.net core

Shalabayev Yerassyl Vladimirovich

.net core

- CLR Basics
- Designing Types
- Essential Types
- Core Facilities
- Threading

ASP.NET Core MVC 2 in detail

- Configuring Applications
- URL Routing & Advanced Routing Features
- Controllers and Actions and Views
- Dependency Injection
- Filters
- API Controllers
- Unit testing
- View Components
- Model Validation
- Identity

Overview

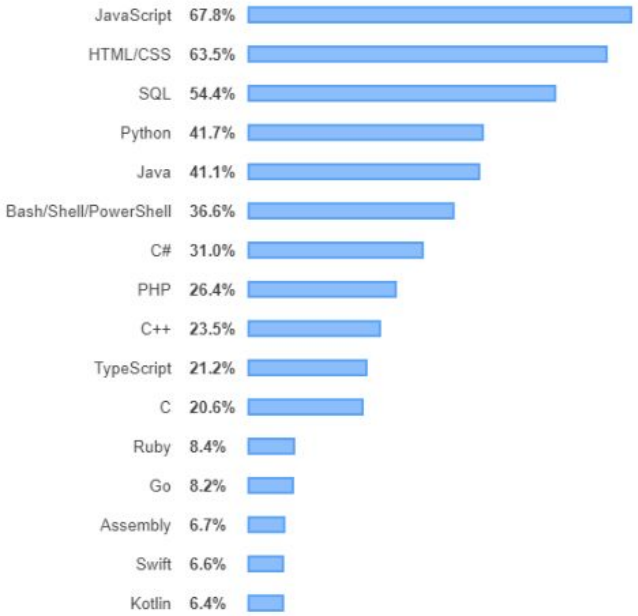
- Community statistics
- Brief history
- The CLR’s Execution Model
- What is dotnet core and C# (CSharp) ?
- Just in time (JIT), Intermediate Language (IL)
- Common Types System

Stackoverflow survey 2019

Most Popular Technologies

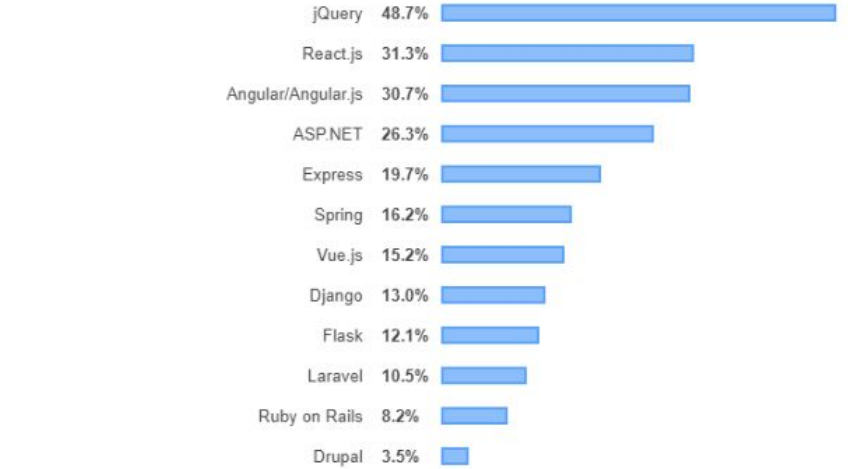
Programming, Scripting, and Markup Languages

