Introducing C# and the .NET Framework

C# is a general-purpose, type-safe, object-oriented programming language. The goal of the language is programmer productivity. To this end, C# balances simplicity, expressiveness, and performance. The chief architect of the language since its first version is Anders Hejlsberg (creator of Turbo Pascal and architect of Delphi). The C# language is platform-neutral and works with a range of platform-specific compilers and frameworks, most notably the Microsoft .NET Framework for Windows.

C# is a rich implementation of the object-orientation paradigm, which includes encapsulation, inheritance, and polymorphism. Encapsulation means creating a boundary around an object, to separate its external (public) behavior from its internal (private) implementation details.

Memory Management

C# relies on the runtime to perform automatic memory management. The Common Language Runtime has a garbage collector that executes as part of your program, reclaiming memory for objects that are no longer referenced. This frees programmers from explicitly deallocating the memory for an object, eliminating the problem of incorrect pointers encountered in languages such as C + +. C# does not eliminate pointers: it merely makes them unnecessary for most programming tasks. For performance-critical hotspots and interoperability, pointers and explicit memory allocation is permitted in blocks that are marked unsafe.

Platform Support

Historically, C# was used almost entirely for writing code to run on Windows platforms. Recently, however, Microsoft and other companies have invested in other platforms, including Linux, macOS, iOS, and Android. Xamarin ™ allows cross-platform C# development for mobile applications, and Portable Class Libraries are becoming increasingly widespread. Microsoft’s ASP.NET Core is a cross-platform lightweight web hosting framework that can run either on the .NET Framework or on .NET Core, an open source cross-platform runtime.

C# and the CLR

C# depends on a runtime equipped with a host of features such as automatic memory management and exception handling. At the core of the Microsoft .NET Framework is the Common Language Runtime (CLR), which provides these runtime features. (The .NET Core and Xamarin frameworks provide similar runtimes.) The CLR is language-neutral, allowing developers to build applications in multiple languages (e.g., C#, F#, Visual Basic .NET, and Managed C + +). C# is one of several managed languages that get compiled into managed code. Managed code is represented in Intermediate Language or IL. The CLR converts the IL into the native code of the machine, such as X86 or X64, usually just prior to execution. This is referred to as Just-In-Time (JIT) compilation.

The CLR and .NET Framework

The .NET Framework consists of the CLR plus a vast set of libraries. The libraries consist of core libraries and applied libraries, which depend on the core libraries.

Other Frameworks

The Microsoft .NET Framework is the most expansive and mature framework, but runs only on Microsoft Windows (desktop/ server). Over the years, other frameworks have emerged to support other platforms. There are currently three major players besides the .NET Framework, all of which are currently owned by Microsoft:

Universal Windows Platform (UWP)

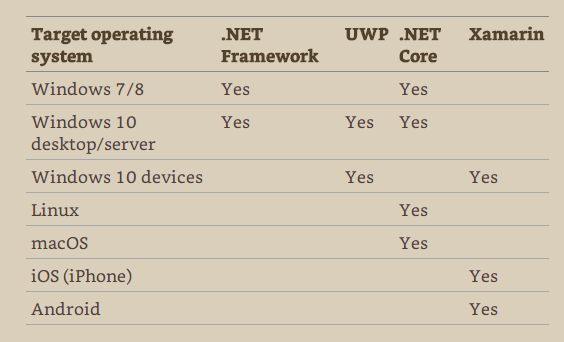
For writing Windows 10 Store Apps and for targeting Windows 10 devices (mobile, XBox, Surface Hub, Hololens). Your app runs in a sandbox to lessen the threat of malware, prohibiting operations such as reading or writing arbitrary files.

.NET Core with ASP.NET Core

An open source framework (originally based on a cut-down version of the .NET Framework) for writing easily deployable Internet apps and micro-services that run on Windows, macOS, and Linux. Unlike the .NET Framework, .NET Core can be packaged with the web application and xcopy-deployed (self-contained deployment).

Xamarin

For writing mobile apps that target iOS, Android, and Windows Mobile. The Xamarin company was purchased by Microsoft in 2016.



The four major frameworks differ in the platforms they support, the libraries that sit on top, and their intended uses.

\*It is even possible to directly leverage this commonality by writing class libraries that work across all four frameworks -> .Net Standard library!

Cool new things about C# 7.0

-int million = 1\_000\_000; The underscores improve readability and are ignored by the compiler

-create variables on the fly:

void Foo (object x)

{

if (x is string s)

Console.WriteLine (s.Length);

}