

หน่วยที่ 2

แนะนำ .NET framework

เรื่องที่จะศึกษา

2.1 เหตุผลที่ควรศึกษา .NET framework

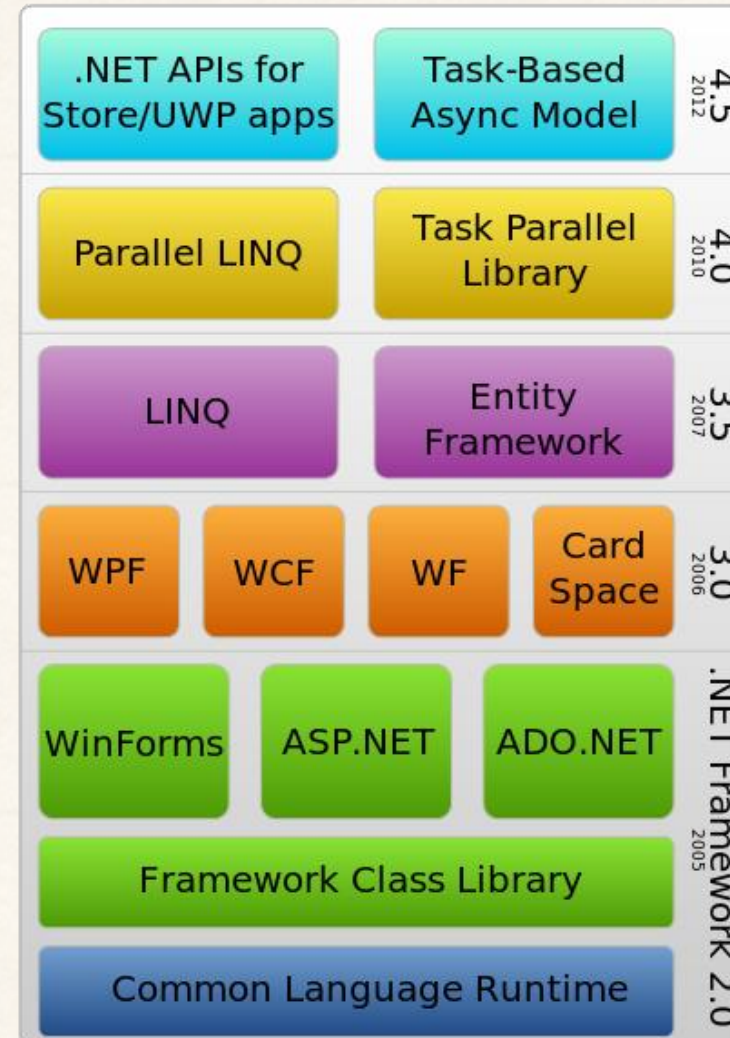
1 การพัฒนาอย่างต่อเนื่องและครอบคลุม

➤ .NET Framework

➤ .NET Core

➤ .NET 5

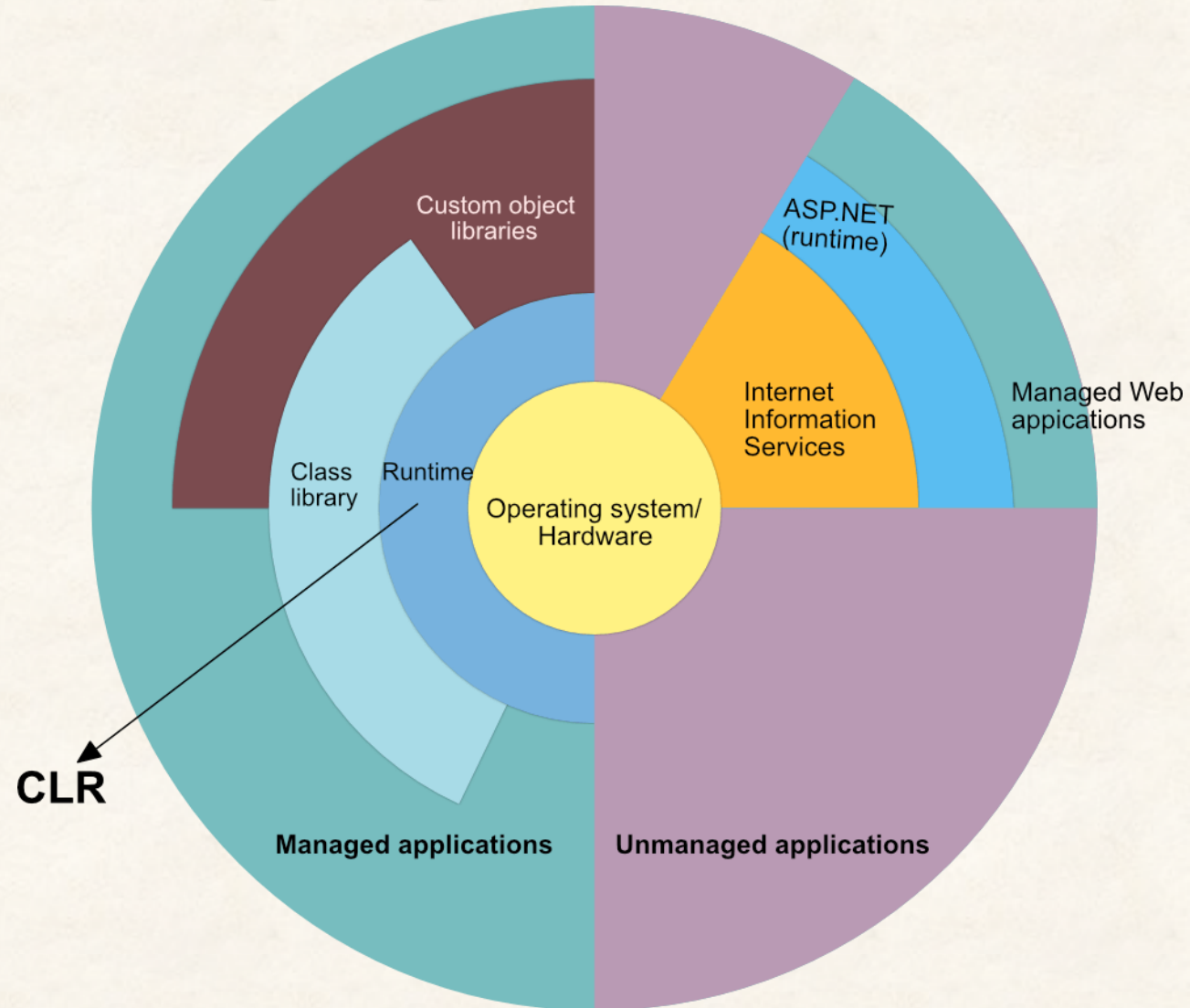
➤ .NET 6



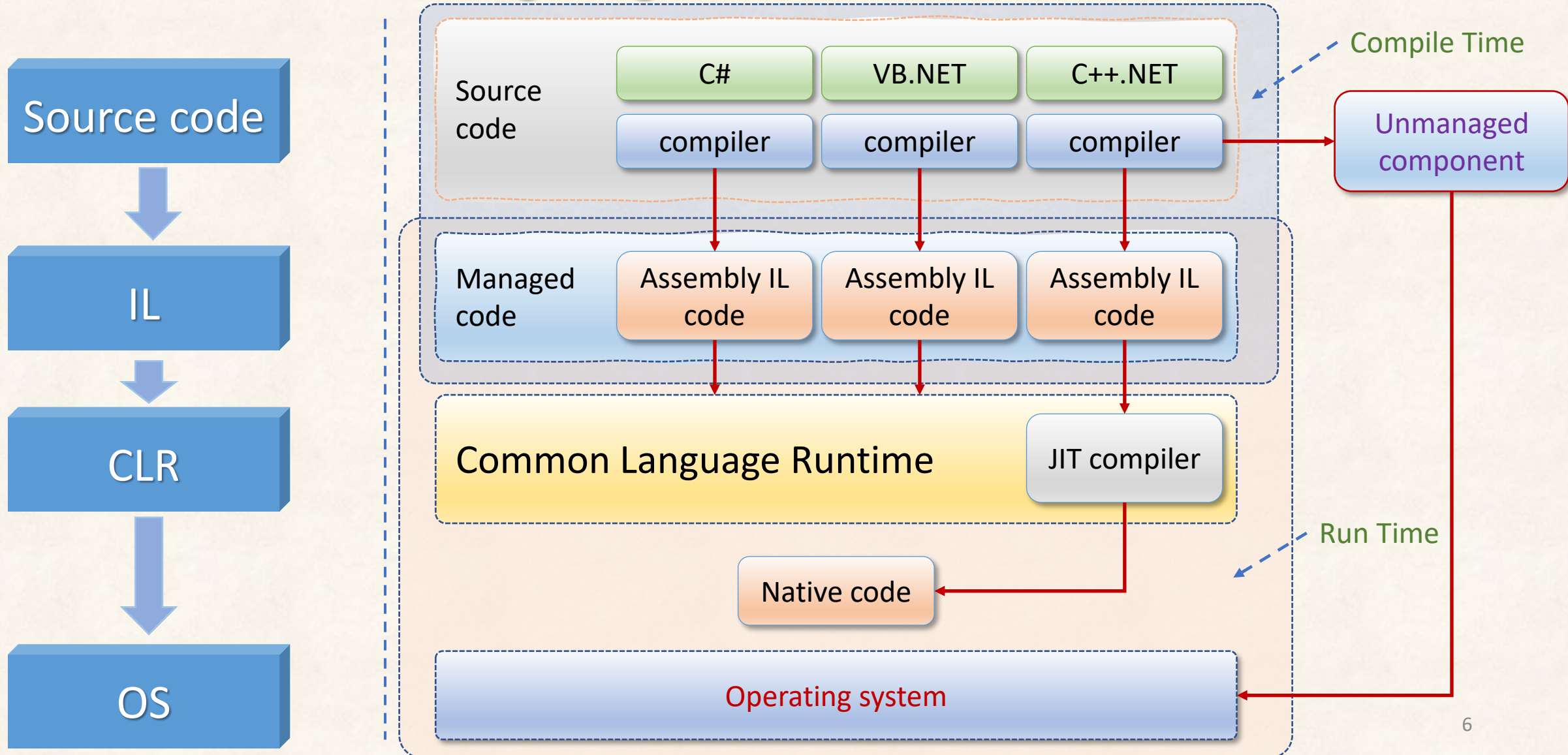
.NET Application Type

- Console Application
- Windows or Desktop Application
- Web Application
- NT Services - Service Control Manager Services
- Web services
- REST API
- Mobile applications
- Components/libraries
- Windows Store Applications

Common Language Runtime (CLR)



Common Language Runtime (CLR)



Common Language Runtime (CLR)

