**.Net and .Net core vs .Net framework**

**What is .Net?**

.Net is a soft ware platform which was introduced by Microsoft in the year 2002.

Microsoft is a leading organization in IT Industry was established by Billgates in USA.

**What we can do by using .Net**

By using .Net we can develop various types of software applications.

**How .Net is supporting to develop various types of software applications**

.Net is integrated with multiple technologies like below

1) Windows technologies – called windows forms

2) Web technologies- called ASP .Net

3) Web service technologies – called WCF ( Windows Communication Foundation)

**What do you mean by .Net language?**

A programming language which is supported by .Net platform is called as .Net language.

.Net supports for multiple programming languages which are called as .Net languages

They are

C#.Net, VB .Net, C++.Net, VC++.Net, VJ#.Net.... (160+)

**How to develop an application by using .Net**

If you want to develop any type of application by using .Net platform we need one .Net

language and one concern technologies.

E.g:- If you want to develop web application by using .Net we need one .Net language

called C#.Net & we need .Net web technology called ASP.Net.

**NOTE**: - NET

Stands for network enabled technology.

History of .Net:-

Before .Net i.e from 1990-98 under Micro Soft family we had 3 popular technologies

1) VB ( Visual Basic)

2) ASP ( Active Server Pages)

3) VC ++ ( Visual Studio ++)

These technologies are having the some of the draw backs like below

**1) Platform dependency: -These technologies will work only on windows operating system.**

**2) Not supporting to develop multi type of applications**

\*Using VB & VC++ we can develop only windows application.

\*Using ASP we can develop any web applications

\*At the same time in the year 1995 Sun Micro Systems has introduced a new software platform

called “JAVA”. Java is a platform independent technology means it will works on all Operating

Systems.

\*Java supports to develop various types of software applications like Windows applications, Web

applications & Web services.

\*Due to the above advantages in Java, Microsoft existing technologies users are attracting

towards java.

To overcome this in the year 1998, in a meeting called PDC (Professionals Developers

Conference) Microsoft has announced about .Net first time.

**In this meeting Microsoft has announced about .Net like below**

\* .Net will be platform independent technology

\* .Net will be supporting to develop various type of software applications

\*.Net will support for multiple programming languages.

From 1998 to 2002 Microsoft R&D team has designed .Net and finally they have released 1st

version of .Net i.e .Net 1.0 in the year 2002.

Versions of .Net:-

\* 1st version released in the year 2002--- .Net 1.0

\* 2nd version released in the year 2003--- .Net 1.1

\* 3rd version released in the year 2005--- .Net 2.0

\* 4th version released in the year 2006--- .Net 3.0

(WCF, Windows Communication Foundation & WPF: Windows Presentation Foundation)

\* 5th version released in the year 2008--- .Net 3.5 ( LINQ- Language integrated Query)

\* 6th version released in the year 2010--- .Net 4.0

\* 7th version released in the year 2012--- .Net 4.5

**CLI specification describes the following four aspects:**

1.The Common Language Specification (CLS): A set of base rules to which any language targeting the CLI should conform to interoperate to other CLS-compliant languages. The CLS rules define a subset of the Common Type System.

2.The Common Type System (CTS): A set of data types and operations that are shared by all CTS-compliant programming languages. According to this all “.NET” Languages has to adopt the rule “Uniform Data Type Structure” i.e., similar data types must be same in size in all Languages of .NET.

3.The Metadata: Information about program structure is language - independent, so that it can be referenced between languages and tools’, making it easy to work with code written in a language the developer is not aware.

4.The Virtual Execution System (VES): The VES loads and executes CLI-compatible programs. All compatible .NET languages compile to Common Intermediate Language (CIL), which is an intermediate code that is abstracted from the platform hardware. When the code is executed, the platform specific VES will compile the CIL to the machine language according to the specific hardware and operating system.

