31927 32998:
Application Development with .NET

Week-1 Lecture

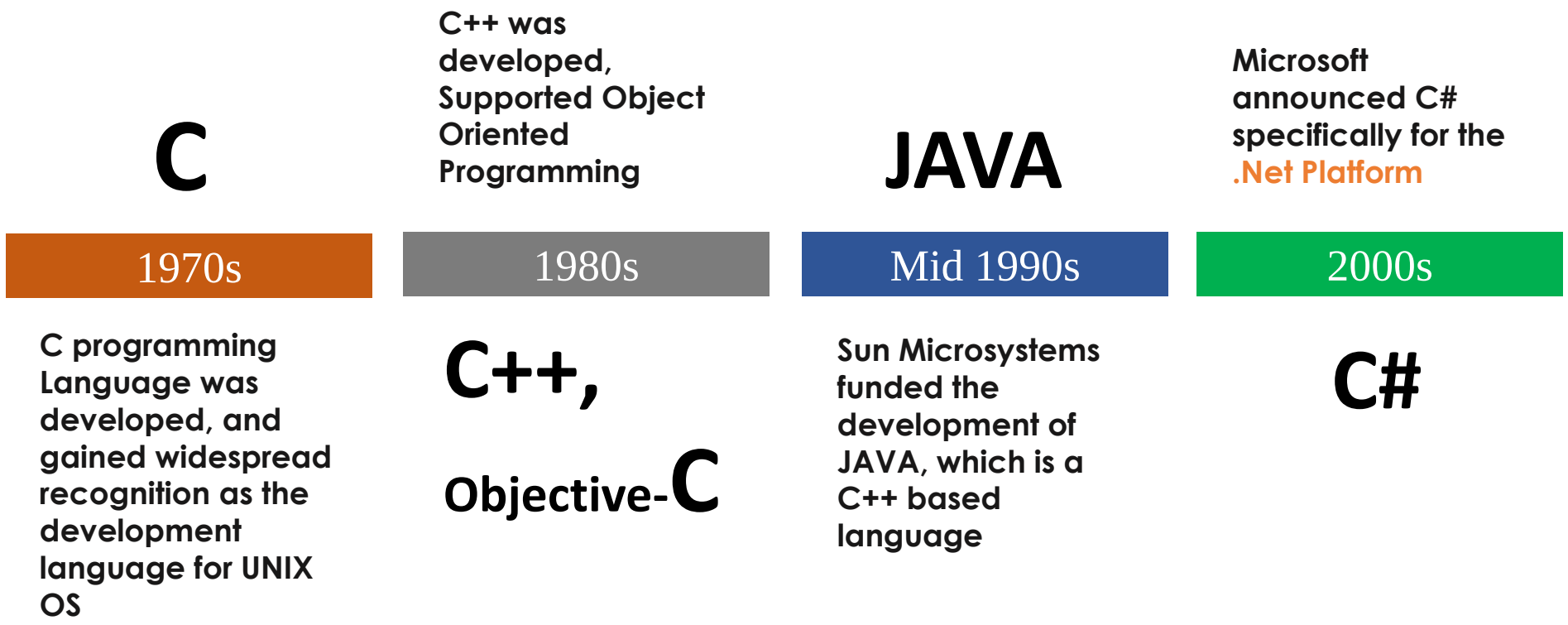
Introduction to Application Development with .Net and .Net Overview

Outline

In this lecture we introduce the following topics

1. The History and basics
2. What is .Net, .Net Core, .Net Framework, Xamarin
3. CLR, FCL, Managed code, JIT
4. Visual Studio introduction
5. C# - First Program
6. Hello World Demo

The C# Language History



.Net History

Late 1980s	Early 1990s	1990s	2000s
DDE	Object linking and Embedding, used with Windows 3.1 -1992. It was build on top of DDE and specifically designed for compound documents. -limited scope -difficult to program	**COM, COM+, DCOM**	Microsoft introduced the .Net platform
Dynamic Data Exchange is one of the first method of inter-process communication in Windows, introduced in 1987	**OLE**	Dynamic Link Libraries, Active X control introduced. -Relatively easier to develop	**.Net**

Reference: https://www.codeproject.com/Articles/1821/Introduction-to-NET

What is .Net?

- .NET is a free (since Nov 2014), cross-platform, open source development platform for building many different types of applications.