Entity
Frame-
work

ASP.NET

WCF

WPF

Win
Forms

Others

Base class library

CLR

Profiling & Debugging
APIs

JIT & NGEN

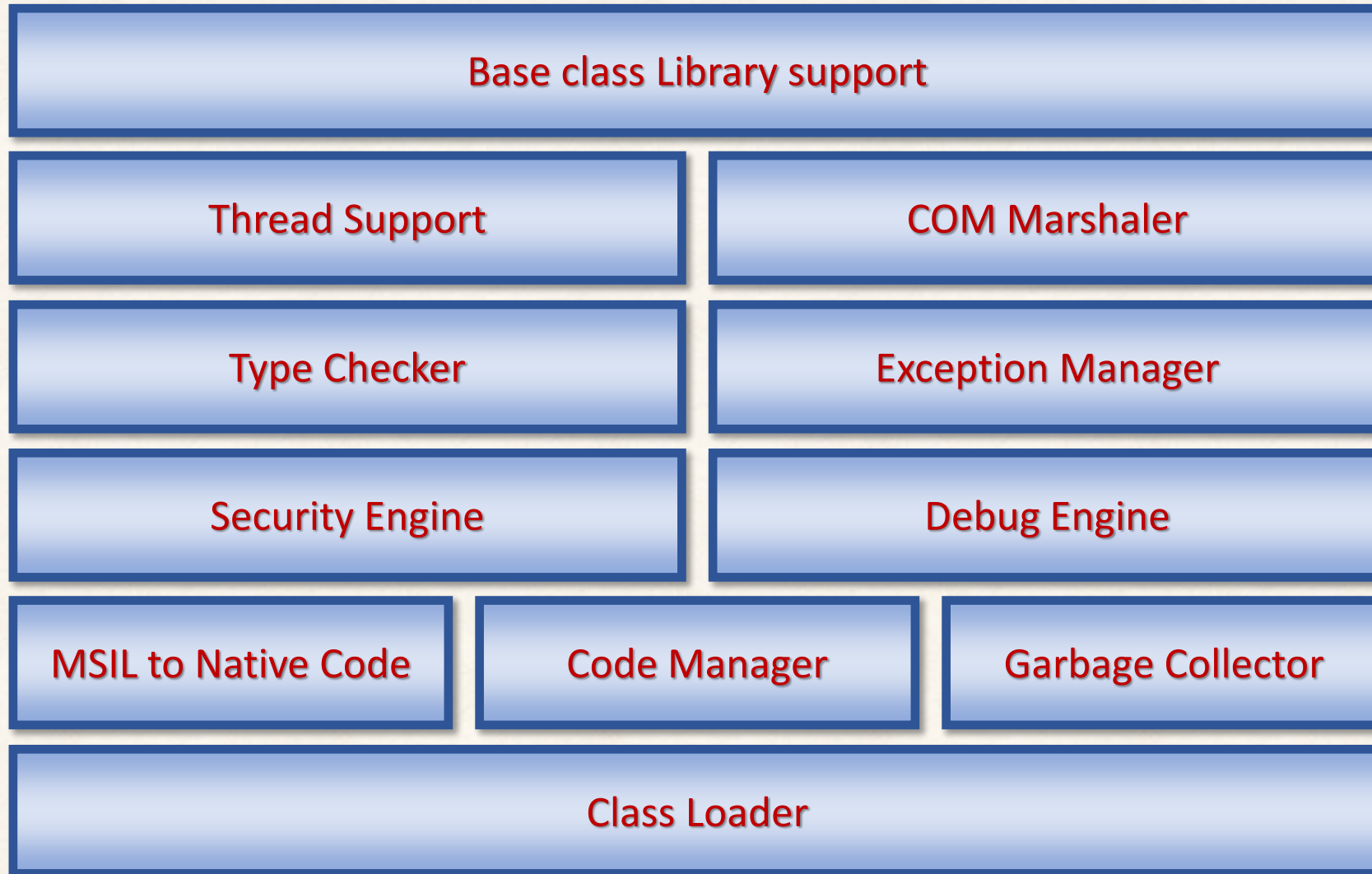
Garbage
Collector

Security
Model

Exception
Handling

Loader &
Binder

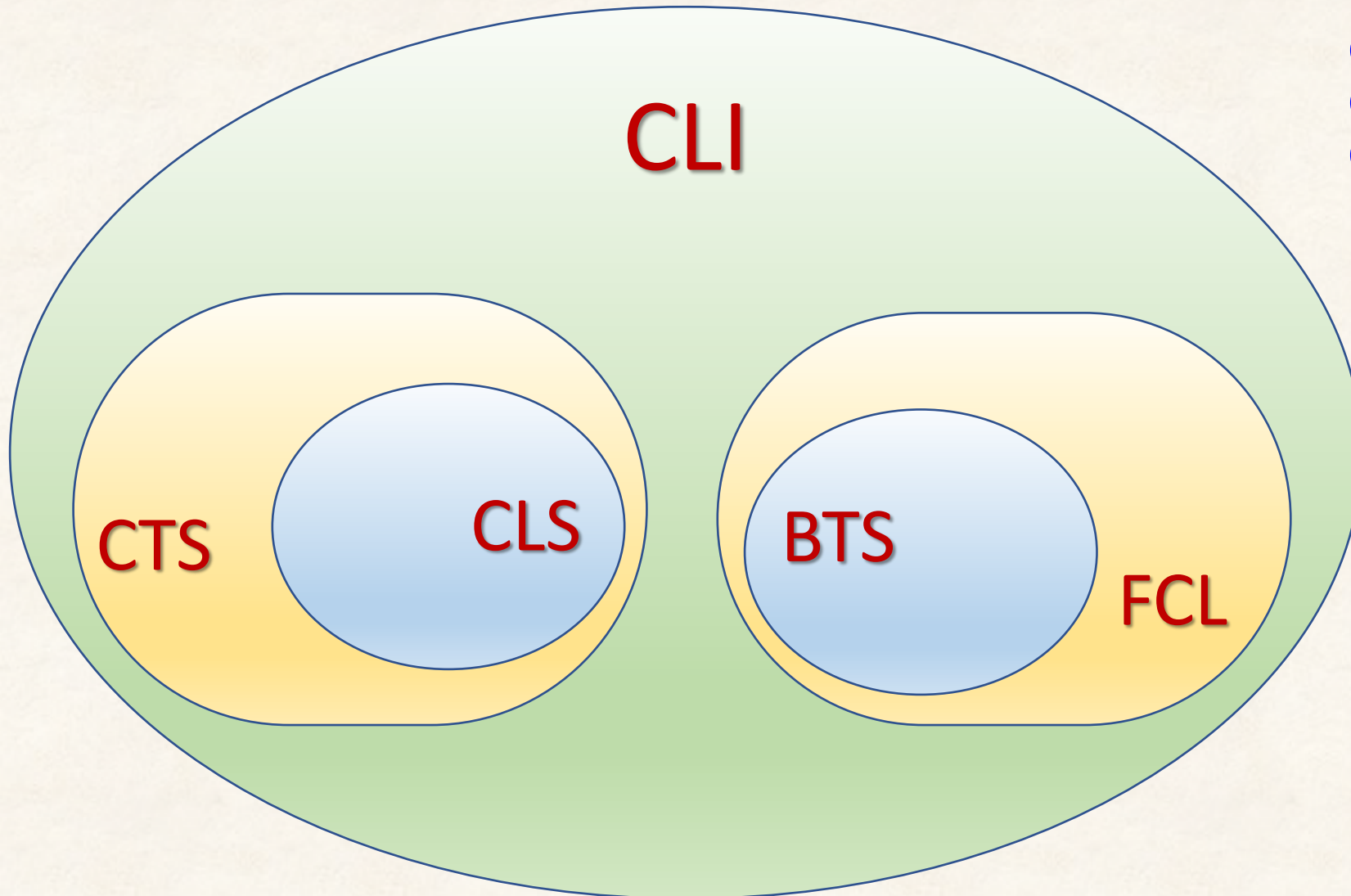
CLR Architecture



CLR Features

- Manages memory
- Thread execution support
- Code execution
- Code safety verification
- Compilation
- Code Security

Common Language Infrastructure (CLI)



CLI = Common Language Infrastructure

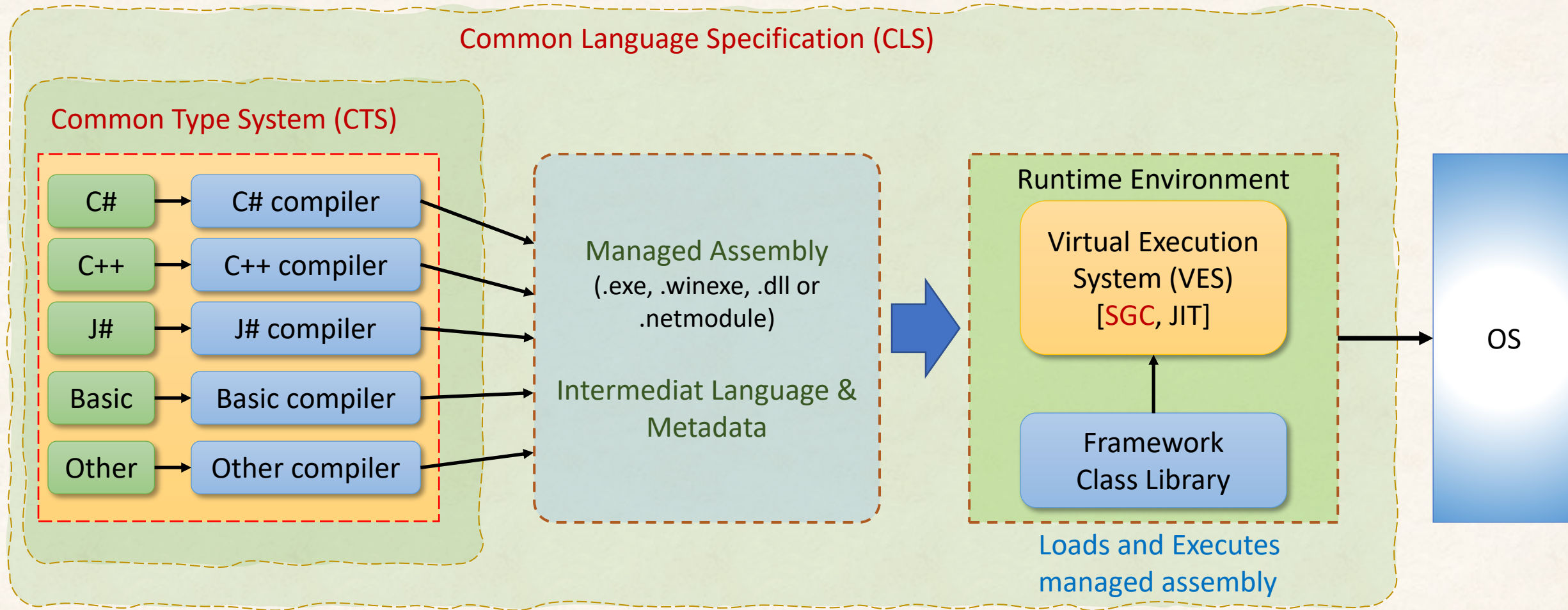
CTS = Common Type System

CLS = Common Language Specification

BCL = Base Class Library

FCL = Framework Class Library

Common Language Infrastructure (CLI)



