C# Application Development
Week One

COMP 3602

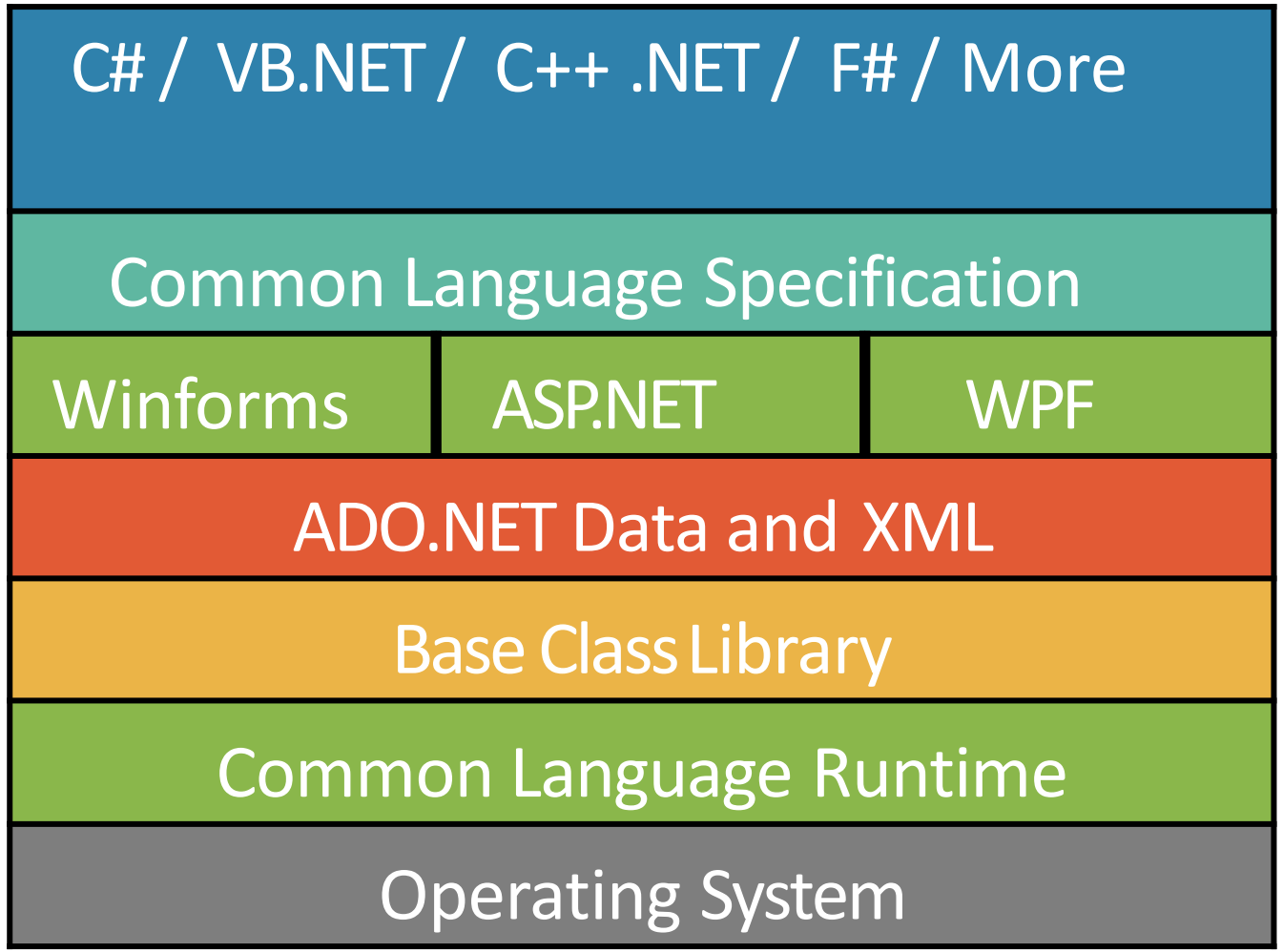
Tonight’s Learning Outcomes

Introduction to C#/.Net

Working with Numeric Data

Reading and Writing to the Console

Getting to know Visual Studio

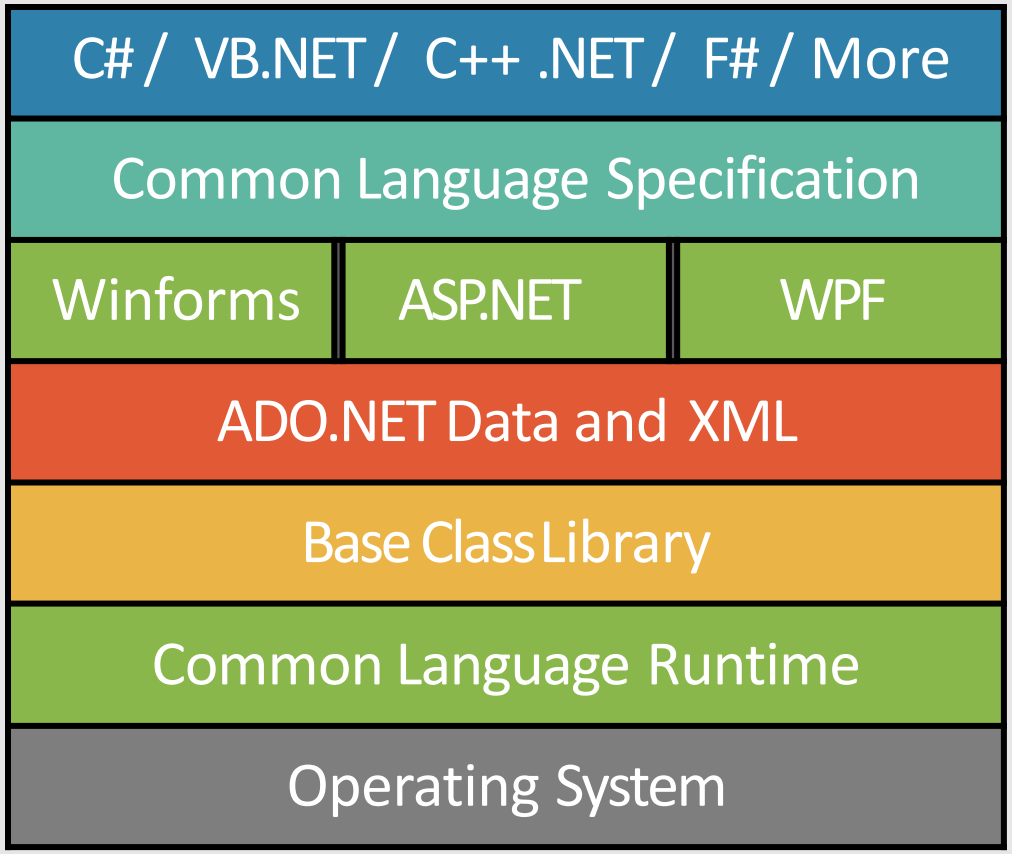


The .NET Framework

- Development environment
- .NET initiative commenced in 1996
- Multiple language support (> 40)
- Base Class Library (> 2000 classes)
- Framework Class Library (> 6000 classes)
- CLR (Memory Management)