- Key features:
  - Support for multiple languages,
  - Editors/IDEs, and libraries to build for web, mobile, desktop, gaming, and IoT.
  - Cross Platform: .Net Core, .Net Framework, Mono/Xamarin
  - Consistent API
  - Libraries

- Languages: C#, F#, Visual Basic.

Reference: https://www.microsoft.com/net/learn/what-is-dotnet ,
https://news.microsoft.com/2014/11/12/microsoft-takes-net-open-source-and-cross-platform-adds-new-development-capabilities-with-visual-studio-2015-net-2015-and-visual-studio-online/

.Net Architecture and Component Stack

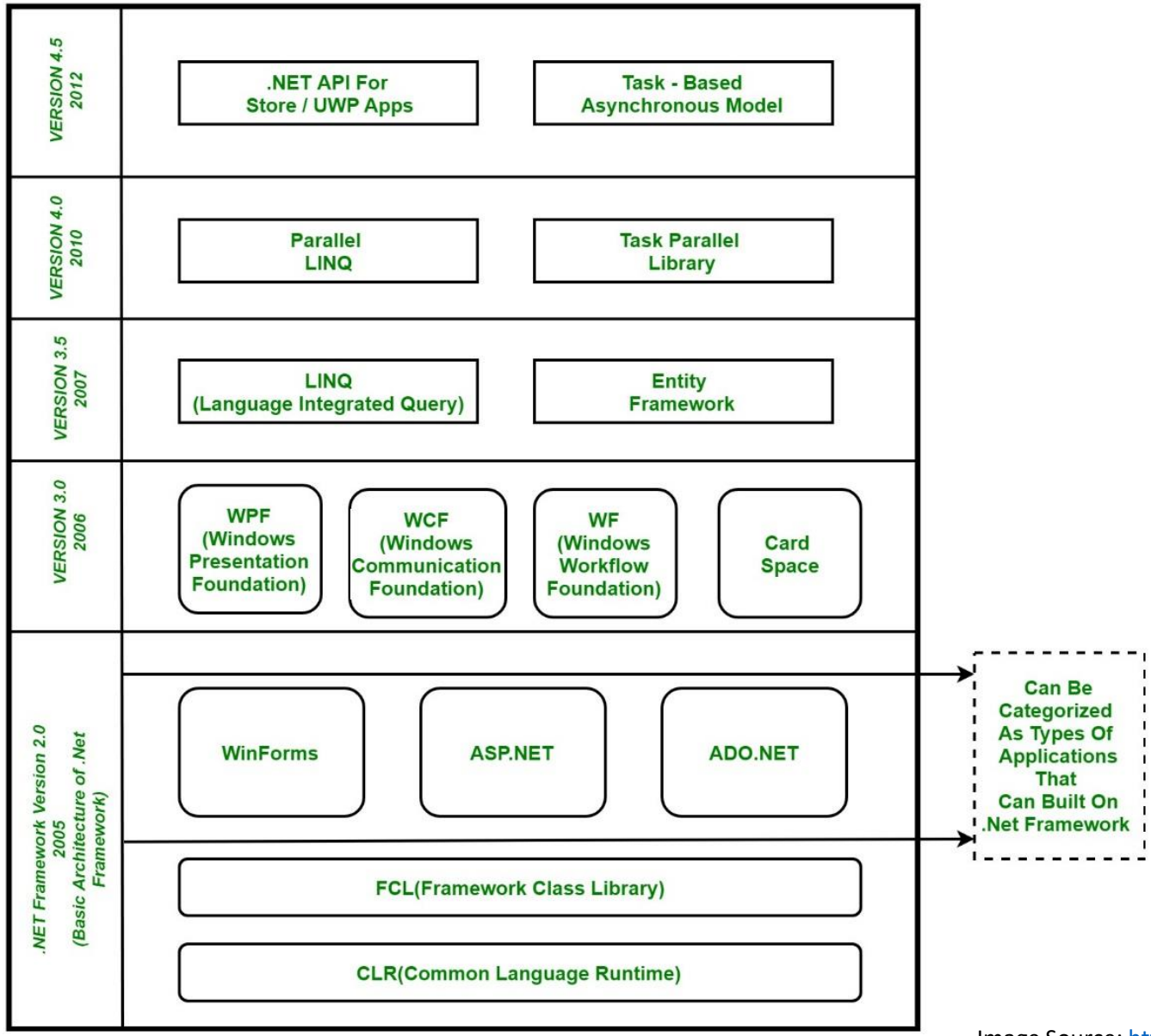


Image Source: https://www.geeksforgeeks.org/c-net-framework-basic-architecture-component-stack/

What is .Net Core?