SGC = Security-Garbage Collection

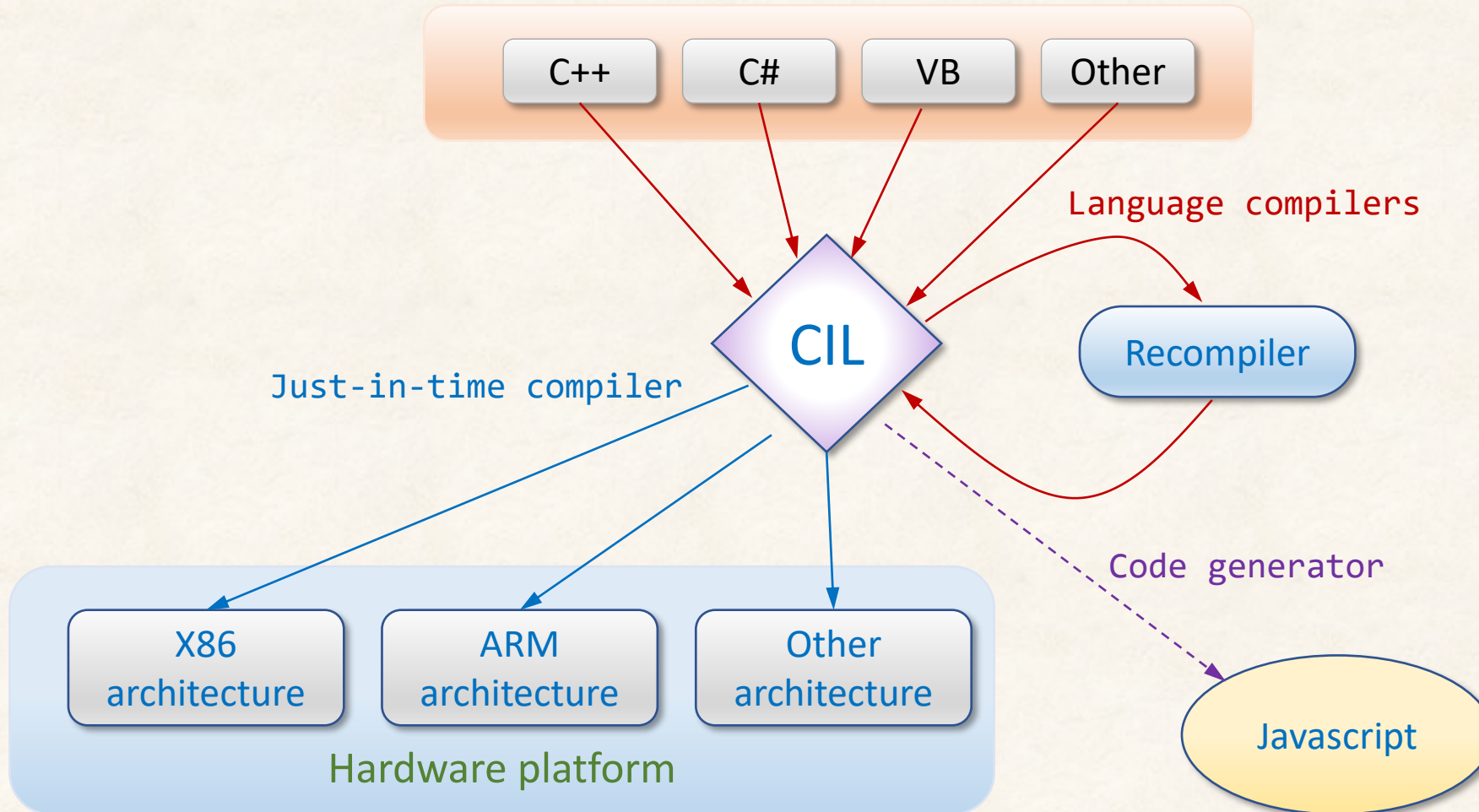
แนวทางค้นคว้าเพิ่มเติมสำหรับผู้สนใจ

- CLI = Common Language Infrastructure
- CTS = Common Type System
- CLS = Common Language Specification
- BCL = Base Class Library
- FCL = Framework Class Library

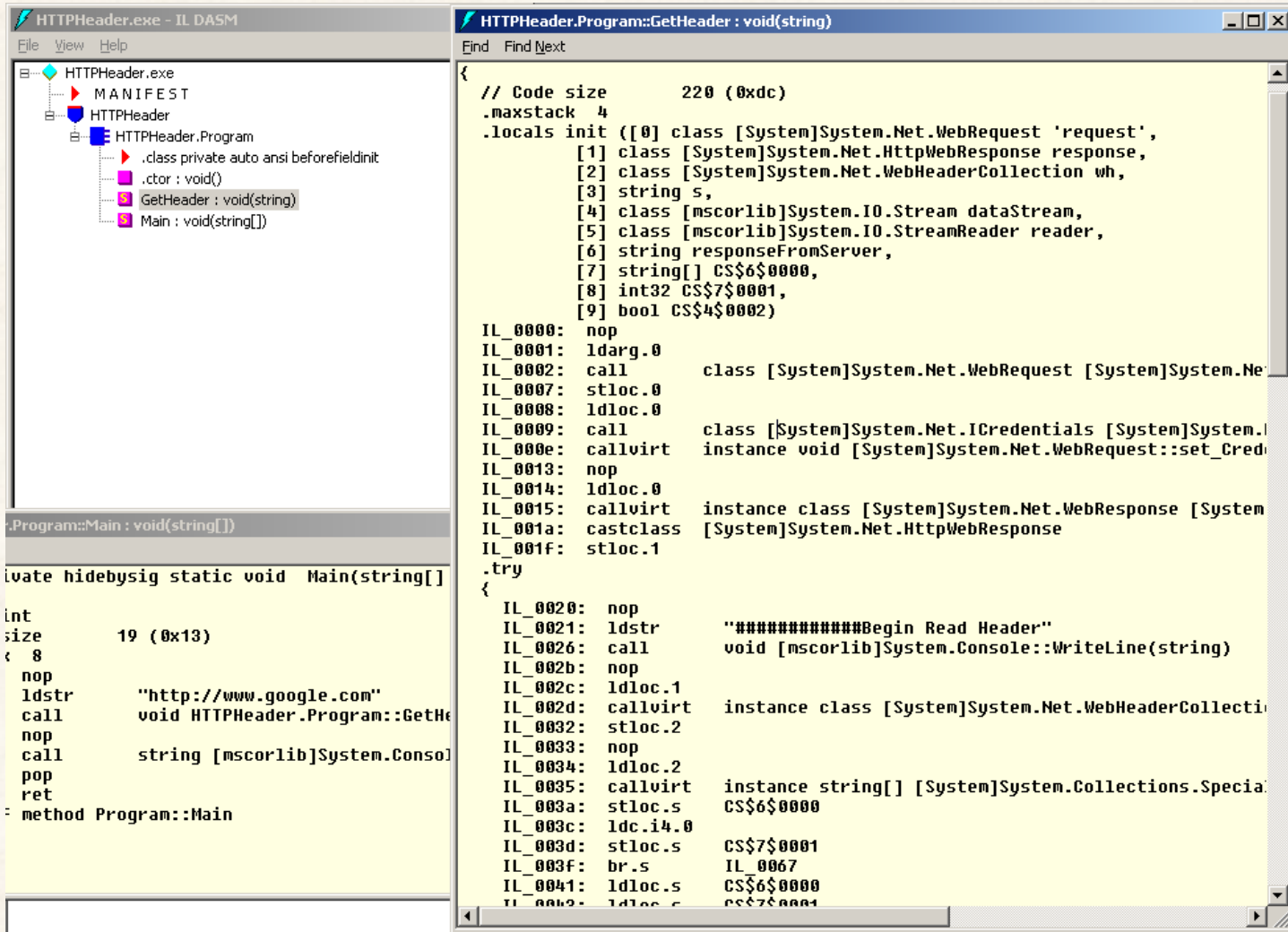
List of CLI languages

- Ada for .Net
- C#:
- C++/CLI
- Component Pascal
- Eiffel
- F#
- F*
- Fantom
- IronPython
- IronScheme
- Limnor Studio
- Oxygene
- PascalABC.NET
- PeachPie
- PowerBuilder
- Small Basic
- Silverfrost FTN95
- Swif
- Synergy DBL .NET
- Team Developer
- Visual Basic (VB.NET)
- Visual COBOL
- PowerShell
- XSharp

Common Intermediate Language (CIL)



Common Intermediate Language (CIL)



The screenshot displays the IL DASM tool interface. The left pane shows the assembly structure for HTTPHeader.exe, including the MANIFEST, HTTPHeader, and HTTPHeader.Program. The right pane shows the IL code for the GetHeader method, which is a void(string) method. The code includes local variable declarations for a WebRequest, response, WebHeaderCollection, string, dataStream, StreamReader, responseFromServer, and an array of strings. The IL code starts with a nop instruction, followed by ldarg.0, call, stloc.0, ldloc.0, call, callvirt, nop, ldloc.0, callvirt, castclass, and stloc.1. A try block follows, containing instructions like nop, ldstr, call, nop, ldloc.1, callvirt, stloc.2, nop, ldloc.2, callvirt, stloc.s, ldc.i4.0, stloc.s, br.s, ldloc.s, and ldloc.s.

```
HTTPHeader.exe - IL DASM
File View Help
B HTTPHeader.exe
  B MANIFEST
  B HTTPHeader
    B HTTPHeader.Program
      .class private auto ansi beforefieldinit HTTPHeader.Program
      .ctor : void()
      GetHeader : void(string)
      Main : void(string[])

HTTPHeader.Program::GetHeader : void(string)
{
  // Code size      220 (0xdc)
  .maxstack 4
  .locals init ([0] class [System]System.Net.WebRequest 'request',
    [1] class [System]System.Net.HttpWebResponse response,
    [2] class [System]System.Net.WebHeaderCollection wh,
    [3] string s,
    [4] class [mscorlib]System.IO.Stream dataStream,
    [5] class [mscorlib]System.IO.StreamReader reader,
    [6] string responseFromServer,
    [7] string[] CS$6$0000,
    [8] int32 CS$7$0001,
    [9] bool CS$4$0002)

  IL_0000: nop
  IL_0001: ldarg.0
  IL_0002: call      class [System]System.Net.WebRequest [System]System.Ne
  IL_0007: stloc.0
  IL_0008: ldloc.0
  IL_0009: call      class [System]System.Net.ICredentials [System]System.I
  IL_000e: callvirt instance void [System]System.Net.WebRequest::set_Cred
  IL_0013: nop
  IL_0014: ldloc.0
  IL_0015: callvirt instance class [System]System.Net.WebResponse [System
  IL_001a: castclass [System]System.Net.HttpWebResponse
  IL_001f: stloc.1
  .try
  {
    IL_0020: nop
    IL_0021: ldstr      "#####Begin Read Header"
    IL_0026: call      void [mscorlib]System.Console::WriteLine(string)
    IL_002b: nop
    IL_002c: ldloc.1
    IL_002d: callvirt instance class [System]System.Net.WebHeaderCollecti
    IL_0032: stloc.2
    IL_0033: nop
    IL_0034: ldloc.2
    IL_0035: callvirt instance string[] [System]System.Collections.Special
    IL_003a: stloc.s  CS$6$0000
    IL_003c: ldc.i4.0
    IL_003d: stloc.s  CS$7$0001
    IL_003f: br.s     IL_0067
    IL_0041: ldloc.s  CS$6$0000
    IL_0042: ldloc.s  CS$7$0001
  }
}
```

```
HTTPHeader.Program::Main : void(string[])
private hidebysig static void Main(string[])
{
  int size      19 (0x13)
  .code 8
  nop
  ldstr      "http://www.google.com"
  call      void HTTPHeader.Program::GetHeader(string)
  nop
  call      string [mscorlib]System.Console::WriteLine(string)
  pop
  ret
}
method Program::Main
```

ข้อดีของ CIL

- ช่วยให้เขียนข้ามภาษาได้ (Interoperability)
- โกล่เคียงภาษาเครื่อง ทำให้แปลงได้รวดเร็ว
- เป็นภาษาที่ไม่ขึ้นกับ CPU
- แปลงเป็นภาษาเครื่องโดย CLR

Common Intermediate Language (CIL)

```
// Calc.cs
using System;
namespace CalculatorExample
{
    // This class contains the app's entry point.
    class Program
    {
        static void Main()
        {
            Calc c = new Calc();
            int ans = c.Add(10, 84);
            Console.WriteLine("10 + 84 is {0}.", ans);
            // Wait for user to press the Enter key before shutting down.
            Console.ReadLine();
        }
    }
    // The C# calculator.
    class Calc
    {
        public int Add(int x, int y)
        {
            return x + y; }
    }
}
```

Common Intermediate Language (CIL)

```
' Calc.vb
Imports System
Namespace CalculatorExample
' A VB "Module" is a class that contains only
' static members.
Module Program
Sub Main()
    Dim c As New Calc
    Dim ans As Integer = c.Add(10, 84)
    Console.WriteLine("10 + 84 is {0}.", ans)
    Console.ReadLine()
End Sub
End Module
Class Calc
    Public Function Add(ByVal x As Integer, ByVal y As Integer) As Integer
        Return x + y
    End Function
End Class
End Namespace
```

Common Intermediate Language (CIL)