All Respondents | Professional Developers

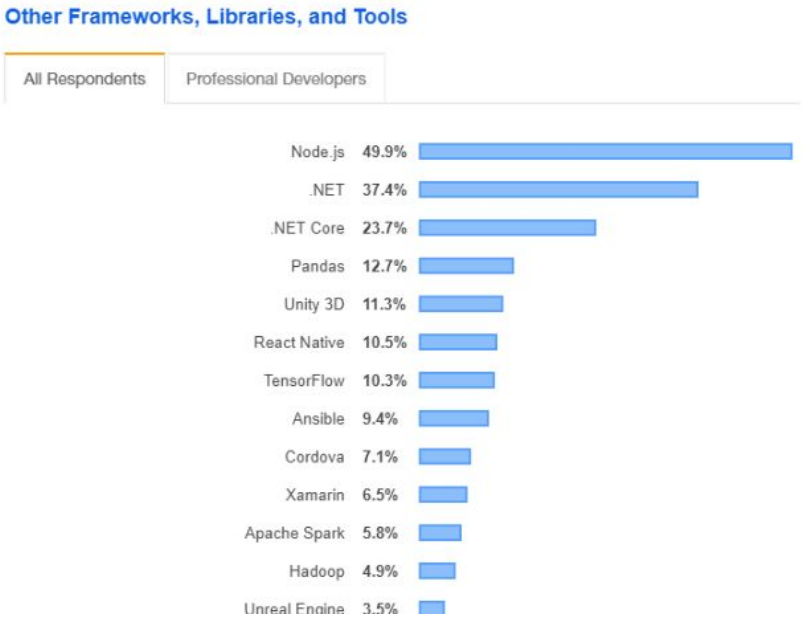


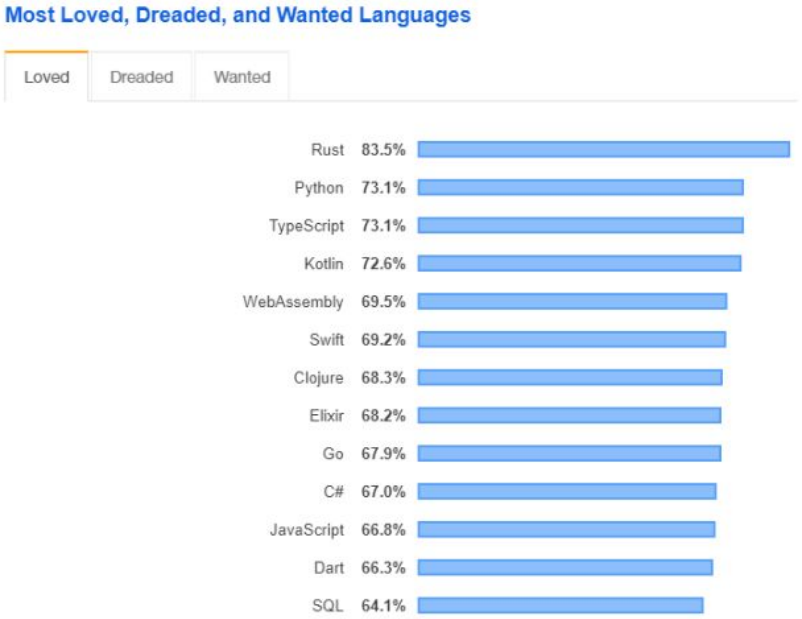
Web Frameworks

All Respondents | Professional Developers

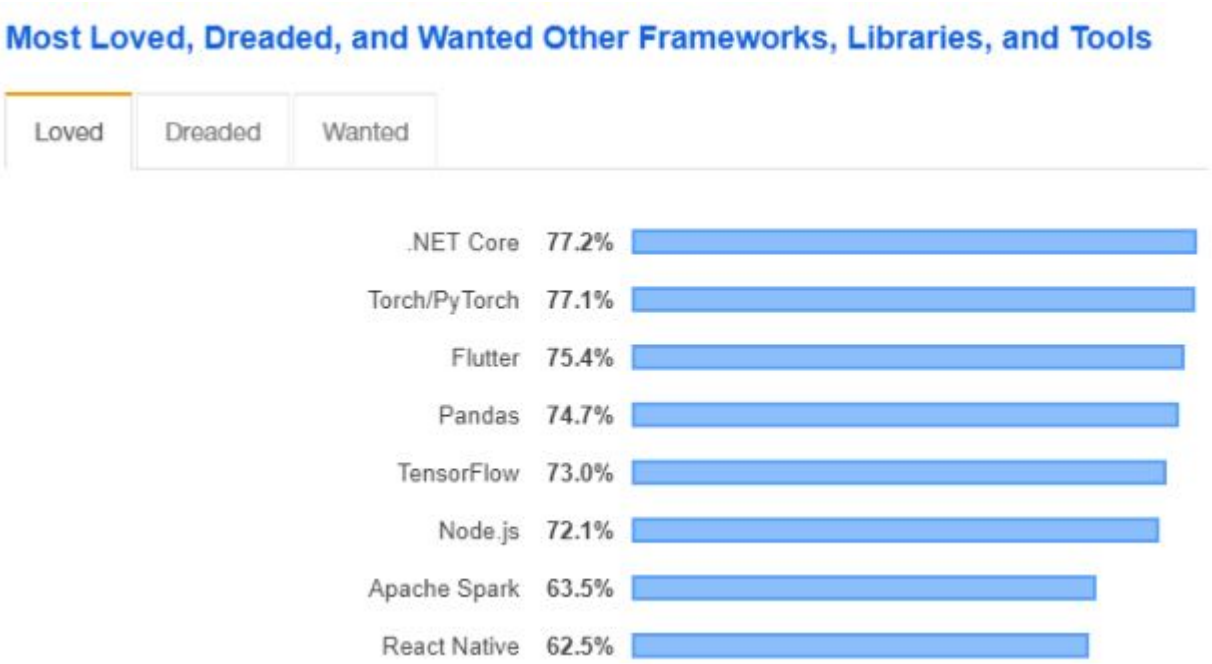


Stackoverflow survey 2019

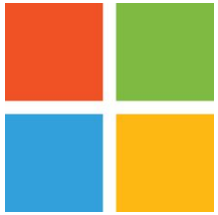




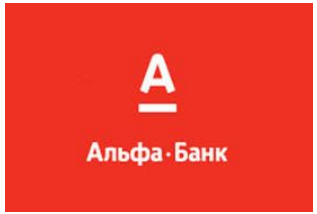
Stackoverflow survey 2019



Who uses



Local



Average salary of .Net developers