- . NET Core is a general purpose development platform maintained by Microsoft and the .NET community on GitHub. It is Open Source (MIT license)
- It is cross-platform, supporting Windows, macOS and Linux, and can be used in device, cloud, and embedded/IoT scenarios.
- Composition:
  - .Net Runtime
  - Framework Libraries: basic data types, composition type and utilities.
  - SDK tools and language compilers
  - The ‘dotnet’ app host: which is used to launch .NET Core apps
- Languages supported: C#, F#, and Visual Basic.

Reference: https://docs.microsoft.com/en-us/dotnet/core/

What is .Net Framework?

- The .NET Framework is a development platform for building apps for web, Windows, Windows Phone, Windows Server, and Microsoft Azure.
- It consists of the **Common Language Runtime (CLR)** and the **.NET Framework class library**, which includes a broad range of functionality and support for many industry standards.
- The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services.

Reference: https://docs.microsoft.com/en-us/dotnet/framework/

What is .Net Framework?

- The .NET Framework provides many services:
  - Memory Management
  - Type and memory safety
  - Security
  - Networking
  - Application Deployment

- The .NET Framework comes with Windows, enabling you to run .NET Framework applications.

Reference: https://docs.microsoft.com/en-us/dotnet/framework/

.Net Core Vs .Net Framework

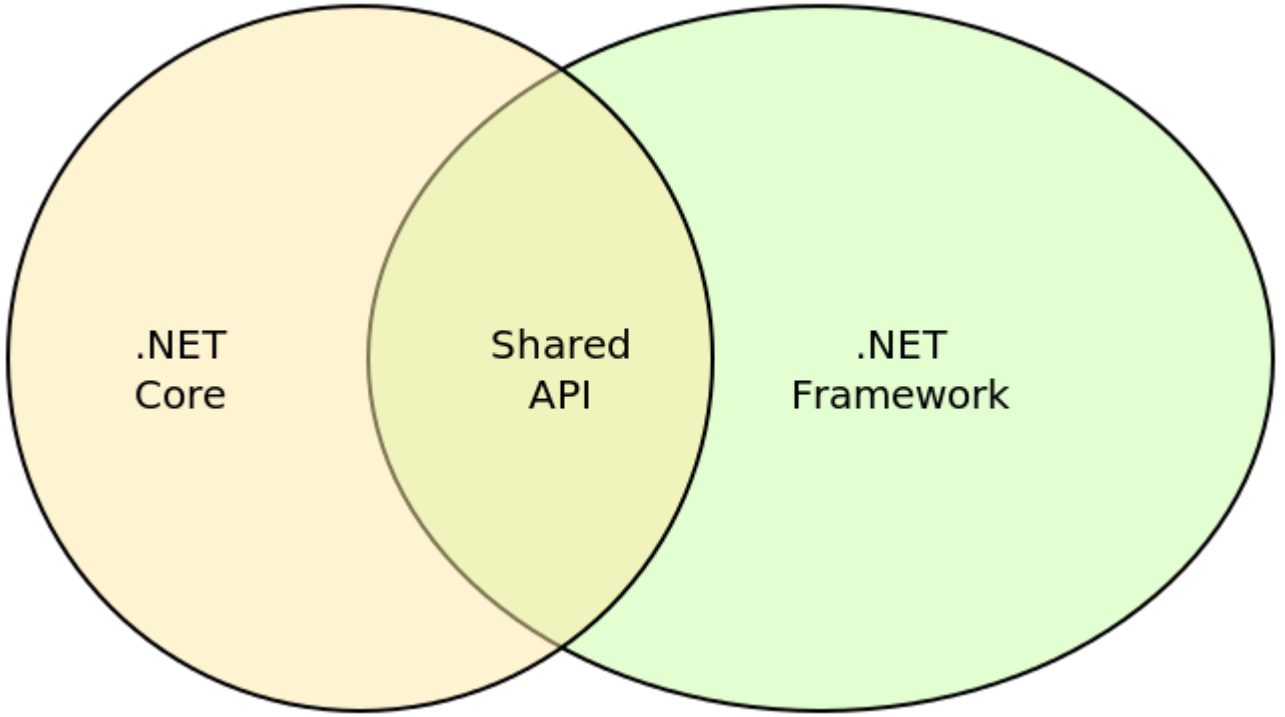


Image source: https://en.m.wikipedia.org/wiki/File:.NET_Framework-Core_relationship.svg

When to use .Net Core?