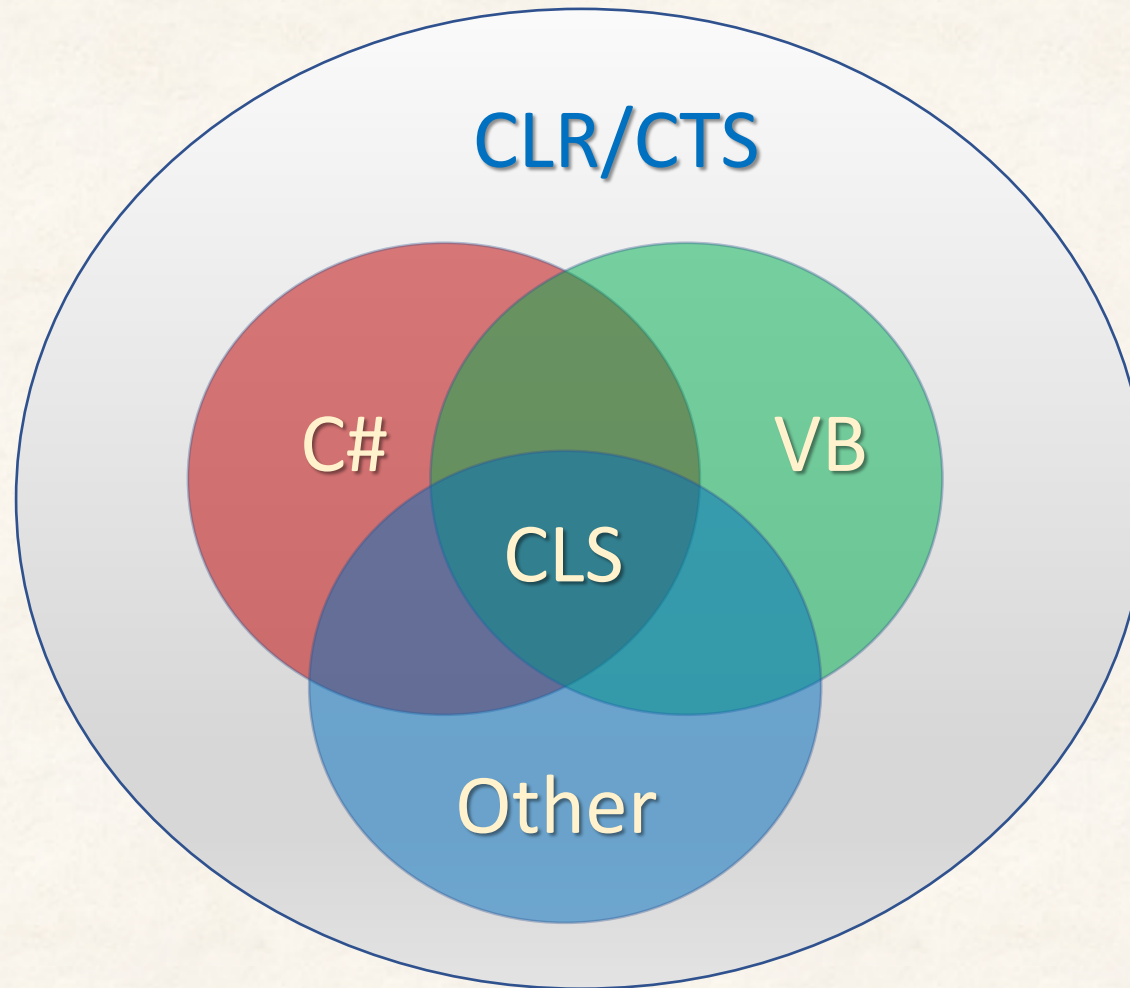
CAL.CS

```
.method public hidebysig instance int32 Add(int32x,
int32 y) cil managed
{
// Code size 9 (0x9)
.maxstack 2
.locals init (int32 V_0)
IL_0000: nop
IL_0001: ldarg.1
IL_0002: ldarg.2
IL_0003: add
IL_0004: stloc.0
IL_0005: br.s IL_0007
IL_0007: ldloc.0
IL_0008: ret
} // end of method Calc::Add
```

CAL.VB

```
.method public instance int32 Add(int32 x,int32 y)
cil managed
{
// Code size 8 (0x8)
.maxstack 2
.locals init (int32 V_0)
IL_0000: ldarg.1
IL_0001: ldarg.2
IL_0002: add.ovf
IL_0003: stloc.0
IL_0004: br.s IL_0006
IL_0006: ldloc.0
IL_0007: ret
} // end of method Calc::Add
```

Common Language Specification (CLS)



Common Language Specification (CLS)

- แต่ละภาษาใน .NET Framework มี syntax ต่างกันไป
 - ใน C# ใช้ operator + ในการเชื่อมต่อความ แต่ VB.NET ใช้ operator &
 - ใน C# operator สามารถทำ overloading ได้ แต่ VB.NET ทำไม่ได้

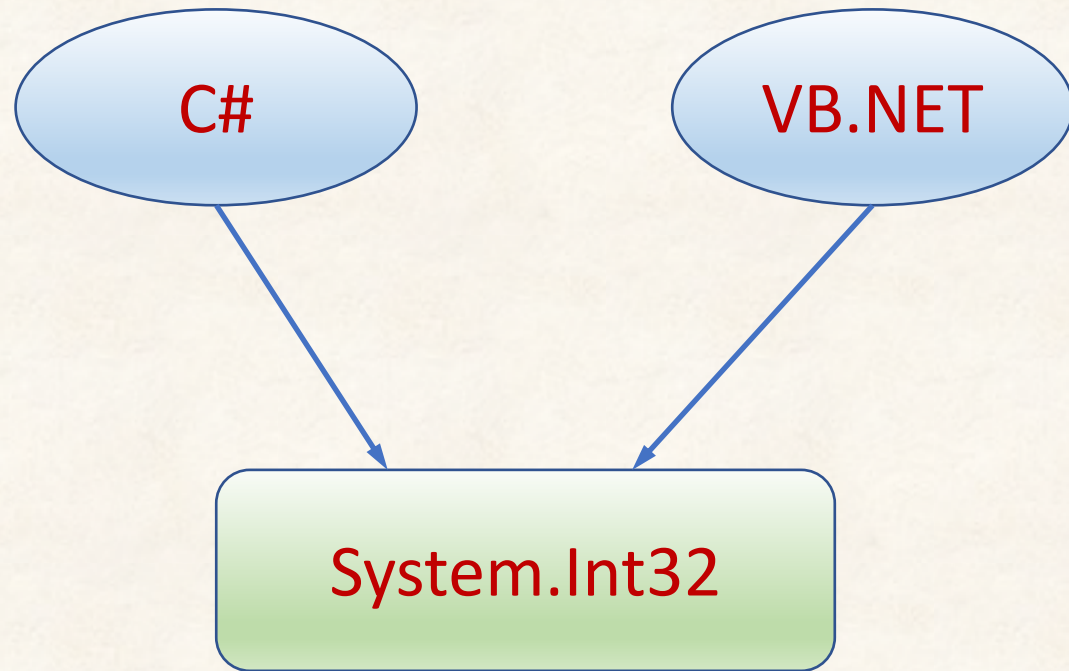
VB.NET

```
Public Sub Foo()  
    ...  
End Sub
```

C#

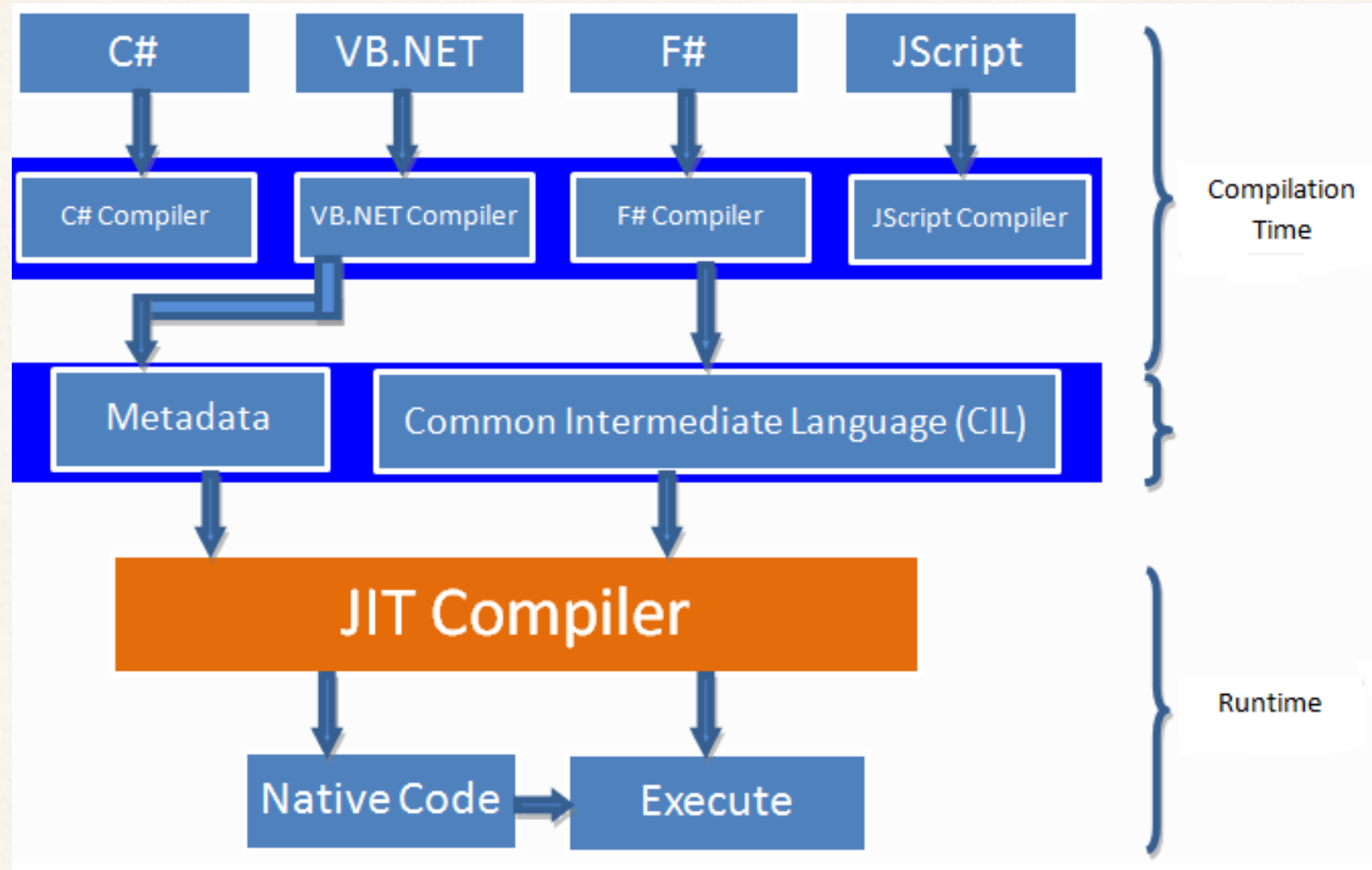
```
Public void Foo()  
{  
    ...  
}
```


Common Type System (CTS)