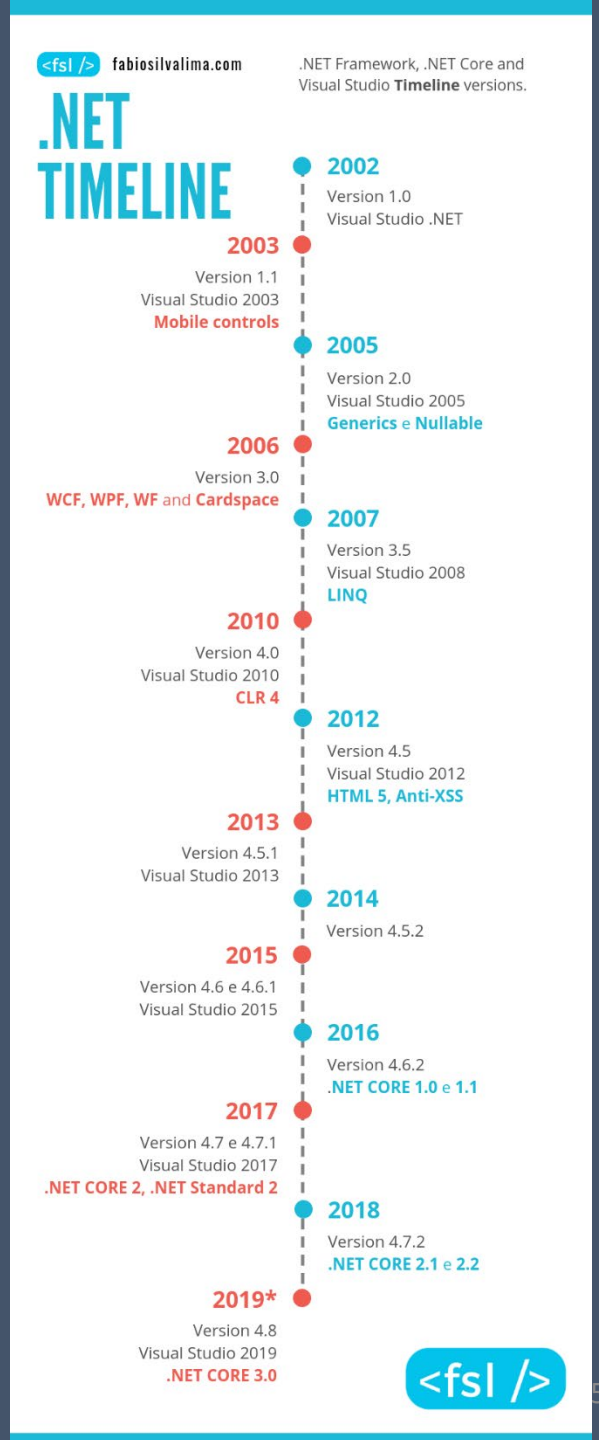
Goals of .Net

- Unify Programming Models
- Make developing easier and faster
- Be portable and safe
- Support multiple and future languages
- Incorporate web standards
- Replace COM and COM+