1. Cross platform requirements: When application needs to run across Windows, Mac, and Linux.
2. Microservices are used: A form of Service Oriented Architecture
3. Docker containers are used.
4. High performance and scalable systems required.
5. Multiple versions of .Net run in parallel.
6. CLI control required.

Reference: https://stackify.com/net-core-vs-net-framework/

When to use .Net Framework?

1. When using technologies not yet available in .Net Core, such as ASP.NET Web Forms, ASP.NET Web page applications, etc.

2. Using third-party libraries or packages not available in .Net core

3. Platform that does not support .Net Core: such as some of the Azure services.

Reference: https://stackify.com/net-core-vs-net-framework/

What is .Net Xamarin/Mono?

- It is a .NET implementation for running apps on all the major mobile operating systems such iOS, Android, or Windows Phone devices.

- It provides a cross-platform development solutions for mobile, tablets, and desktop applications.

- Check out Cross-Platform Samples at:

https://docs.microsoft.com/en-us/xamarin/cross-platform/samples/index

Reference: https://www.microsoft.com/net/learn/what-is-dotnet
https://docs.microsoft.com/en-us/xamarin/

The Common Language Runtime (CLR)

- CLR is the foundation of .Net Framework
- Code Management is the fundamental principle of Runtime.
- CLR is like an agent which manages:
  - Code execution
  - Providing core services, like memory management etc.
  - Enforcing type safety
- Advantages:
  - Language independent
  - Platform independent

Reference: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

The .Net Framework Class Library (FCL)

- It is a large library of tested, reusable code which developers can call from their own applications
- It is a collection of reusable classes, interfaces, and value types
- It provides the .core functionalities of the .Net Framework:
  - Basic Data types
  - Data structure implementation
  - Garbage collection
  - Data access and database connectivity
  - Network Communication
  - Support for GUI development, etc.

Reference: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

What is Managed Code?

- In short, Managed code is that whose execution is managed by a Runtime (e.g. CLR)
- CLR takes the managed code, compiles to machine code and executes it!
- Example: Managed code is written in one of the languages such as C#, Visual Basic, F#, etc. and run on top of .NET

- Example of Unmanaged code: Code written in C/C++, where the programmer have to take care of EVERYTHING!

Reference: https://docs.microsoft.com/en-us/dotnet/standard/managed-code

CLR, FCL, Custom Apps Relationship!