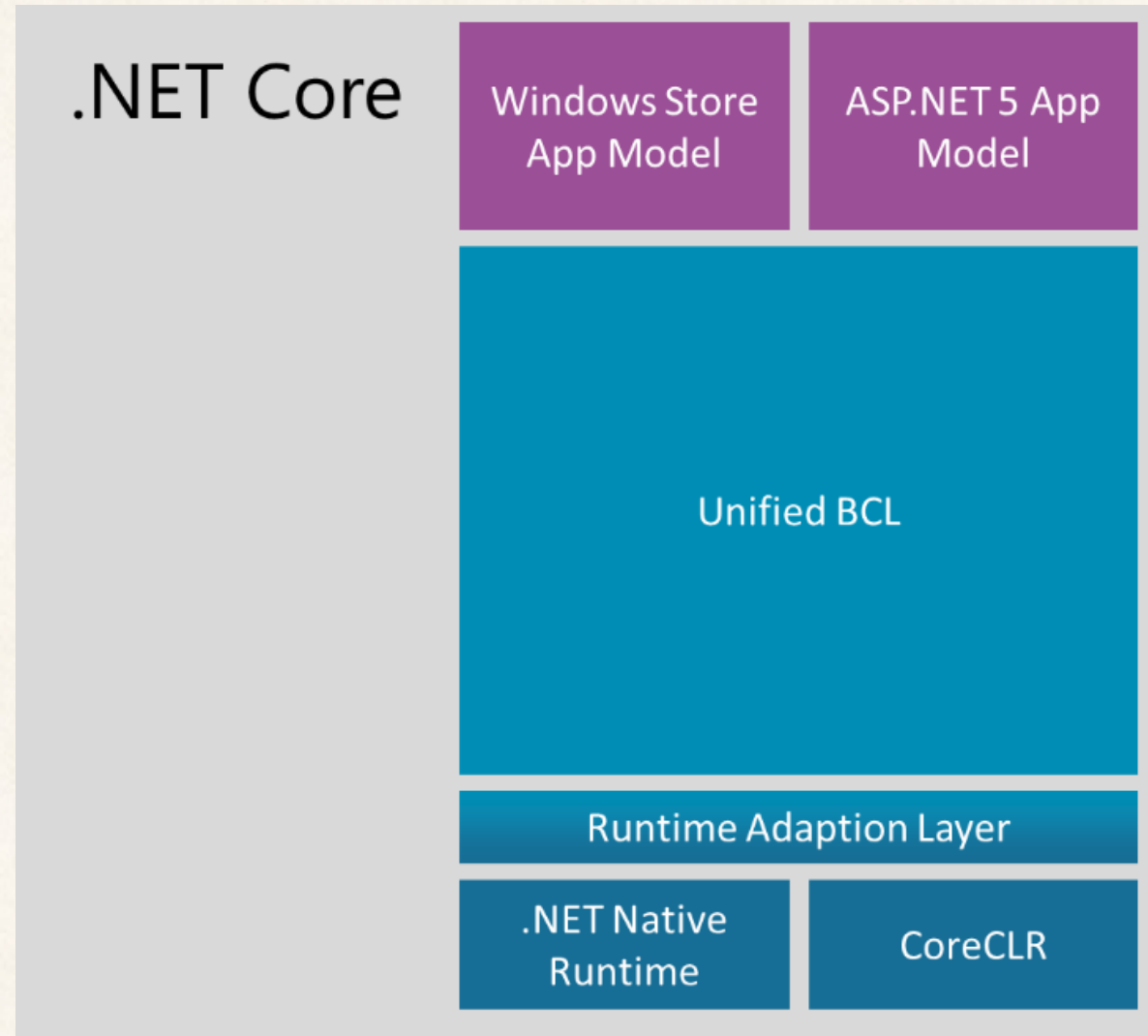


MSIL Assembler Name	Class Library Name	Type	Description
bool	System.Boolean	Value	values either true or false
char	System.Char	Value	Unicode character (16 bits)
class System.Object	System.Object	Reference	base object of all classes
class System.String	System.String	Reference	Unicode string (16-bit characters)
float32	System.Single	Value	IEEE 32-bit float
float64	System.Double	Value	IEEE 64-bit float
int16	System.Int16	Value	signed 16-bit integer
int32	System.Int32	Value	signed 32-bit integer
int64	System.Int64	Value	signed 64-bit integer
natural int	System.IntPtr	Value	signed integer, natural size
unsigned int8	System.Byte	Value	unsigned 8-bit integer