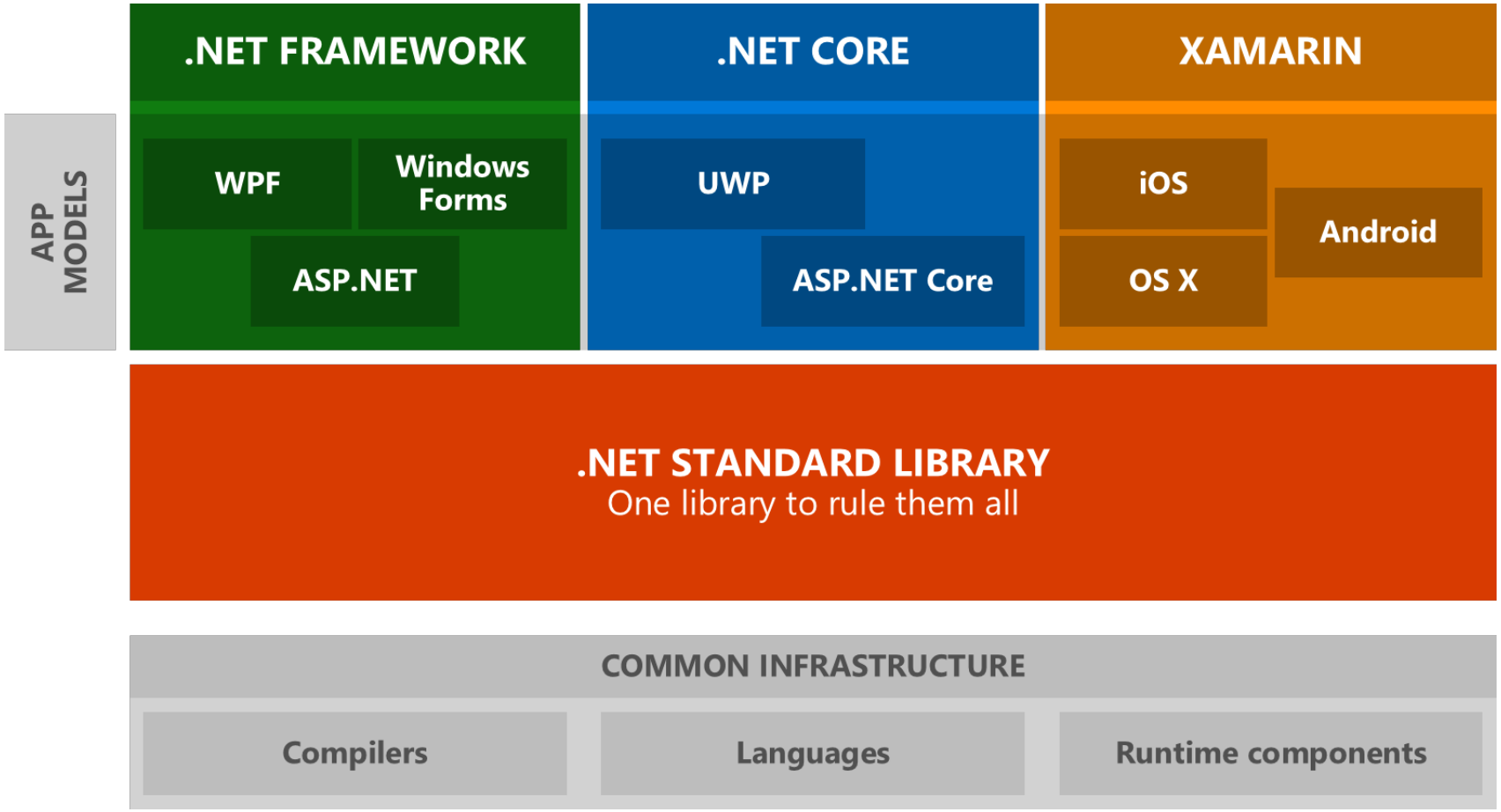
.NET Framework History

- .NET Framework, introduced in 2002 (Windows Only)
- .NET Core, introduced in 2016 (Multi-Platform)
- .NET Framework was "discontinued" in 2020
- .NET Core was discontinued in 2020