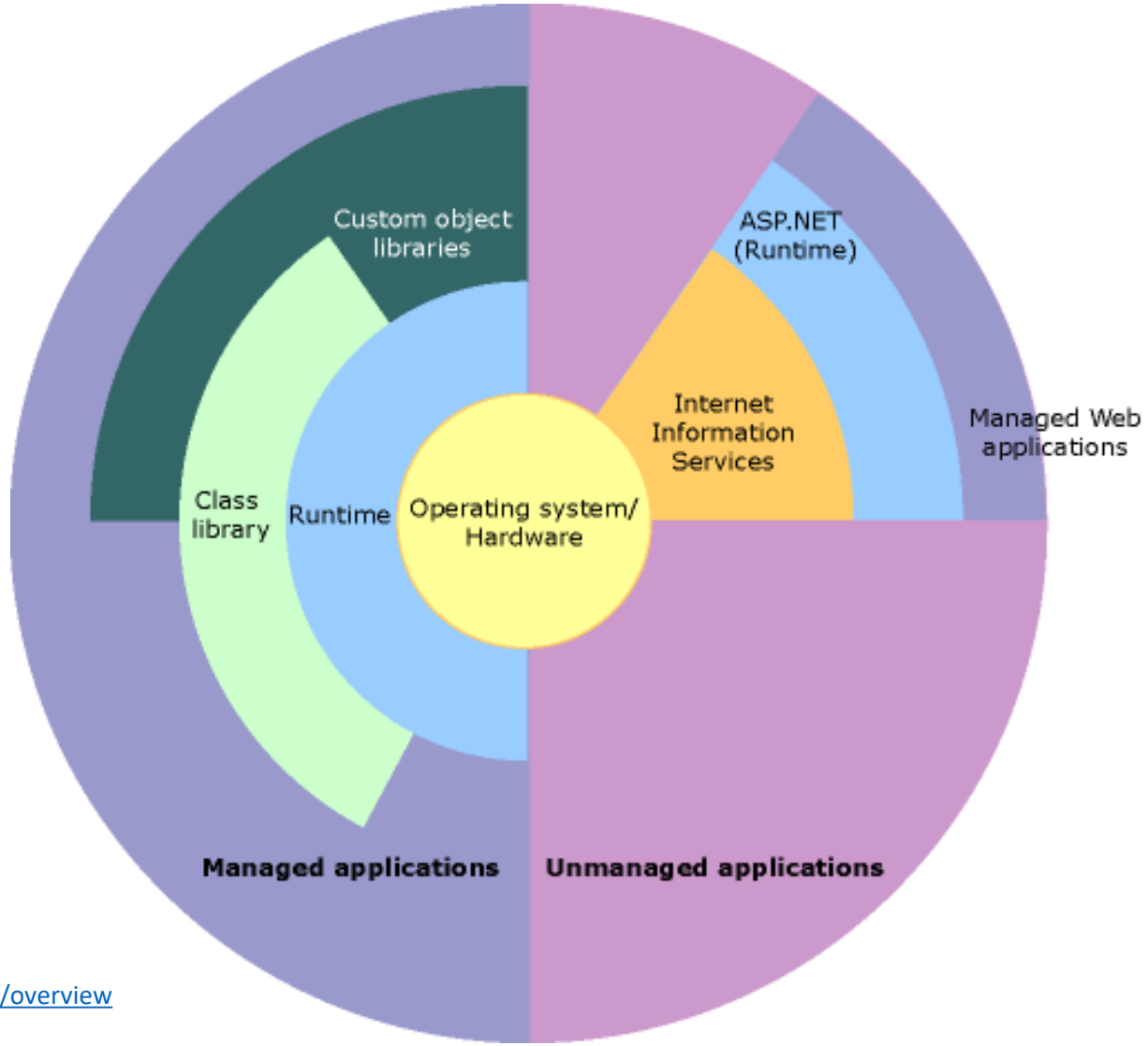