C# Source Code => Compiled by using C# Compiler => CIL Code

VB Source Code => Compiled by using VB Compiler => CIL Code

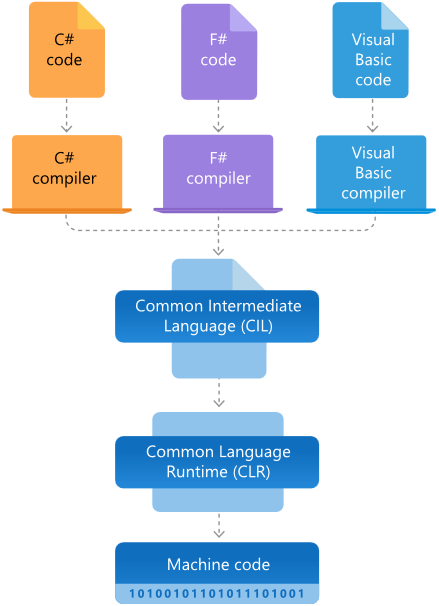
F# Source Code => Compiled by using F# Compiler => CIL Code

**What is .NET Framework and .NET Core?**

**Ans:** .NET is a developer platform made up of tools, programming languages, and libraries for building many different types of applications. There are various implementations of .NET, and each implementation allows .NET code to execute in different places - Linux, macOS, Windows, iOS, Android, and many more. Various implementations of the .NET include:

1. **.NET Framework:** it is the original implementation of .NET and it supports running websites, services, desktop apps, and more on Windows.
2. **.NET Core:** it is a cross-platform implementation for running websites, services, and console apps on Windows, Linux, and macOS.
3. **Xamarin/Mono:** it is a .NET implementation for running apps on all the major mobile operating systems, including iOS and Android.
4. The Class Library provides a set of APIs and types for common functionality. It provides types for strings, dates, numbers, etc. The Class Library includes APIs for reading and writing files, connecting to databases, drawing, and more.
5. The Common Language Runtime (CLR) is the heart of .NET Framework and the execution engine that handles running applications. It provides services like thread management, garbage collection, type-safety, exception handling, and more.

**Architecture of .NET Framework CLR:** .NET applications can be written in any .NET Language like C#, F#, or Visual Basic. Source Code we write by using some .NET Language is compiled into a language-agnostic Common Intermediate Language (CIL) and the compiled code is stored as assemblies (files with a “.dll” or “.exe” extension). When we run the applications, CLR takes the assemblies and uses a just-in-time compiler (JIT) to turn it into machine code that can execute on the specific architecture of the computer it is running on.



**What is .NET Framework used for?**

**Ans:** .NET Framework is used to create and run software applications. .NET apps can run on many operating systems, using different implementations of .NET. .NET Framework is used for running .NET apps on Windows.

**Who uses .NET Framework?**

**Ans:** Software developers and the users of their applications both use .NET Framework:

* Users need to install .NET Framework to run application built with the .NET Framework. In most cases, .NET Framework is already installed with Windows. If needed, you can download .NET Framework.
* Software developers use .NET Framework to build many different types of applications - websites, services, desktop apps, and more with Visual Studio. Visual Studio is an integrated development environment (IDE) that provides development productivity tools and debugging capabilities. See the .NET customer showcase for examples of what people is building with .NET.

**Why do I need .NET Framework?**

**Ans:** You need .NET Framework installed to run applications on Windows that were created using .NET Framework. It is already included in many versions of Windows. You only need to download and install .NET Framework if prompted to do so.

**How does .NET Framework work?**

**Ans:** .NET Framework applications can be written in many languages like C#, F#, or Visual Basic and compiled to Common Intermediate Language (CIL). The Common Language Runtime (CLR) runs .NET applications on a given machine, converting the CIL to machine code. See Architecture of .NET Framework for more info.

**What are the main components/features of .NET Framework?**

**Ans:** The two major components of .NET Framework are the Common Language Runtime (CLR) and the .NET Framework Class Library. The CLR is the execution engine that handles running applications. The Class Library provides a set of APIs and types for common functionality.

**How many versions do we have for .NET Framework?**

**Ans:** There are multiple versions of .NET Framework, but each new version adds new features but retains features from previous versions. List of .NET Framework Versions:

|  |  |  |  |
| --- | --- | --- | --- |
| .NET Framework 1.0 | .NET Framework 1.1 | .NET Framework 2.0 | .NET Framework 3.0 |
| .NET Framework 3.5 | .NET Framework 4 | .NET Framework 4.5 | .NET Framework 4.5.1 |
| .NET Framework 4.5.2 | .NET Framework 4.6 | .NET Framework 4.6.1 | .NET Framework 4.6.2 |
| .NET Framework 4.7 | .NET Framework 4.7.1 | .NET Framework 4.7.2 | .NET Framework 4.8 |

**Architecture of .NET Core:** The two main components of .NET Core are CoreCLR and CoreFX, respectively, which are comparable to the Common Language Runtime (CLR) and the Framework Class Library (FCL) of the .NET Framework's Common Language Infrastructure (CLI) implementation.