JIT : Just-In-Time compiler

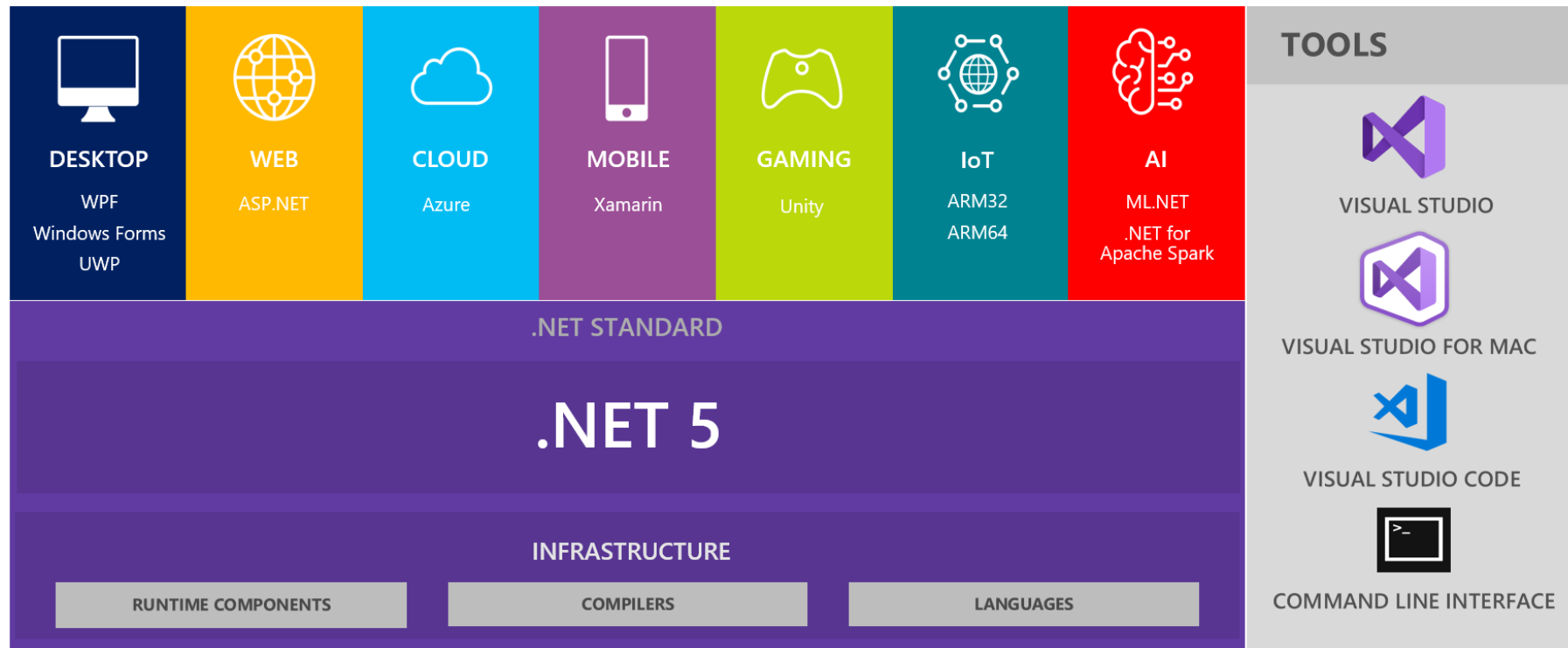


.NET Core

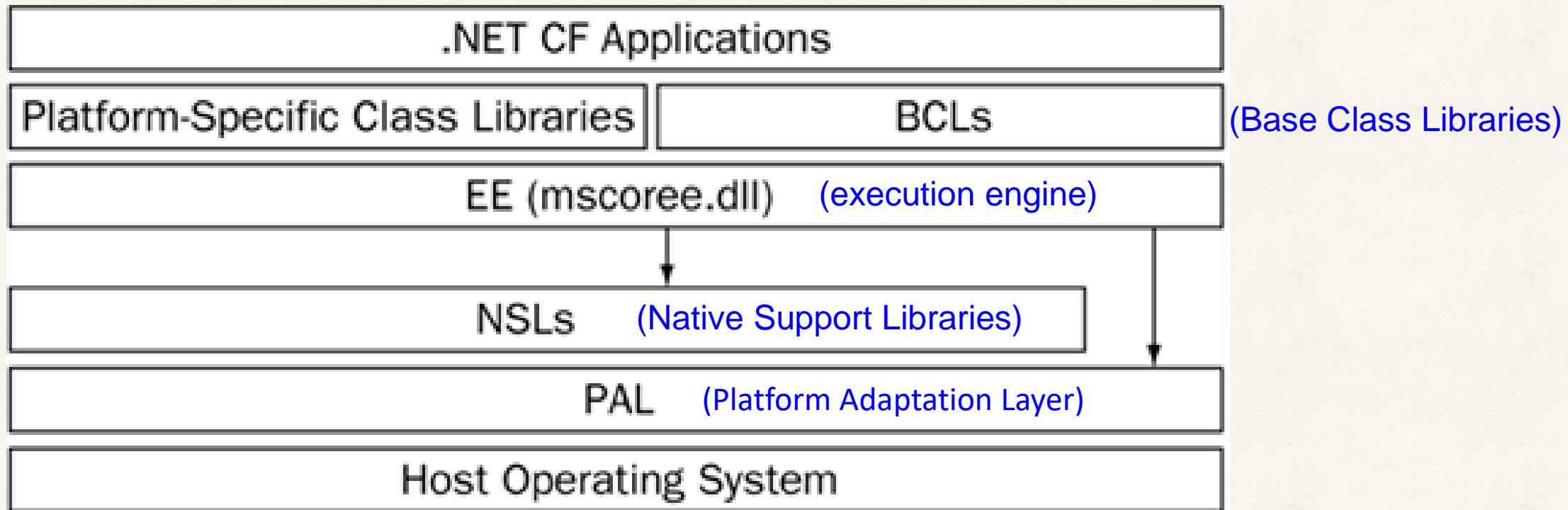


.NET 5

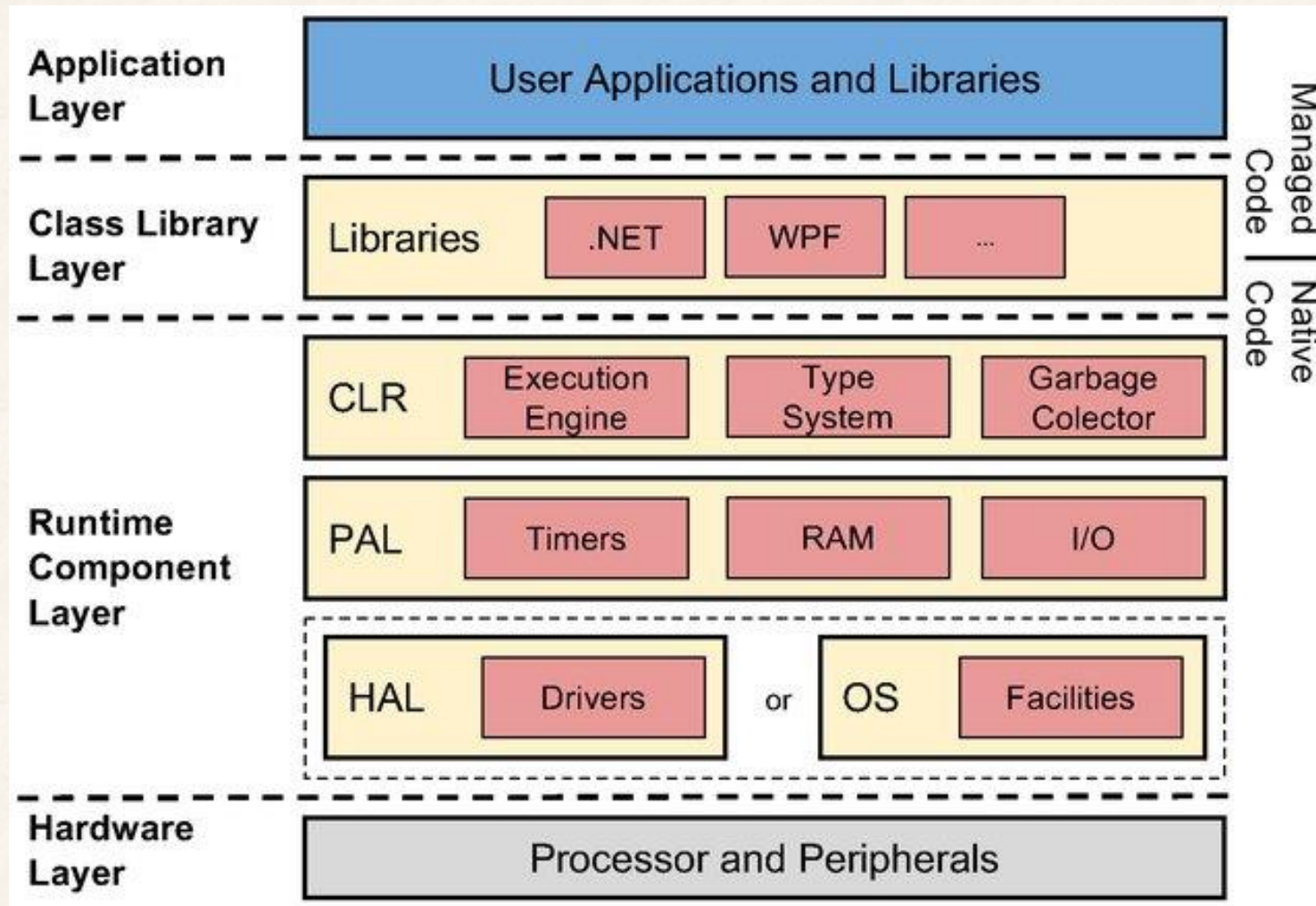
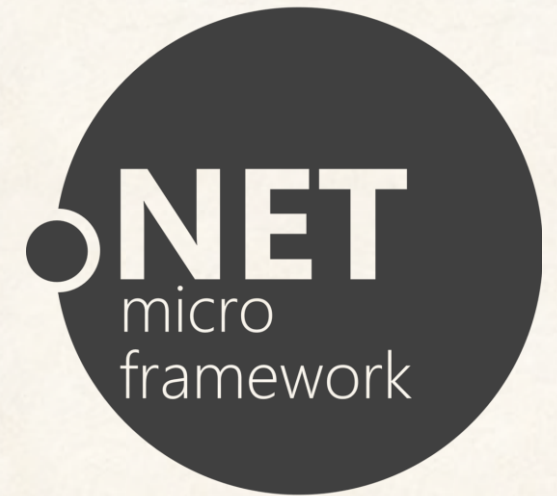
.NET – A unified platform



.NET Compact Framework (.NET CF)

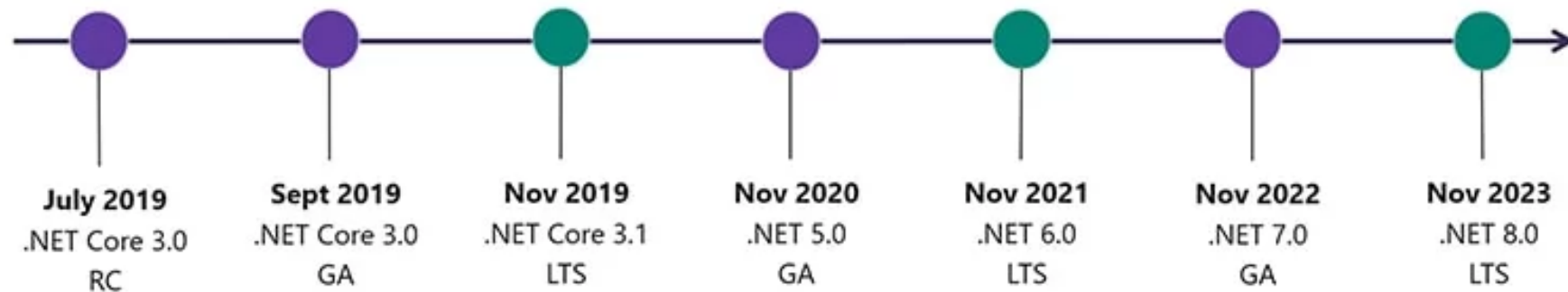


.NET Micro Framework



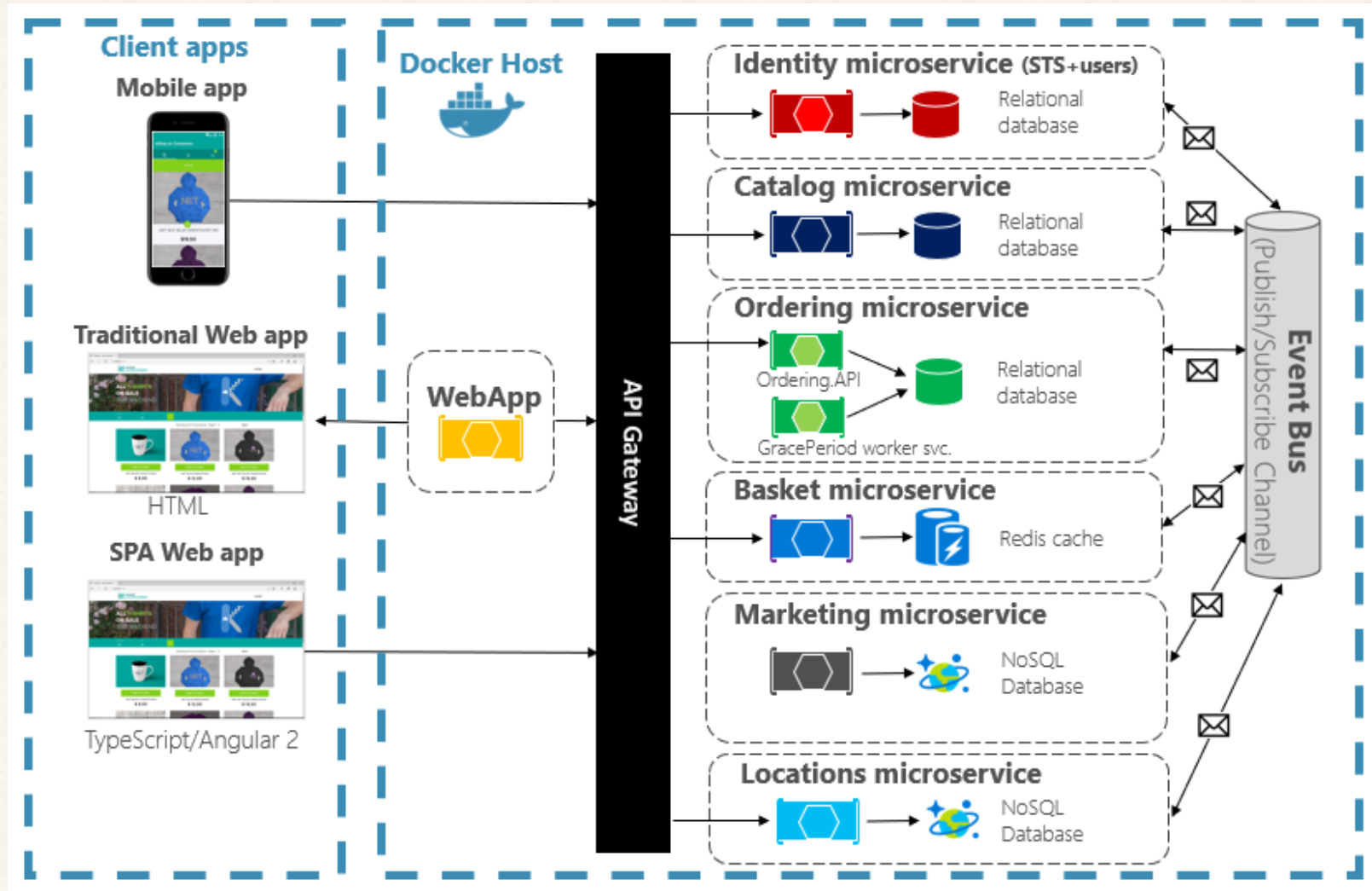
.NET ในอนาคต

.NET Schedule



- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed

2 รองรับ Cloud Native Application



3 Game engine: Unity



- Unity ทำงานบน .NET และ mono
- รองรับภาษาโปรแกรมในภาษา UnityScript, C#, Boo (syntax คล้าย python)
- รองรับไฟล์สำหรับการ Render ที่หลากหลาย เช่น 3ds Max, Maya, Blender, Adobe Photoshop เป็นต้น
- รองรับฟังก์ชันทาง physics เช่น Nvidia, PhysX physics
- รองรับการทำ Animation
- รองรับการทำ 2D และ 3D
- Multiplatform





4 ASP.NET Core Blazor

- Blazor เป็น web development framework จาก Microsoft
- อยู่บน .NET Platform ทั้งฝั่ง Client และ Server
- แต่มีพื้นฐานที่ต่างไปจากภาษา client-side frameworks ที่มีที่มาจาก JavaScript

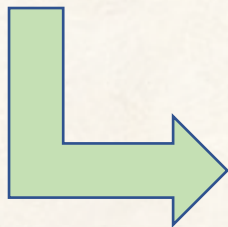
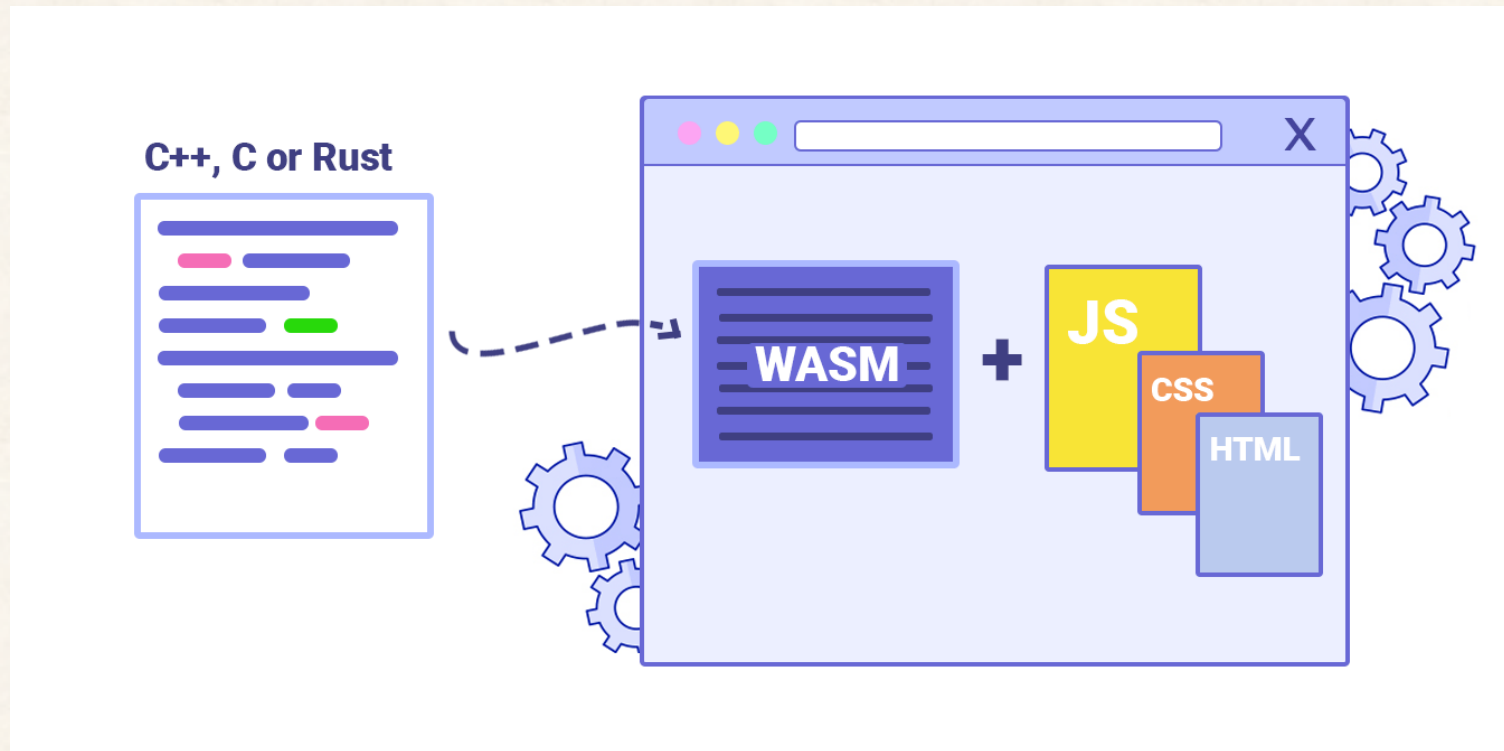
ข้อดีของ Blazor

- C# Language (ฝัง browser ก็เช่นเดียวกัน)
- มีเครื่องมือ (IDE) ที่ดี
- มีสเถียรภาพสูง
- สภาพแวดล้อมที่เป็นหนึ่งเดียวกับการพัฒนา platform อื่นๆ
- มีมาตรฐาน (HTML, CSS, Web Assembly,...)
- Code reuse
- Component based
- Skill reuse.