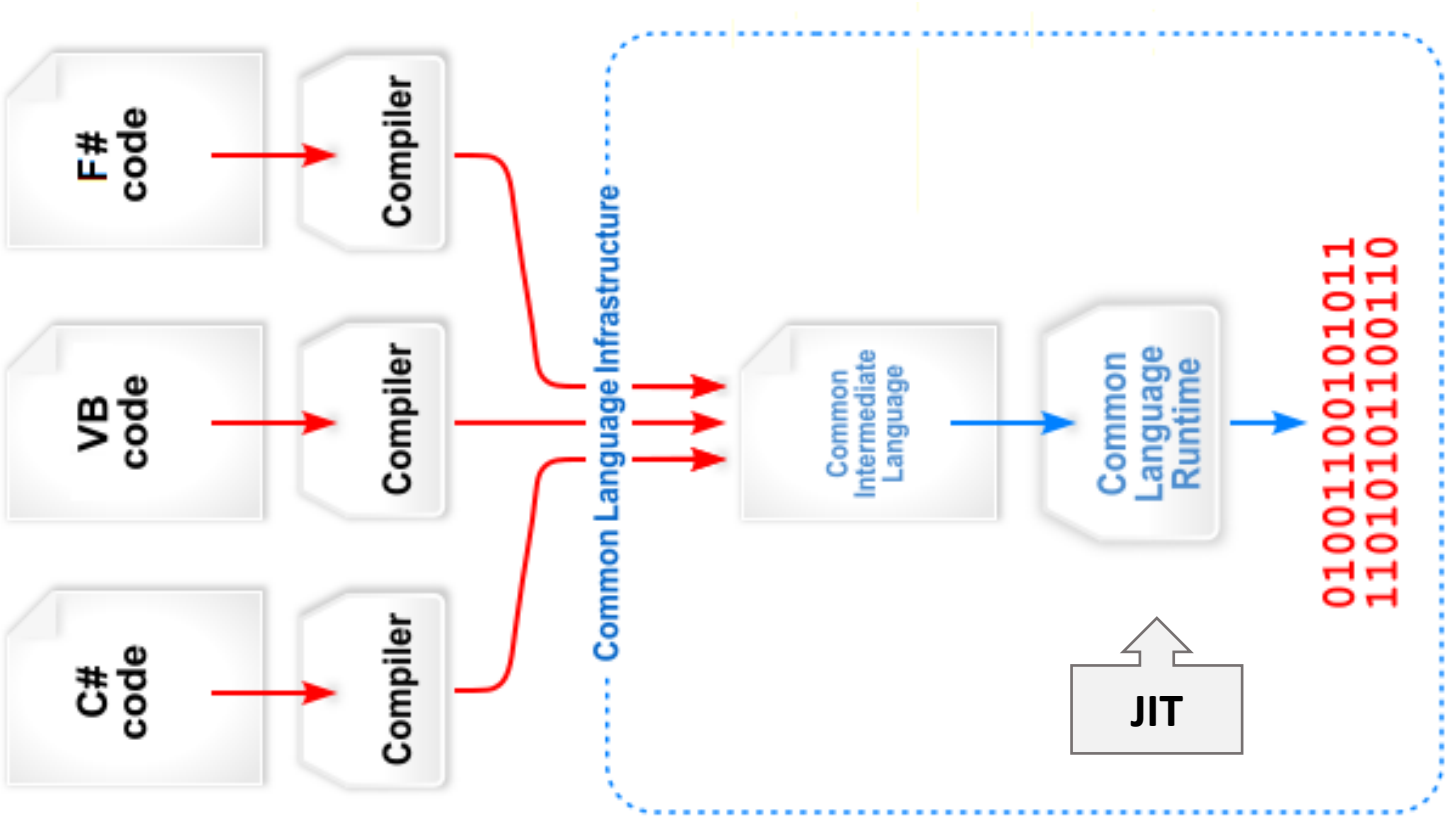