- .NET 5 was introduced in 2020
- .NET 5 replaces and is a convergence of the .NET Framework and .NET Core



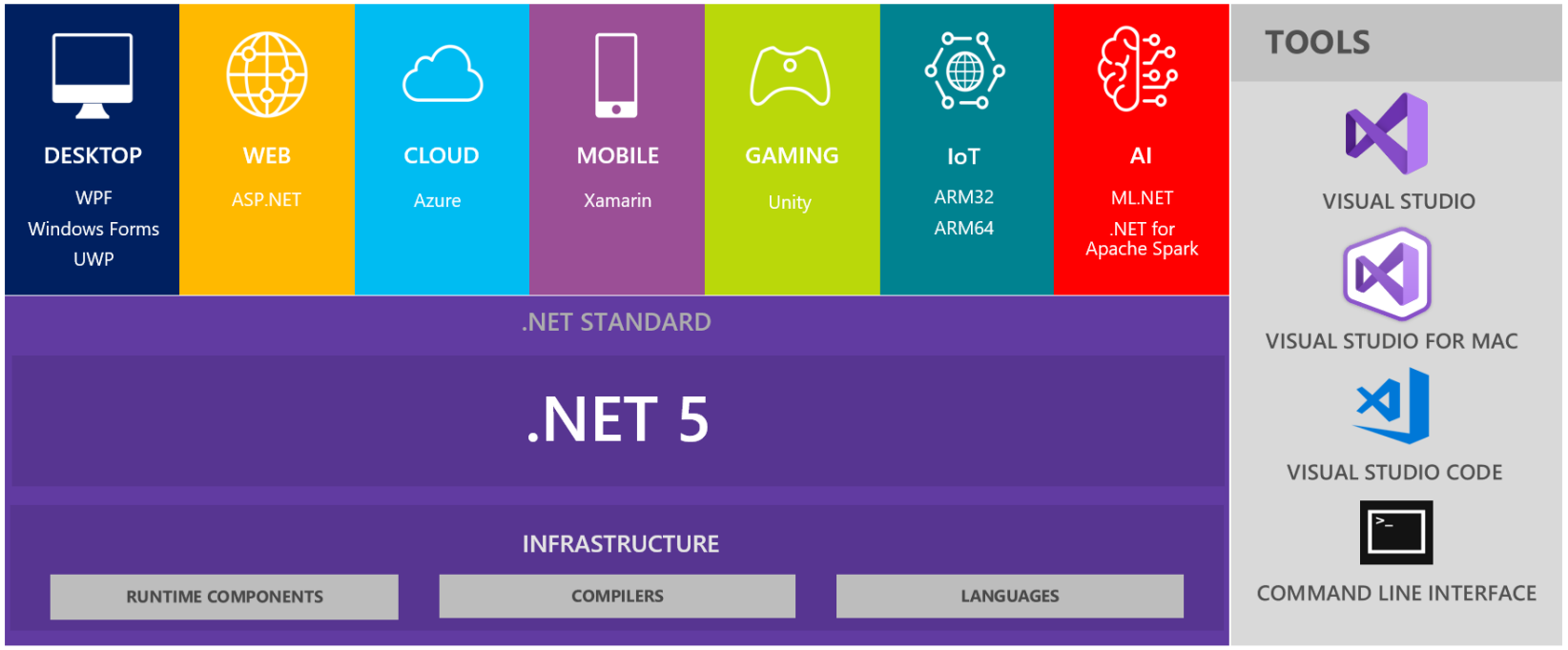
Source: https://www.fabiosilvalima.com/untold-story-net-versions/