Junior developer

From 150k - 250k ₸

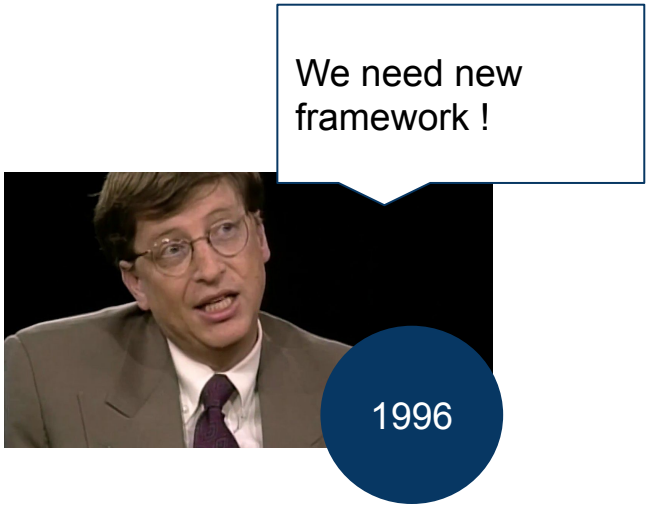
Middle developer

From 250k - 450k ₸

Senior developer

From 450k - ??? ₸

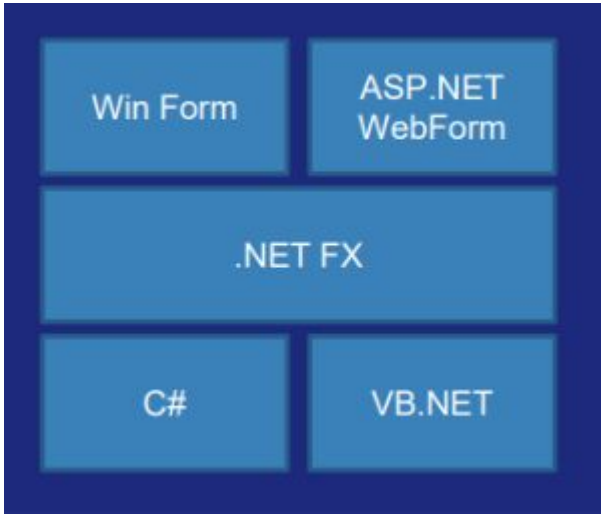
History



Requirements in 1996

- ...starting after Java
- JVM -> Jitted execution engine
- High cost per megabyte
- Raising internet, not cloud
- COM3 and Visual Basic 6 needed a successor
- Windows Only (no MacOS, no Linux n°1 Cloud OS)