1. CoreFX is the foundational class libraries for .NET Core. It includes types for collections, file systems, console, JSON, XML and many others.
2. CoreCLR is the .NET execution engine in .NET Core, performing functions such as garbage collection and compilation to machine code. As a CLI implementation of Virtual Execution System (VES), CoreCLR is a complete runtime and virtual machine for managed execution of .NET programs and includes a just-in-time compiler called RyuJIT.

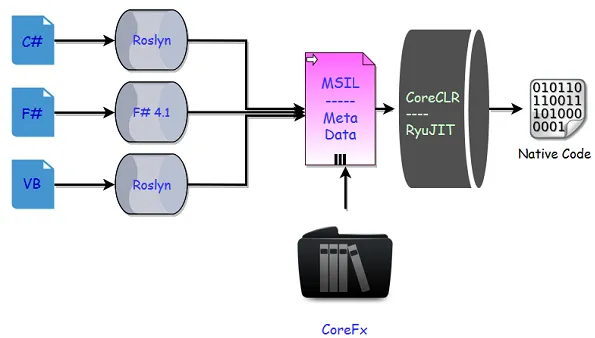
**Note:** .NET Core releases have a single product version, that is, there is no separate CLR version.

**What is CoreFX?**

**Ans:**CoreFX, also referred to as the Unified Base Class Library, consists of the basic and fundamental classed that form the core of the .Net Core platform. These set of libraries comprise the System. (and to a limited extent Microsoft.) namespaces. Majority of the .NET Core APIs are also available in the .NET Framework, so you can think of CoreFX as an extension of the .NET Framework Class Library.

**What is CoreCLR?**

**Ans:**CoreCLR is the .NET execution engine in .NET Core which is a complete runtime and virtual machine for managed execution of .NET programs and includes a just-in-time compiler called RyuJIT, performing functions such as garbage collection and compilation to machine code. CoreCLR is built from the same code base of the Framework CLR.



**What is .NET Core?**

**Ans:** The .NET Core platform is a new .NET stack that is optimized for open source development. .NET Core has two major components. It includes a runtime that is built from the same codebase as the .NET Framework CLR. The .NET Core runtime includes the same GC and JIT (RyuJIT), but doesn’t include features like Application Domains or Code Access Security. .NET Core also includes the base class libraries. These libraries are the same code as the .NET Framework class libraries, but have been factored to enable to ship as smaller set of libraries. .NET Core refers to several technologies including ASP.NET Core, Entity Framework Core, and more.

**What are the characteristics of .NET Core?**

**Ans:** .NET Core has the following characteristics:

* **Cross platform:** Runs on Windows, macOS, and Linux operating systems.
* **Open source:** The .NET Core framework is open source, using MIT and Apache 2 licenses. .NET Core is a .NET Foundation project.
* **Modern:** It implements modern paradigms like asynchronous programming, no-copy patterns using struts’, and resource governance for containers.
* **Performance:** Delivers high performance with features like hardware intrinsics, tiered compilation, and Span<T>.
* **Consistent across environments:** Runs your code with the same behavior on multiple operating systems and architectures, including x64, x86, and ARM.
* **Command-line tools:** Includes easy-to-use command-line tools that can be used for local development and for continuous integration.
* **Flexible deployment:** You can include .NET Core in your app or install it side-by-side (user-wide or system-wide installations). Can be used with Docker containers.

**What is the difference between .NET Core and .NET Framework?**

**Ans:** .NET Core and .NET Framework share many of the same components and you can share code across the two. Some key differences include:

* .NET Core is cross-platform and runs on Linux, macOS, and Windows. .NET Framework only runs on Windows.
* .NET Core is open-source and accepts contributions from the community. The .NET Framework source code is available, but does not take direct contributions.
* The majority of .NET innovation happens in .NET Core.
* .NET Framework is included in Windows and automatically updated machine-wide by Windows Update. .NET Core is shipped independently.