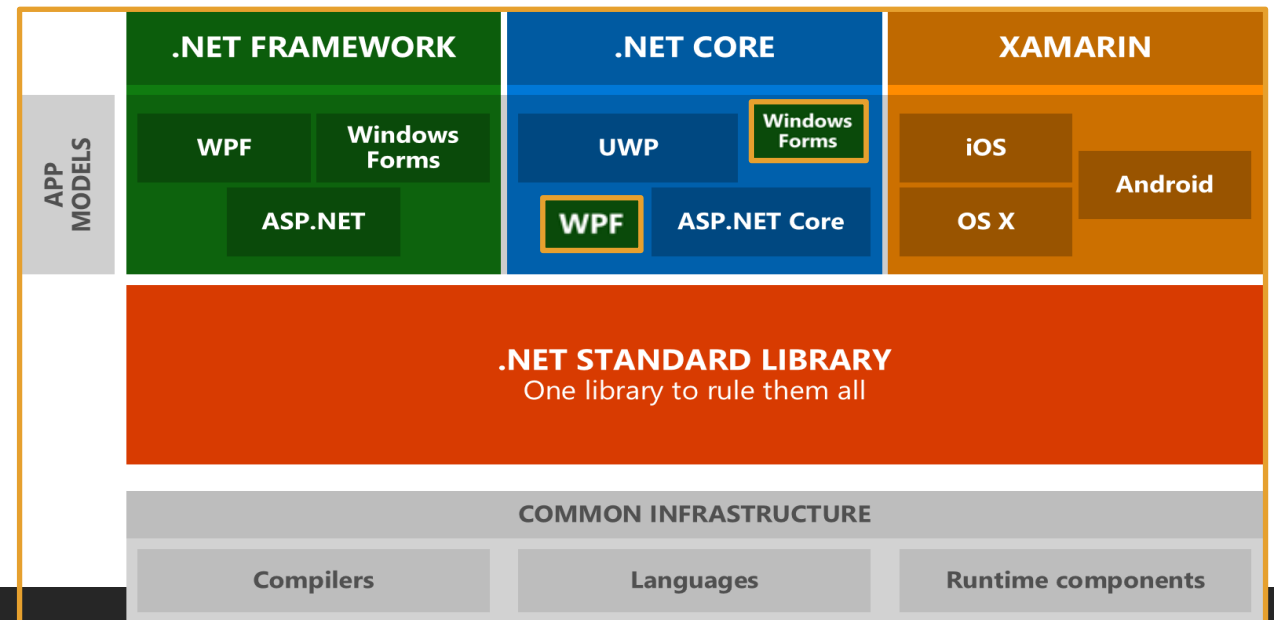
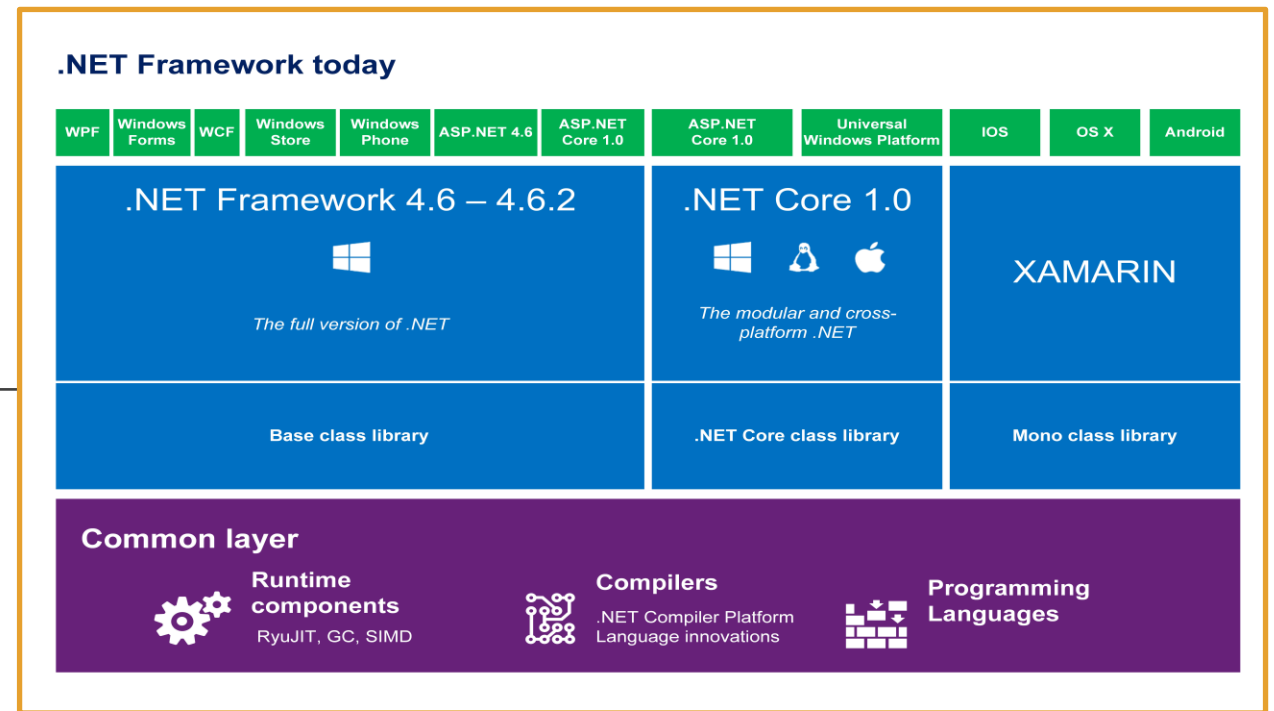
- The .NET Framework is the original .NET implementation that has existed since 2002.
- Versions 4.5 and later implement the .NET Standard
- It contains additional Windows-specific APIs. Like APIs for:
 - Windows desktop development with Windows Forms and WPF.
- The .NET Framework is optimized for building Windows desktop applications.
- [.NET Framework Guide](#)