.Net and Mono



2001

2004

Asp.net MVC

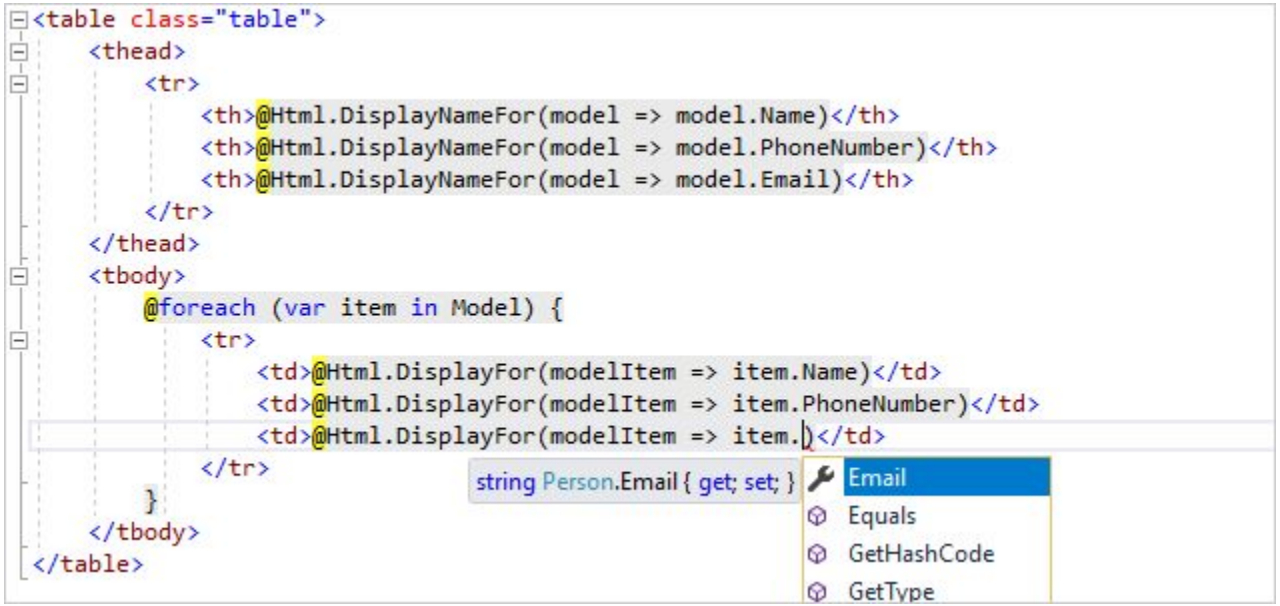
MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns.

Using the MVC pattern for websites, requests are routed to a Controller which is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display, and provides it with the Model. The View renders the final page, based on the data in the Model.

2009

Asp.net mvc 3 and razor

Razor is an ASP.NET programming syntax used to create dynamic web pages with the C# or VB.NET programming languages




```
<table class="table">
    <thead>
        <tr>
            <th>@Html.DisplayNameFor(model => model.Name)</th>
            <th>@Html.DisplayNameFor(model => model.PhoneNumber)</th>
            <th>@Html.DisplayNameFor(model => model.Email)</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model) {
            <tr>
                <td>@Html.DisplayFor(modelItem => item.Name)</td>
                <td>@Html.DisplayFor(modelItem => item.PhoneNumber)</td>
                <td>@Html.DisplayFor(modelItem => item.)</td>
            </tr>
        }
    </tbody>
</table>
```

2010

.net core

2016

.net core

- Cloud first (high dencity, light, optimized)
- Cross platform
  - Windows, linux on cloud
  - Windows, linux, mac os on desktop
  - iOS, Android
  - (ARM)(x86) on IoT
- Open source
  - Open source standarts
  - Contributions from community
- Mono
  - What .net should be
  - Xamarin