.Net Current State



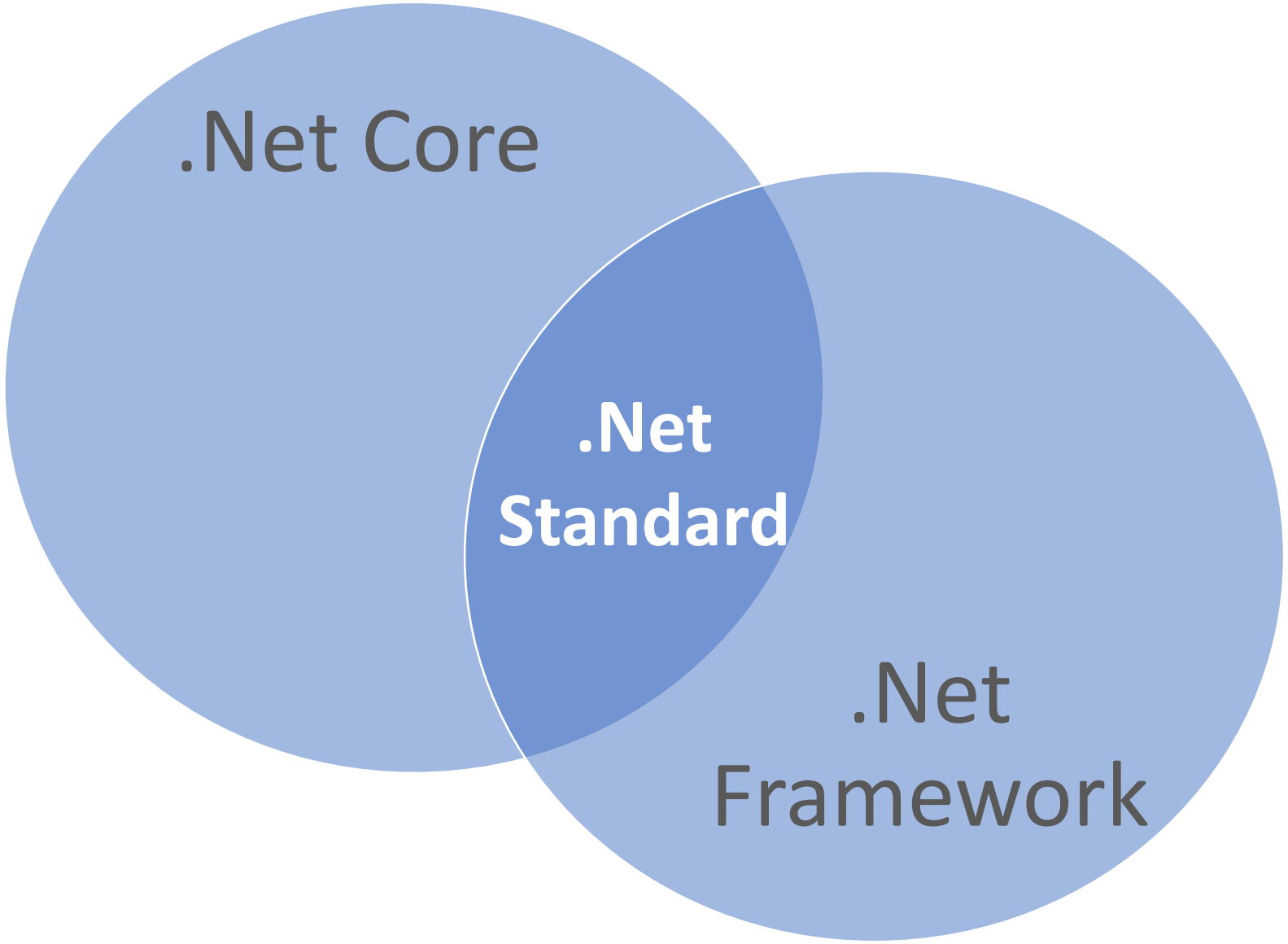
Source: https://docs.microsoft.com/en-us/dotnet/standard/library-guidance/cross-platform-targeting