.NET Core

<https://docs.microsoft.com/en-us/dotnet/standard/components#net-core>
<https://docs.microsoft.com/en-us/dotnet/core/about>

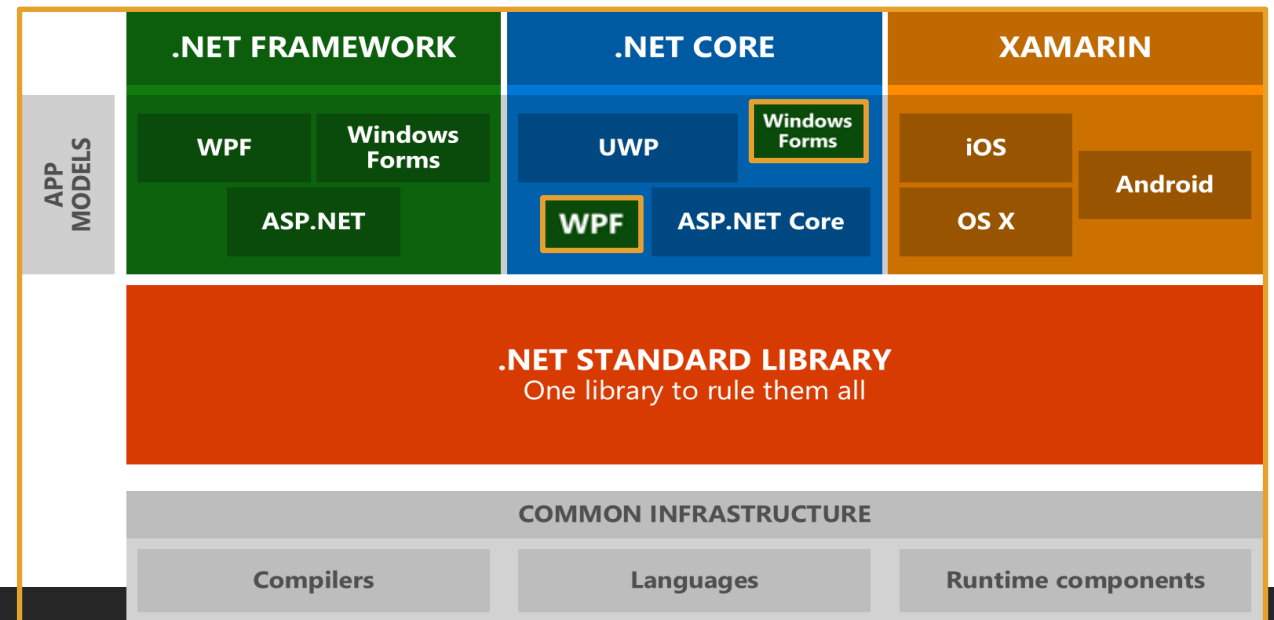
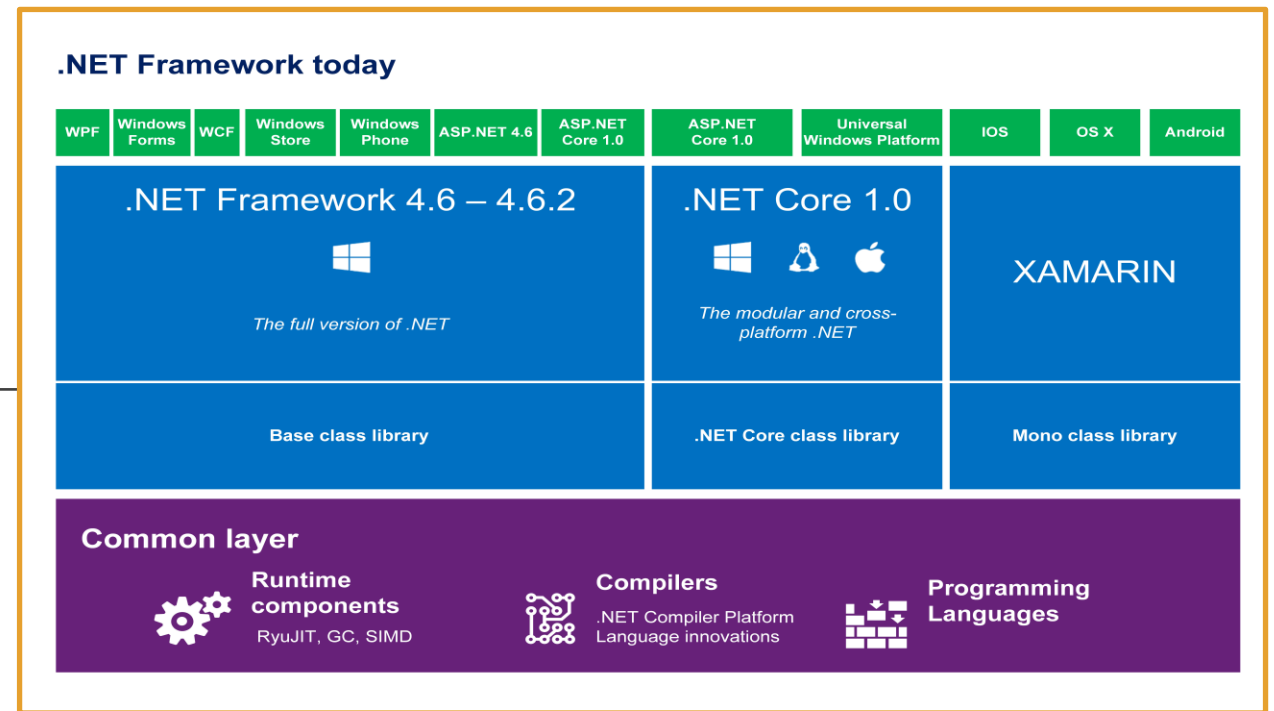
- Since 2016 .NET Core has been a cross-platform implementation of .NET. It's designed to handle server and cloud workloads at scale.
- It runs on Windows, macOS, and Linux.
- It implements the .NET Standard.
- Code that targets .NET Standard can run on:
 - .NET Core.
 - ASP.NET Core,
 - Windows Forms, and
 - Windows Presentation Foundation (WPF)
- [.NET Core Guide](#)



UWP – Universal Windows Platform

<https://docs.microsoft.com/en-us/dotnet/standard/components#mono>

- UWP is an implementation of .NET that is used for building, touch-enabled Windows apps and software for the IoT.
- It's designed to unify different devices including PCs, tablets, phablets, phones, and Xbox.
- UWP provides:
 - a centralized app store,
 - an execution environment (AppContainer)
 - a set of Windows APIs.
- Apps can be written in C++, C#, Visual Basic, and JavaScript.
- When using C# and Visual Basic, the .NET APIs are provided by .NET Core.