.net core - New deployments

- Application Lifecycle Management
  - Source Code Management
  - Continuous Integration
  - Continuous Delivery
  - Testing
- Containerization and Docker
- Microservices

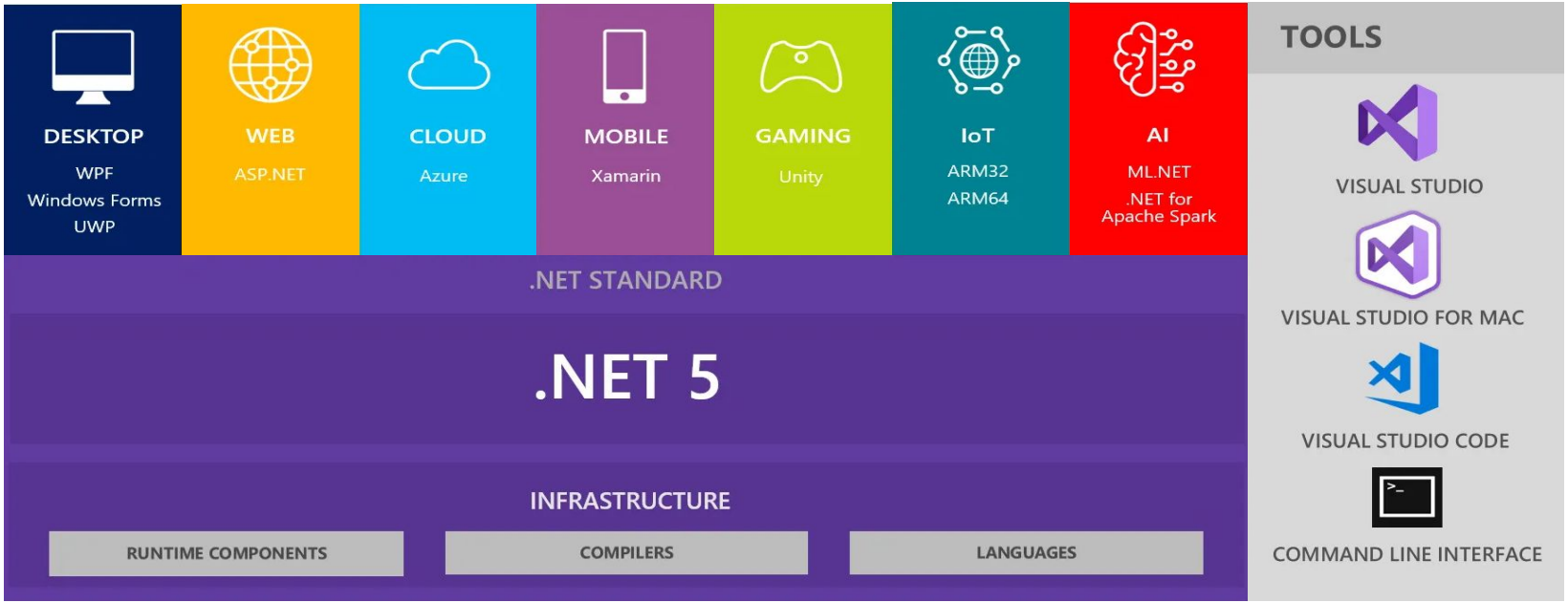
Welcome .net core

- Cross-platform
- Open source
- Microservices architecture
- Containers
- Modern Architecture
- Modular Design
- Various development tools
- A need for high-performance and scalable systems
- Allows side by side of .NET versions per application level

.net core vs .net framework

.NET Core	.NET Framework
You need training, searching and developing	Develop easier for legacy teams
Windows, macOS, and Linux on AMD64, x86, and ARM	Windows-only, PC-only, deeply tied to IIS
Modular	A whole framework
UWP, ASP.NET Core, Razor Pages, CLI	WPF, Windows Forms, ASP.NET (WebForm, MVC, Pages)
.NET Core is much faster High-performance and scalable system without UI	Speed is not an important concern
You are using Docker containers	You run your app in old fashion

What you can create ?



The CLR’s Execution Model

What is .net ?

C# is a PL language.