Image Source: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview

Just-in-Time (JIT) Compiling



- .Net Languages compile to a platform independent language called Common Intermediate Language (CIL)
- CLR Compiles CIL to machine readable code which can execute on the current machine/platform

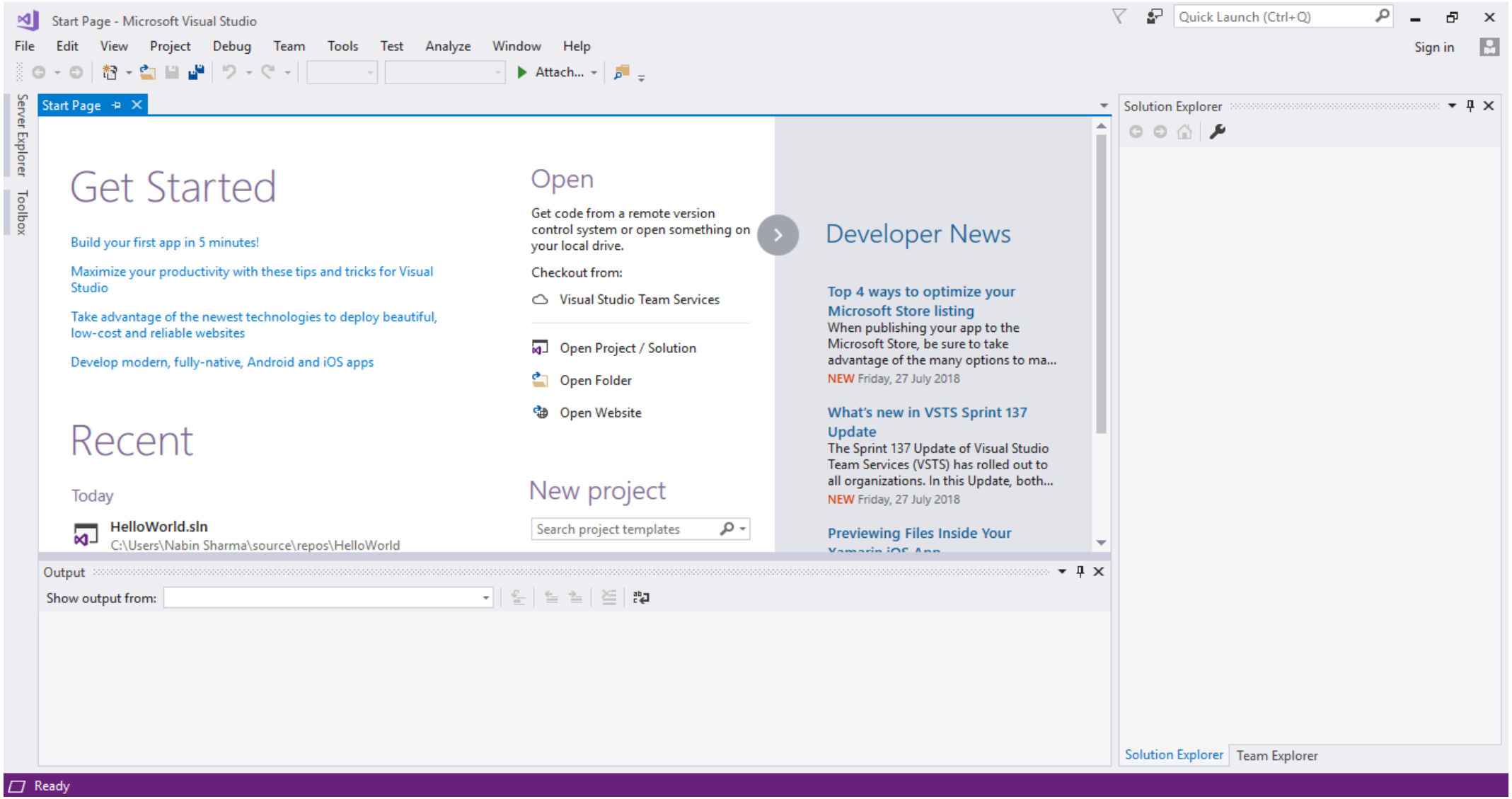
Image adapted from: https://en.wikipedia.org/wiki/File%3aOverview_of_the_Common_Language_Infrastructure.svg