.NET – A unified platform

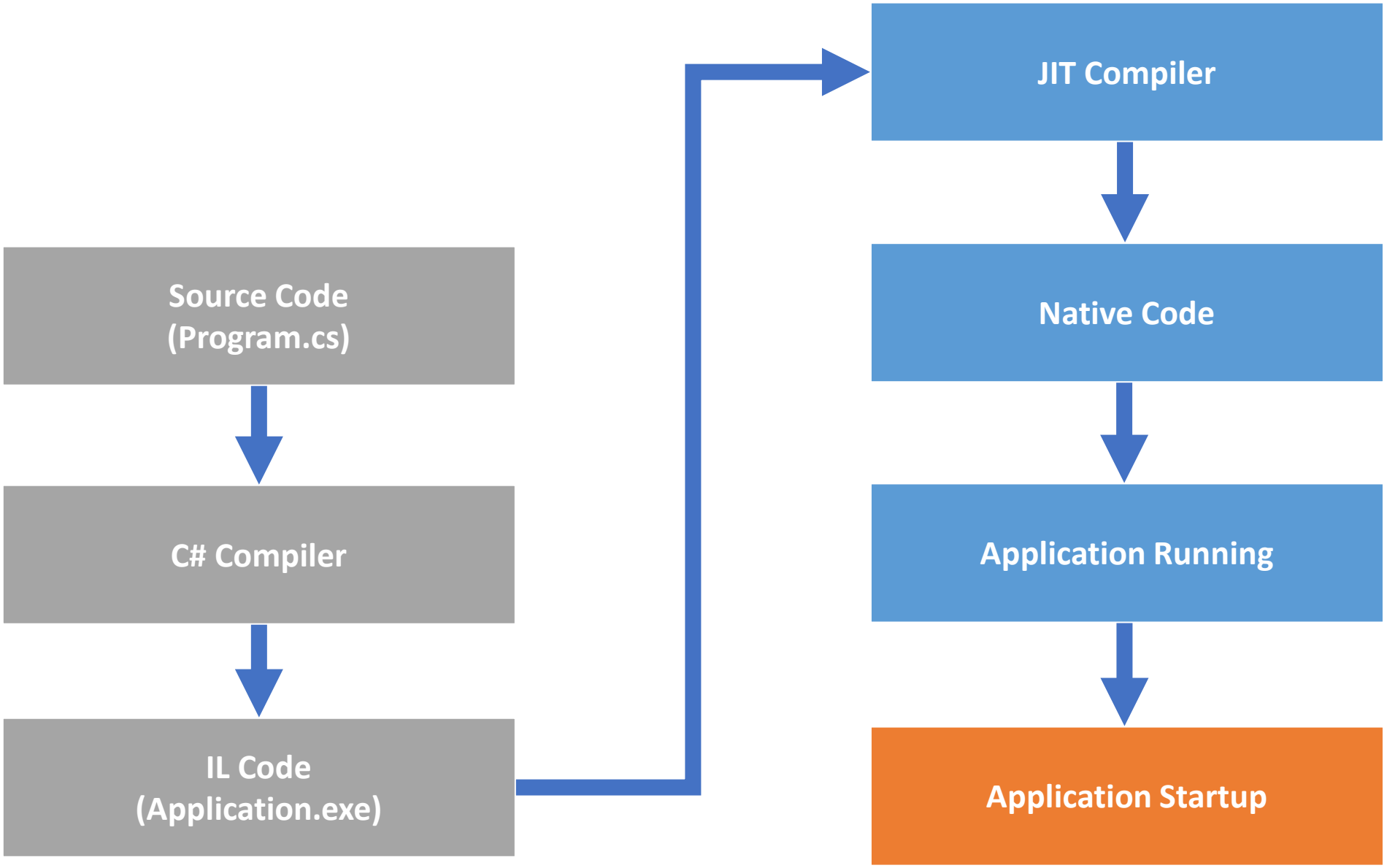


Source: https://devblogs.microsoft.com/dotnet/introducing-net-5/

.Net Standard



Compile and Run



Compile and Run – C# to MSIL (aka CIL)

```
public class Program
{
    public static void Main()
    {
        System.Console.WriteLine("Hello World");
    }
}
```

Results λ SQL IL Tree

```
Program.Main:
IL_0000:  nop
IL_0001:  ldstr       "Hello World"