.NET Runtimes

<https://docs.microsoft.com/en-us/dotnet/standard/components#net-runtimes>
<https://mattwarren.org/2018/10/02/A-History-of-.NET-Runtimes/>

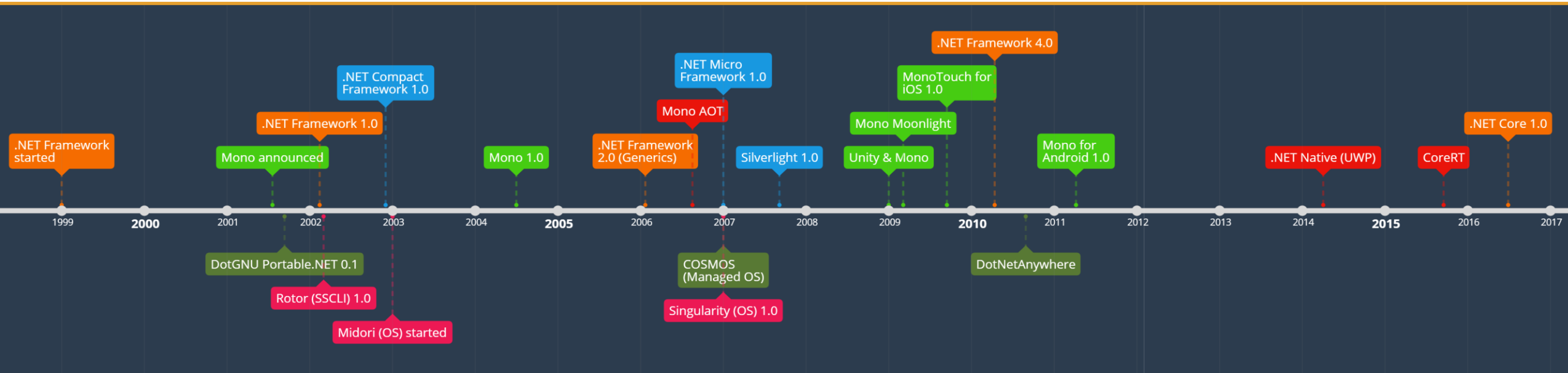
A 'runtime' is the execution environment for a managed program. The OS is part of the 'runtime environment' but is NOT part of the '.NET runtime'. Here are some examples of '.NET runtimes'.

Common Language Runtime (CLR) for the .NET Framework

Core Common Language Runtime (CoreCLR) for .NET Core

.NET Native for Universal Windows Platform

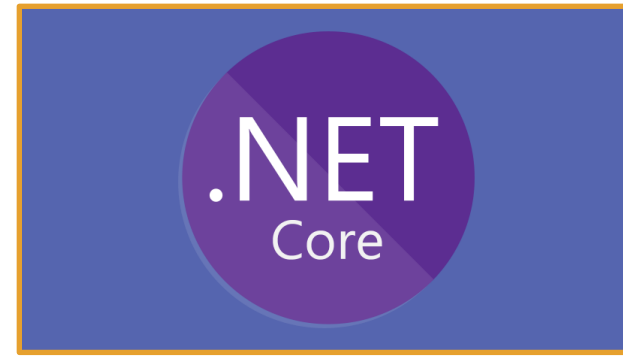
The Mono runtime for Xamarin.iOS, Xamarin.Android, Xamarin.Mac, and the Mono desktop framework



ASP.NET Core

<https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-3.1&tabs=windows>



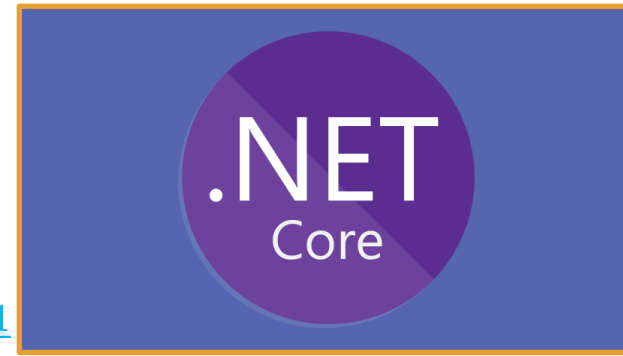
ASP.NET Core is a cross-platform, high-performance, open-source framework for building modern, cloud-based, Internet-connected applications. With **ASP.NET Core**, you can:

- Build web apps and services, IoT apps, and mobile backends.
- Use your favorite development tools on Windows, macOS, and Linux.
- Deploy to the cloud or on-premises.
- Run on .NET Core or .NET Framework

ASP.NET Core is a redesign of ASP.NET 4.x, with architectural changes that result in a leaner, more modular framework.

ASP.NET Core - Benefits

<https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1>



- Integrated testing, *web UI*, and *web APIs*.
- Develop and run on Windows, macOS, and Linux.
- Support for hosting *Remote Procedure Call (RPC)* services using *gRPC*.
- A cloud-ready, environment-based configuration system.
- Built-in *dependency injection*.
- A lightweight, modular *HTTP request pipeline*.
- Ability to host on *Docker*, *IIS*, *Kestrel* and many more.
- **ASP.NET Core** integrates with client-side frameworks and libraries like *Angular*, *React*, and *Bootstrap*.

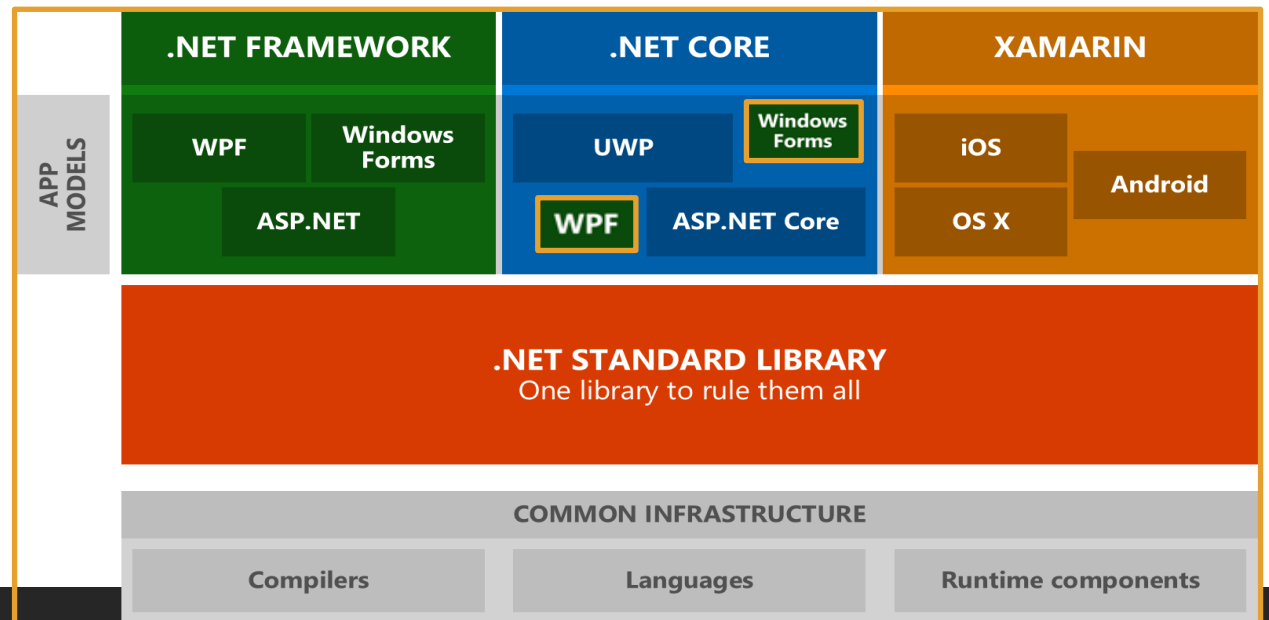
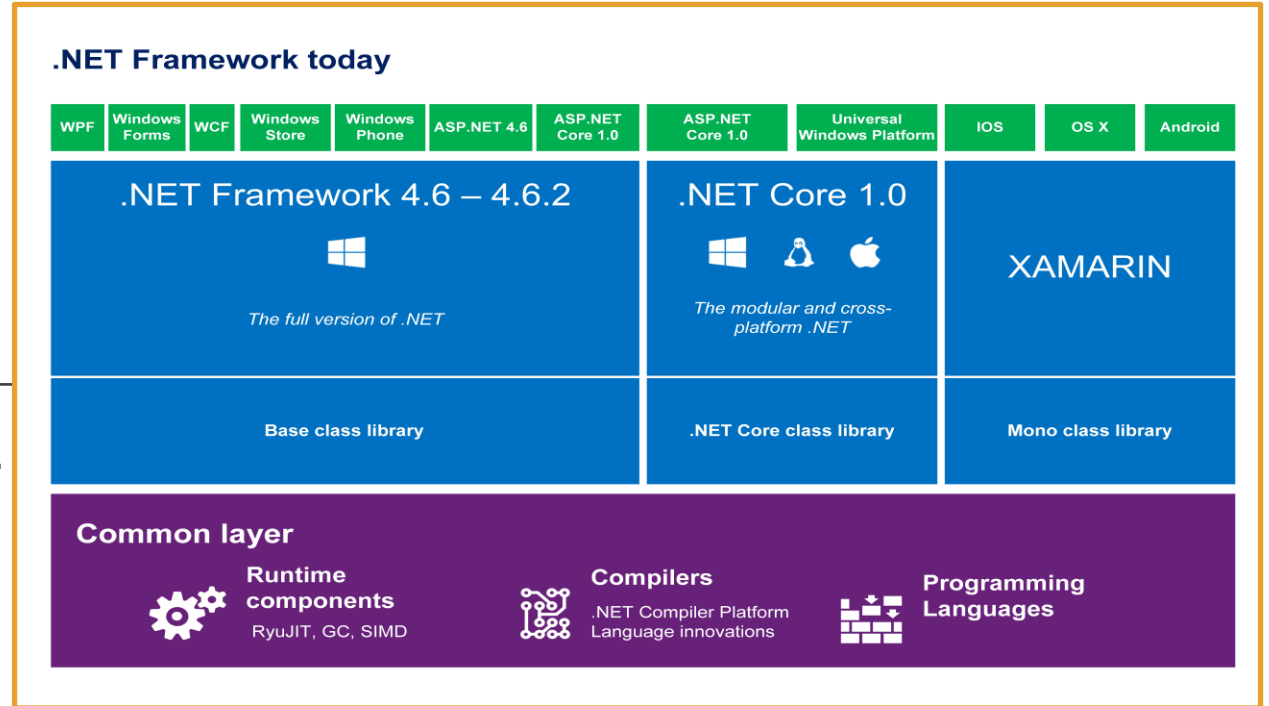
Mono(extra)