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

\+

C#
Compiler

=

Assembly
Program.dll

IL code

```
.method private hidebysig static void Main() cil managed
{
.entrypoint
// Code size 13
.maxstack 8
IL_0000: nop
IL_0001: ldstr "Hello. My name is Inigo Montoya."
IL_0006: call void [System.Console]System.Console::WriteLine(string)
IL_000b: nop
IL_000c: ret
} // End of method System.Void HelloWorld::Main()
```

Build time

Lecturer understands, CPU does not

Lecturer is confused, CPU does not understand

Just in time

C# code + C# compiler

IL code

```
.method private hidebysig static void Main() cil managed
{
.entrypoint
// Code size 13
.maxstack 8
IL_0000: nop
IL_0001: ldstr "Hello. My name is Inigo Montoya."
IL_0006: call void [System.Console]System.Console::WriteLine(string)
IL_000b: nop
IL_000c: ret
} // End of method System.Void HelloWorld::Main()
```

\+

JIT

=

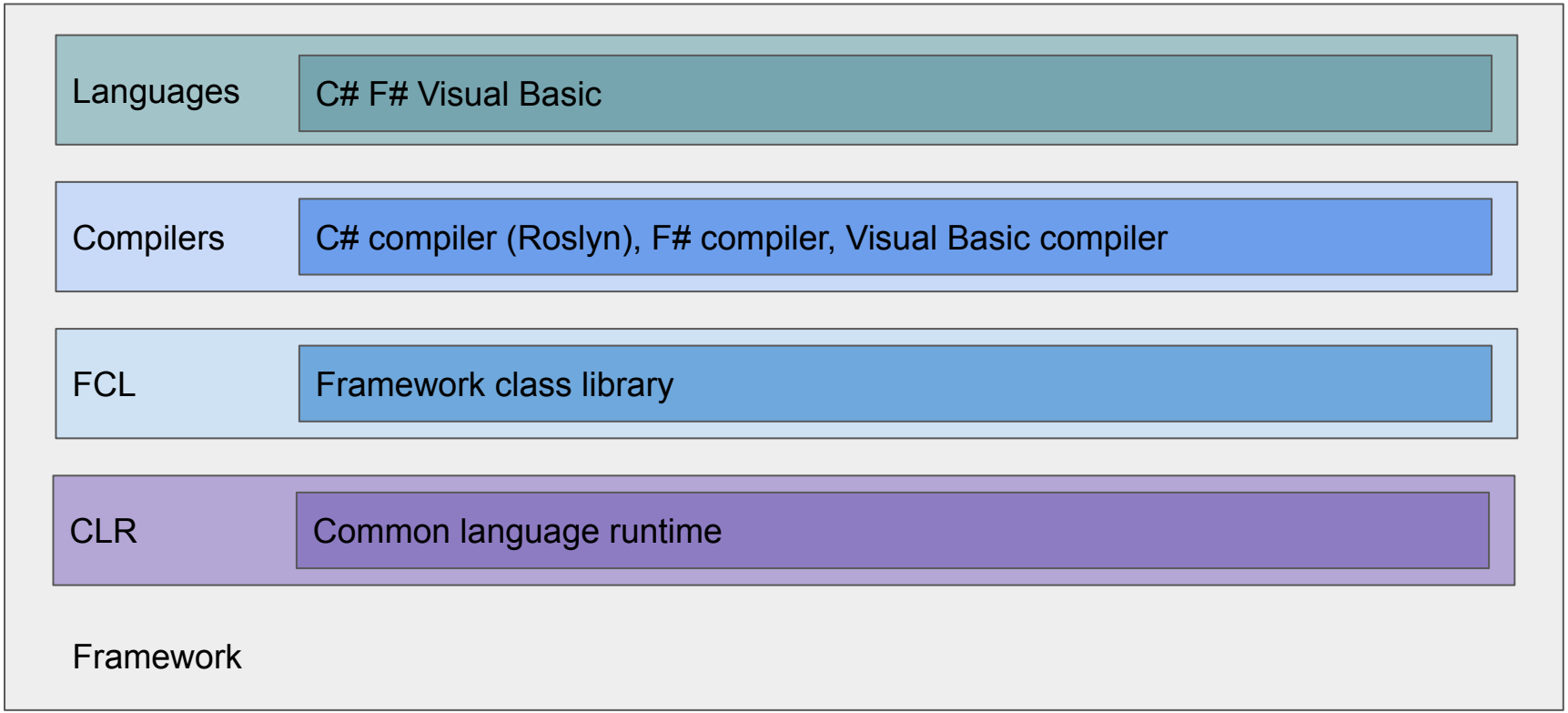
Native code

```
loc_0041150D: var_B4 = 2
              For var_30 = 2 To 1 Step &H11
loc_00411544: var_FC = var_E4
loc_0041154A: GoTo loc_004117DC
loc_0041156A: var_60 = Str(var_30)
loc_00411577: var_6C = "Path" & var_60
loc_0041157A: var_74 = 8
loc_00411596: var_4 = 9
loc_0041159D: var_8C = &H405330
loc_004115A7: var_94 = 8
loc_004115E8: var_60 = CStr("Path" & var_60)
loc_004115FF: var_6C = GetSetting("PictureVIEWER","Path",var
loc_00411602: var_74 = 8
loc_0041160F: var_44 = GetSetting("PictureVIEWER","Path",var
loc_00411625: var_8C = "SHS.ShackS"
loc_0041162F: var_94 = 8
loc_00411648: Var_Ret_2 = var_44 + "SHS.ShackS" '__vbaVarAdd
loc_0041164F: call MSVBVM60.DLL.__vbaStrVarMove(Var_Ret_2, v
loc_00411667: call Open #(00000020h, FFFFFFFFh, 00000001h, M
```