IL_0006:  call        System.Console.WriteLine
IL_000B:  nop
IL_000C:  ret

Program..ctor:
IL_0000:  ldarg.0
IL_0001:  call        System.Object..ctor
IL_0006:  nop
IL_0007:  ret
```

Press Alt+Shift+R for a full decompilation.

Compile and Run – Assembly code generated by JIT Compiler

Disassembly | Program.cs

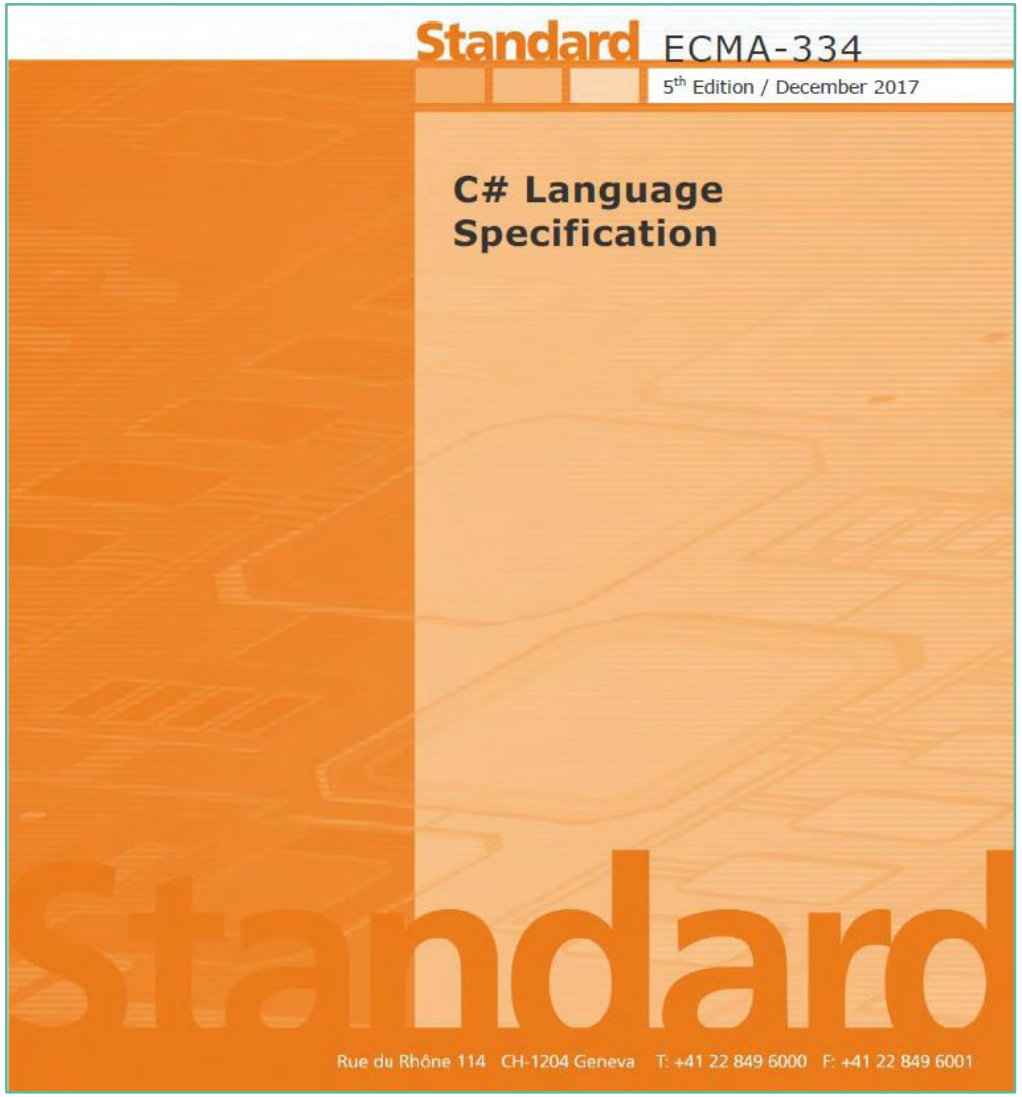
Address: ConsoleApp1.Program.Main(string[])