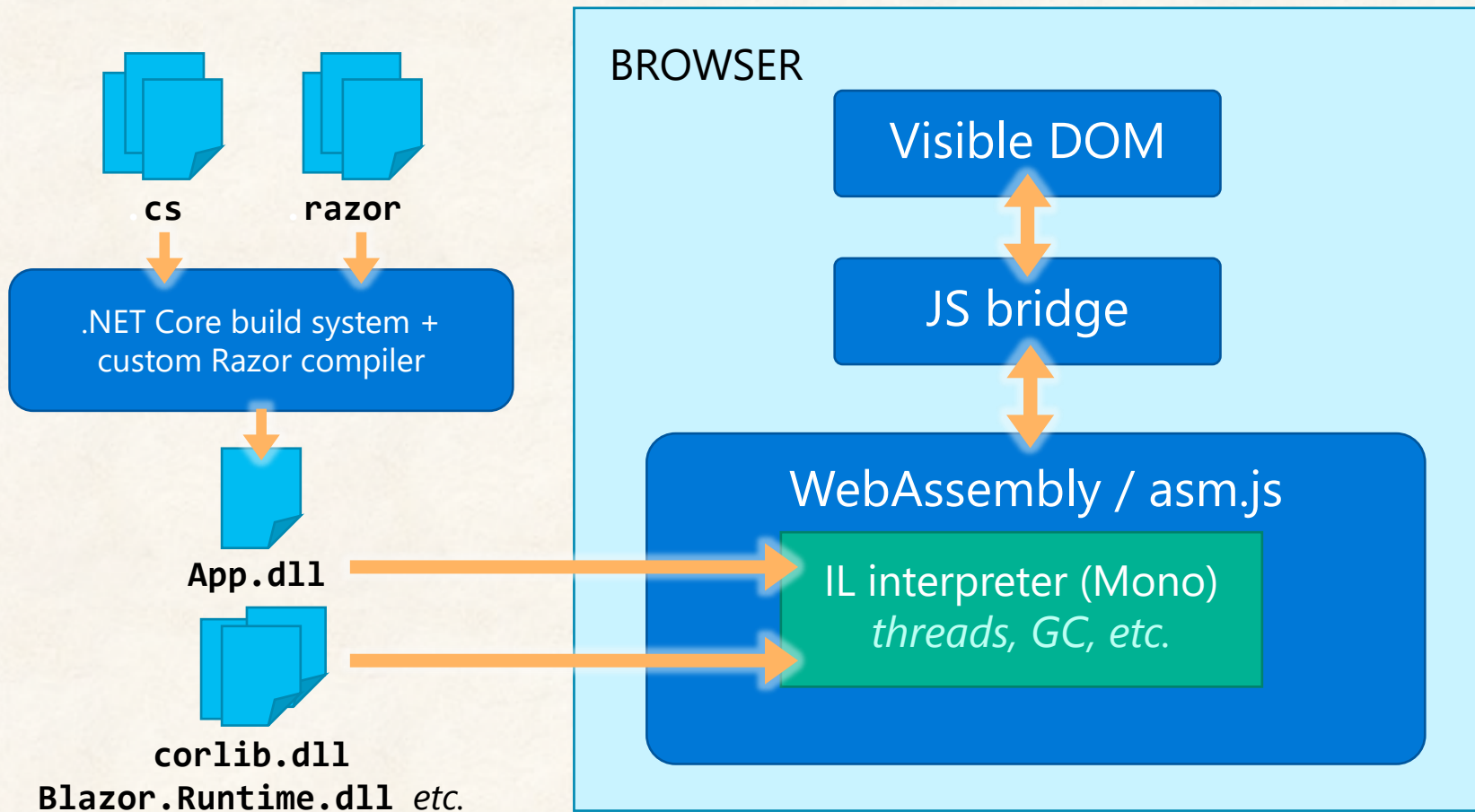
รูปแบบของ Blazor

- Blazor WebAssembly – Client Side
- Blazor Server – Server Side
- ใช้ code เดียวกัน
 - ยกเว้นบางกรณีเช่นการเข้าถึงข้อมูลในฝั่ง server

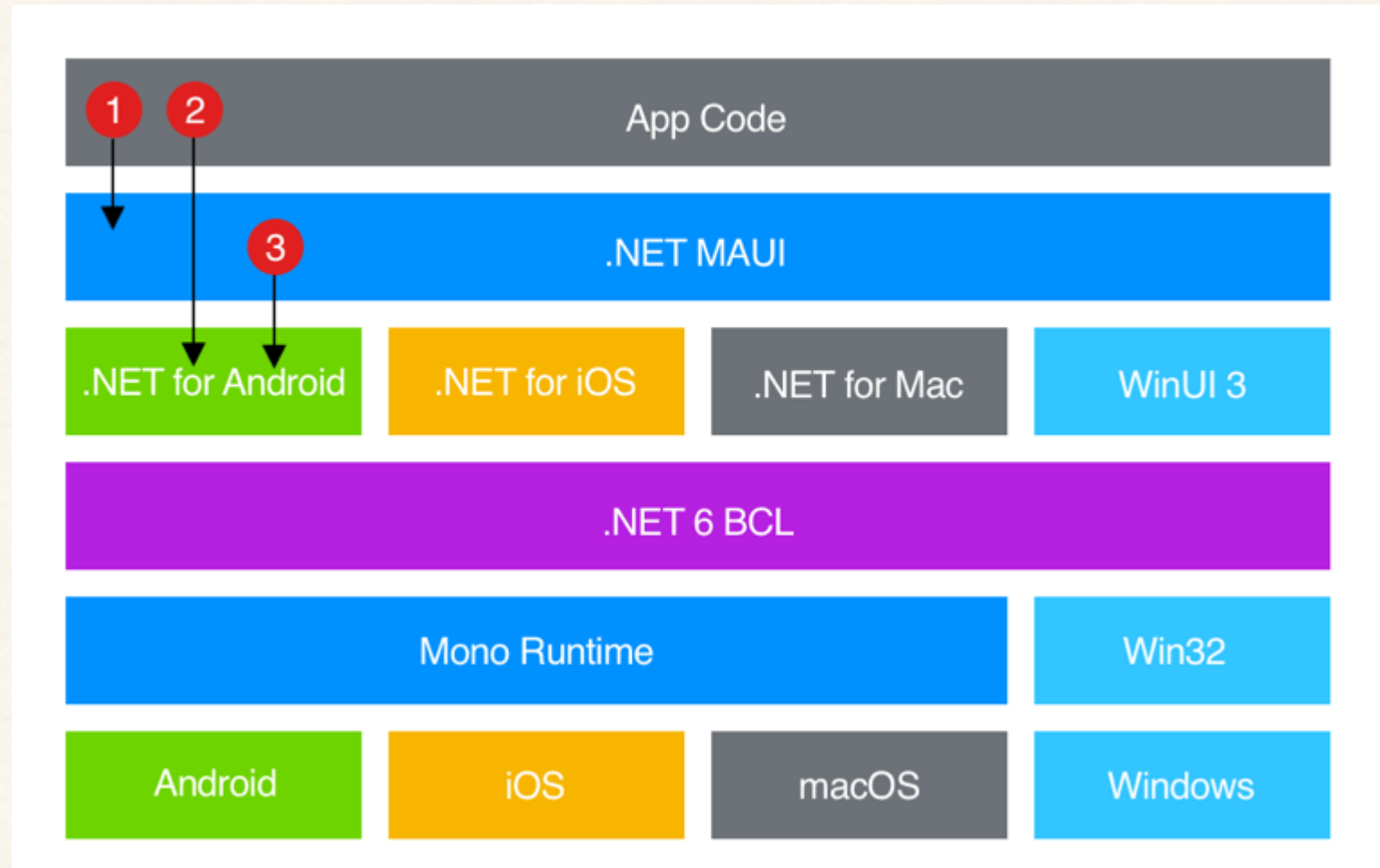
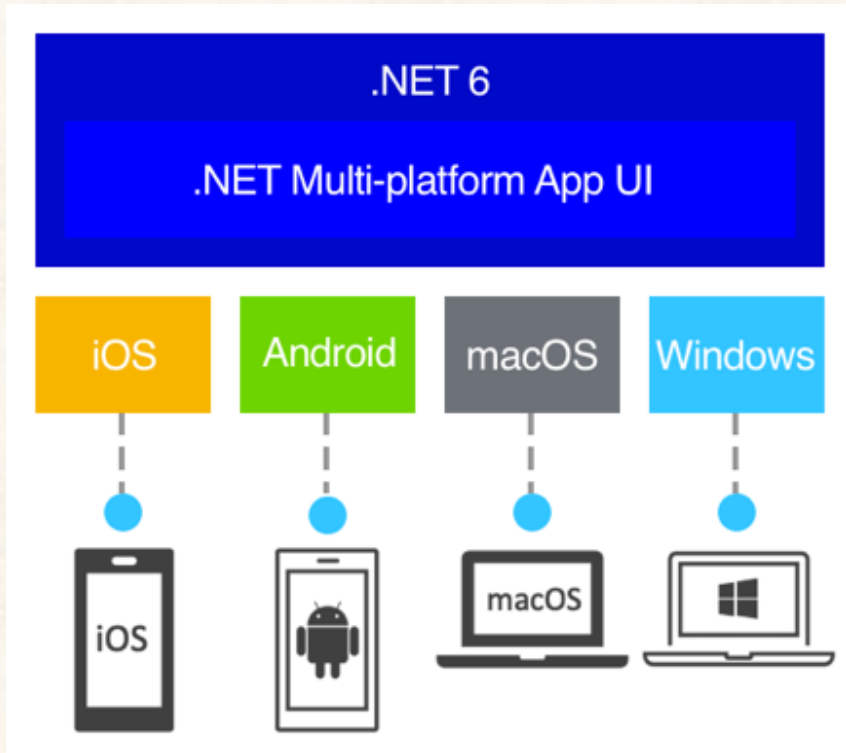
WebAssembly



การทำงานของ Blazor



5 .NET MAUI (Multi-platform App UI)