RUNTIME

CPU understands, lecturer does not

.net core is a framework



The Framework Class Library

The FCL is a set of DLL assemblies that contain several thousand type definitions in which each type exposes some functionality. Microsoft is producing additional libraries such as the Windows Azure SDK and the DirectX SDK. These additional libraries provide even more types, exposing even more functionality for your use. In fact, Microsoft is producing many libraries at a phenomenal rate, making it easier than ever for developers to use various Microsoft technologies.

CLR

IL code

input

produces smart native code or reuse existed

Tells CPU what to do

- Determine CPU type
- Manage memory
- Type safety

The Common Type System

The CTS specification states that a type can contain zero or more members.

- **Field** A data variable that is part of the object's state. Fields are identified by their name and type.
- **Method** A function that performs an operation on the object, often changing the object's state. Methods have a name, a signature, and modifiers. The signature specifies the number of parameters (and their sequence), the types of the parameters, whether a value is returned by the method, and if so, the type of the value returned by the method.

The Common Type System

- **Property** To the caller, this member looks like a field. But to the type implementer, it looks like a method (or two). Properties allow an implementer to validate input parameters and object state before accessing the value and/or calculating a value only when necessary. They also allow a user of the type to have simplified syntax. Finally, properties allow you to create read-only or write-only “fields."
- **Event** An event allows a notification mechanism between an object and other interested objects. For example, a button could offer an event that notifies other objects when the button is clicked.

```
class Animal
{

    public int age;


    0 references
    public String Name { get; set; }


    0 references
    public void Move() { }


    public delegate void AnimalMovedTo(int position);
    public event AnimalMovedTo AnimalMoved;


}
```

Access to a members

- **Private** The member is accessible only by other members in the same class type
- **Family** The member is accessible by derived types, regardless of whether they are within the same assembly. Note that many languages (such as C++ and C#) refer to family as **protected**
- **Family and assembly** The member is accessible by derived types, but only if the derived type is defined in the same assembly. **Private protected** since c# 7.2
- **Assembly** The member is accessible by any code in the same assembly. Many languages refer to assembly as **internal**.
- **Family or assembly** The member is accessible by derived types in any assembly. The member is also accessible by any types in the same assembly. C# refers to family or assembly as **protected internal**.
- **Public** The member is accessible by any code in any assembly.

```
2 references
internal class Animal //default
{
    protected int age;

    protected internal int weight;

    0 references
    private String Name { get; set; }


    0 references
    internal void Move() { }

}
```