Microsoft Visual Studio

- Visual Studio is an **Integrated Development Environment (IDE)** provided by Microsoft
- It is used to develop computer programs, Websites, Web Apps, Web Services, as well as mobile Apps.
- Includes:
  - Intuitive Code Editor
  - Supporting IntelliSense (the code completion component)
  - Integrated Debugger
  - Form Designer
  - Class Designer
  - Database Schema Designer, many more wonderful features!

Reference: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Visual Studio Community Edition 2017



Check: https://visualstudio.microsoft.com/downloads/

C#

- C# (pronounced "C sharp")
- It is a modern, Object oriented and type-safe language
- General purpose
- Curly brace and semicolon family {C; C++; Java…}
- Engineered from scratch for .NET
- Familiar programming constructs
- New mechanisms (delegates, attributes, LinQ, etc.)

First Program in C# - The Hello World!

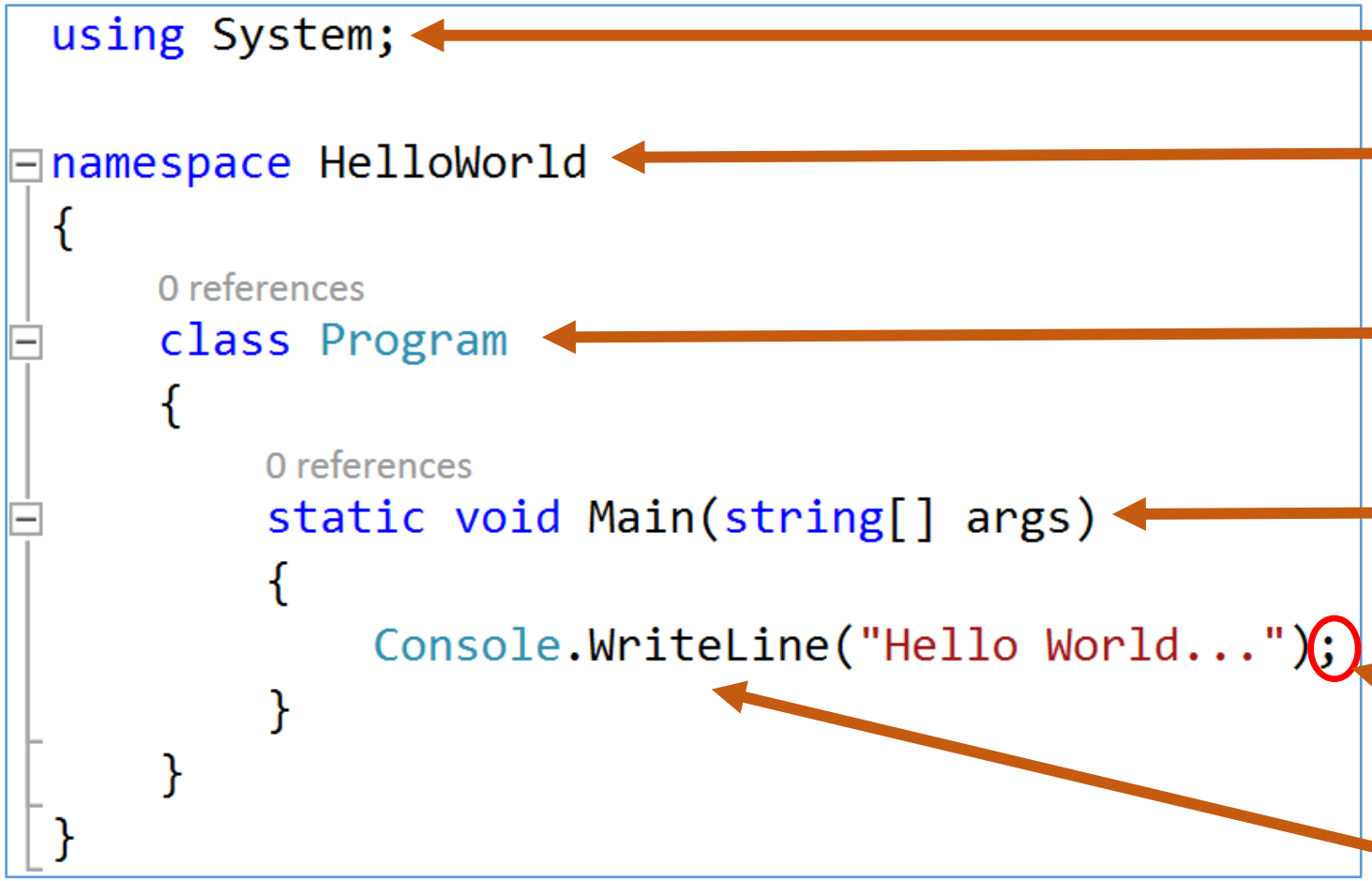
```
using System;

namespace HelloWorld
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World...");
        }
    }
}
```

Output:

C# - Some basics

- Namespace: It is a Keyword used to declare a scope. .Net uses namespace to organize its classes. It help to control the scope classes and methods in large projects.
- To access the required namespaces for a program, using directive is used. Using is a keyword.
- Most C# program starts with a section of using directive. It lists the namespaces which will be frequently used in the program.

First Program in C# - The Hello World!



System is a Namespace

HelloWorld is a new Namespace created

Program is the name of the class, and can be renamed!

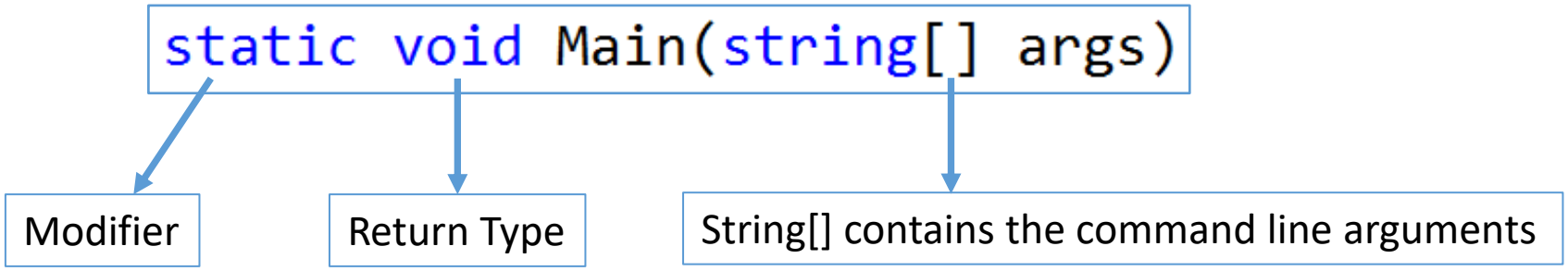
Main() is the entry point of a C# program! It is the first method which is invoked

End of Statement marker

WriteLine() writes String/data to the standard output stream, followed by a line terminator

Reference: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/main-and-command-args/

First Program in C# - The Hello World!



Why Main is static?

A static member doesn't belong to a specific object. Hence Main method can be used without creating an object!

Main needs to be static in order to allow it to be the entry point to a program.

Reference: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/static

The Hello World Demo …