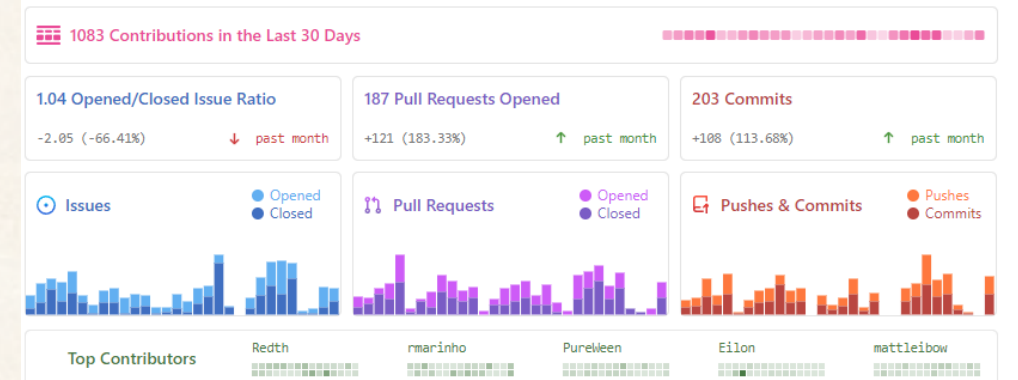


.NET MAUI (github)



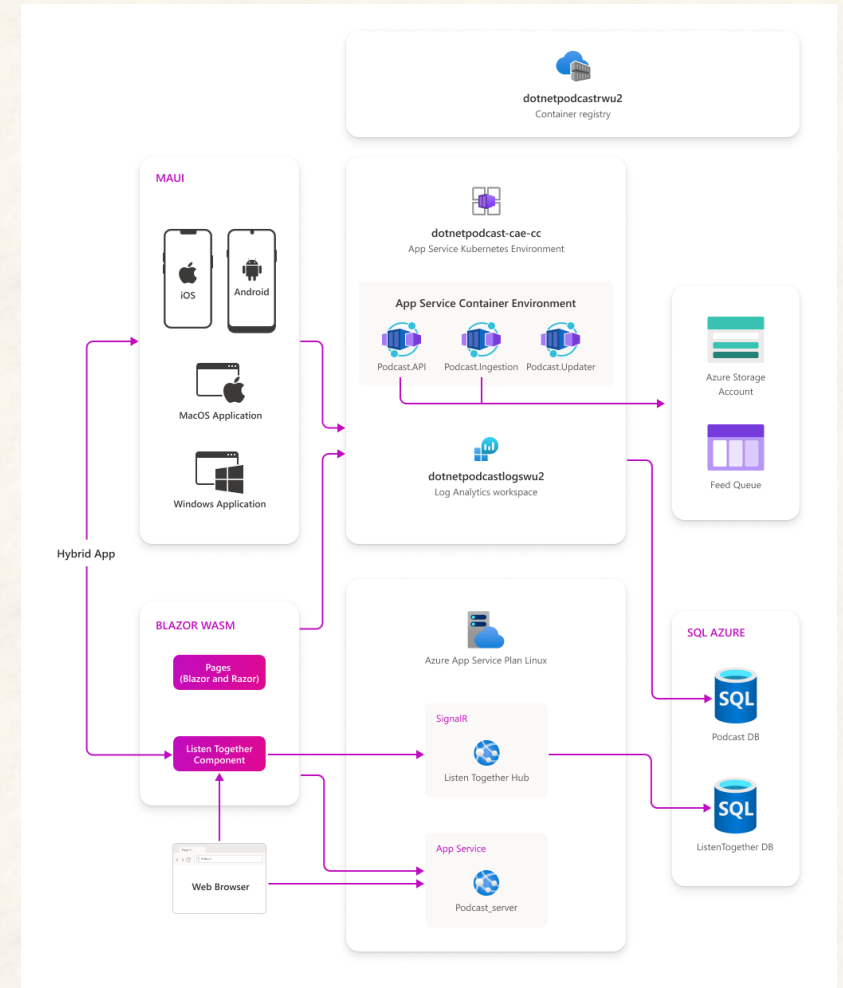
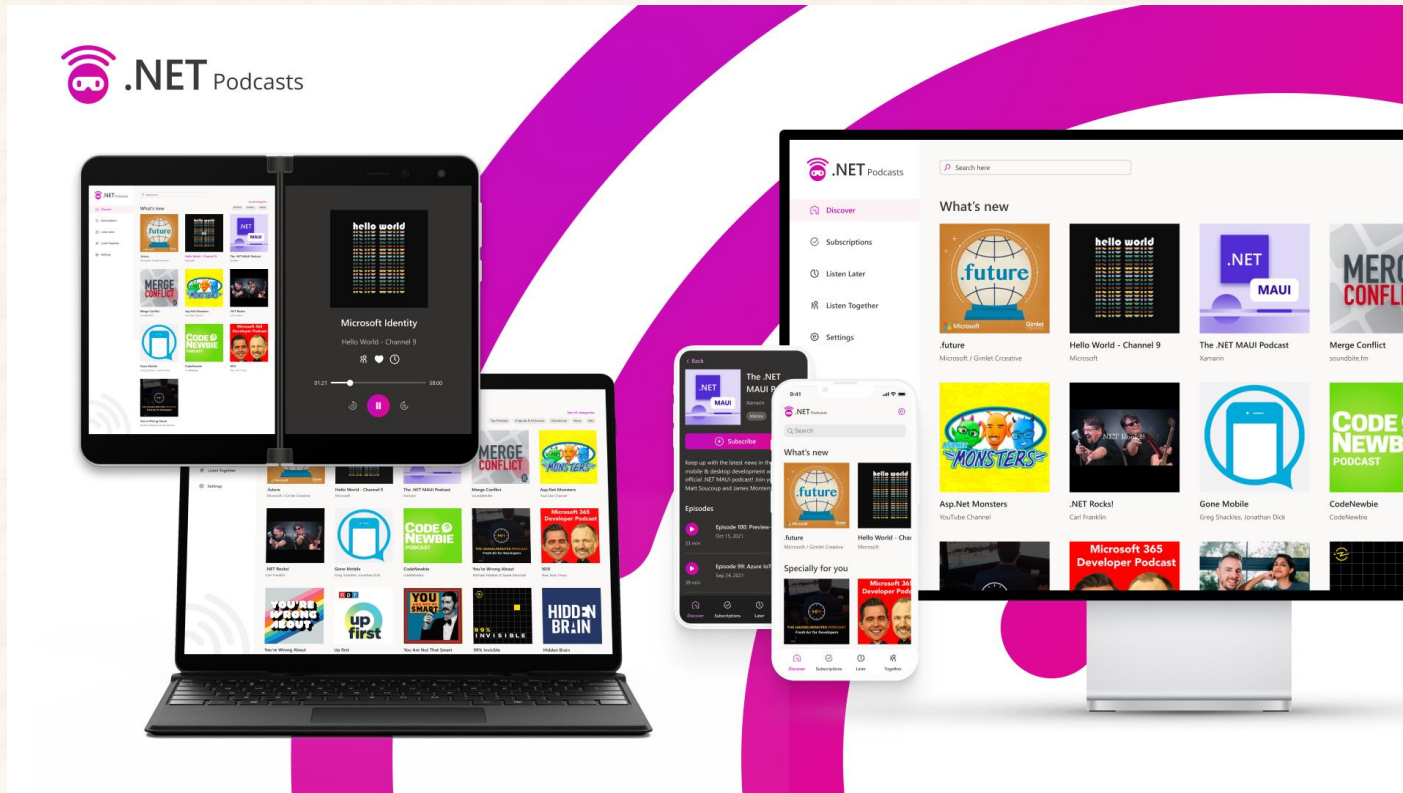
<https://github.com/dotnet/maui>

Stats

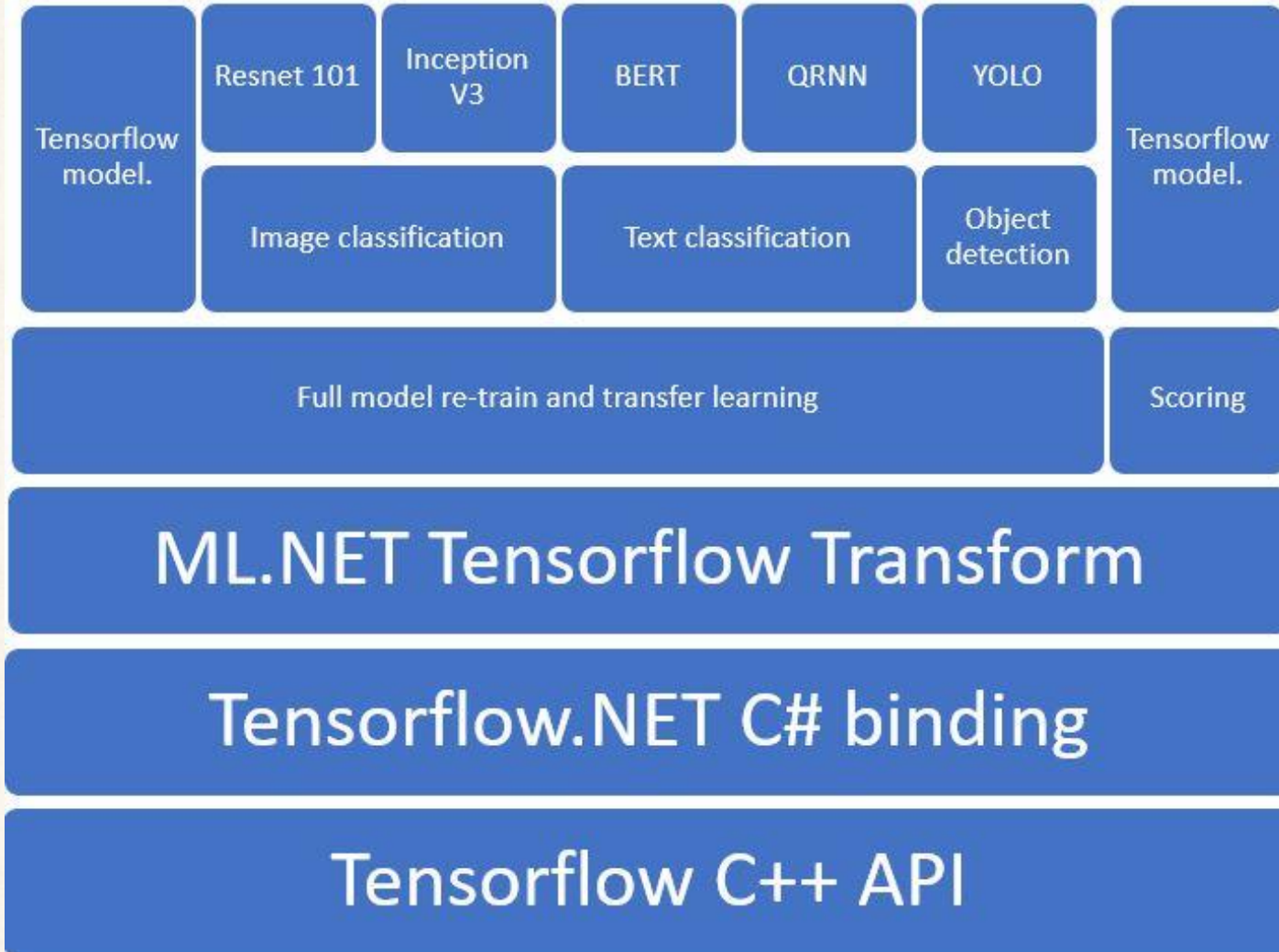


.NET Podcasts(github)

<https://github.com/microsoft/dotnet-podcasts>



6 ML.NET และ AI (Artificial Intelligence)



<https://github.com/dotnet/machinelearning>

<https://github.com/dotnet/machinelearning-samples>

<https://dotnet.microsoft.com/en-us/learn/ml-dotnet/get-started-tutorial/intro>

<https://docs.microsoft.com/th-th/dotnet/machine-learning/>

<https://www.codemag.com/Article/1911042/ML.NET-Machine-Learning-for-.NET-Developers>

7 Open source

Github.com เป็น host ของ .NET จำนวนมาก