<https://docs.microsoft.com/en-us/dotnet/standard/components#mono>
<https://www.mono-project.com/docs/about-mono/>

- Mono is focused primarily on a small footprint.
- It's mainly used when a small runtime is required.
- Mono powers Xamarin applications on

Android macOS iOS tvOS watchOS

- Mono powers games built using the Unity engine.
- It supports all .NET Standard versions.
- It's used to run .NET applications that rely on Unix.
- Is used with a Just-In-Time compiler.
- Features a full static compiler (ahead-of-time compilation) used on platforms like iOS.



Xamarin (extra)

<https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin>

- Xamarin is an open-source platform for building applications for iOS, Android, and Windows with .NET.
- Xamarin is an abstraction layer that manages communication of shared code with underlying platform code.
- Xamarin runs in a managed environment that provides memory allocation and garbage collection.
- Xamarin enables developers to share an average of 80% of their application across platforms.
- Xamarin applications can be written on PC or Mac and compile into native application packages



Xamarin (extra)

<https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin>

Xamarin allows you to create native UI on each platform and write business logic in C# that is shared across platforms. In most cases, 80% of application code is sharable using Xamarin.

Xamarin is built on top of Mono. Mono runs on most platforms including Linux, Unix, FreeBSD, and macOS. The Mono execution environment automatically handles tasks such as memory allocation, garbage collection and interoperability with underlying platforms.

Xamarin is for developers with the following goals:

- Share code, test and business logic across platforms.
- Write cross-platform applications in C# with Visual Studio.