Viewing Options

```
00F7084D  push        ebx
00F7084E  sub         esp,30h
00F70851  xor         ebx,ebx
00F70853  mov         dword ptr [ebp-10h],ebx
00F70856  mov         dword ptr [ebp-1Ch],ebx
00F70859  mov         dword ptr [ebp-3Ch],ecx
00F7085C  cmp         dword ptr ds:[0D542F0h],0
00F70863  je          00F7086A
00F70865  call        53BAFCD0
00F7086A  nop
00F7086B  mov         ecx,dword ptr ds:[3AF2344h]
00F70871  call        52852FA8
00F70876  nop
00F70877  nop
00F70878  lea         esp,[ebp-0Ch]
00F7087B  pop         ebx
00F7087C  pop         esi
00F7087D  pop         edi
00F7087E  pop         ebp
```

99 %

The C# Language



- Is an open standard; ECMA-334
- Designed specifically for .NET
- C-based syntax; like C/C++ and Java
- Case-sensitive
- Code blocks delimited by { }
- Semi-colon statement terminators;

Hello World – C# Edition

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

Namespace **Class** **Method** **Parameter**

Numeric Data Types - Integral

C# Type Name	.NET Type Name	Value Range	Size (bytes)	Usage
byte	System.Byte	0 to 255	1	Avoid
sbyte	System.SByte	-128 to 127	1	Avoid
short	System.Int16	-32,768 to 32,767	2	Avoid
ushort	System.UInt16	0 to 65,535	2	**No**
int	System.Int32	-2,147,483,648 to 2,147,483,647	4	Yes
uint	System.UInt32	0 to 4,294,967,295	4	**No**
long	System.Int64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8	Yes
ulong	System.UInt64	0 to 18,446,744,073,709,551,615	8	**No**

Data Types – Primitive Types

- `string` is an alias for System.`String`
- `int` is an alias for System.`Int32`
- `float` is an alias for System.`Single`

- `string` is a reference type, but it acts like a value type in C#
  - E.g. we can use ==

Magic: Actual C# `Int32` (aka `int`) implementation:

```
    public struct Int32 : IComparable, IFormattable, IConvertible
        , IComparable<Int32>, IEquatable<Int32>
///     , IArithmetic<Int32>
#else
    public struct Int32 : IComparable, IFormattable, IConvertible
#endif
    {
        internal int m_value;

        public const int MaxValue = 0x7fffffff;
        public const int MinValue = unchecked((int)0x80000000);
```

How Data is Stored in a Byte

11

	MSB							LSB
Bit Position:	7	6	5	4	3	2	1	0
Value:	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
	0	1	0	0	0	0	0	1

Byte Value: **65**

ASCII Character: **A**

Integer Arithmetic

```
9   class Program
10  {
        0 references
11      static void Main(string[] args)
12      {
13          Console.Title = "Integer Arithmetic";
14
15          int opA = 10;
16          int opB = 3;
17          double opC = 3.0;
18
19          int resultA = opA / opB;      // 3
20          int resultB = opA % opB;      // 1
21
22          double resultC = opA / opB;   // 3
23          double resultD = opA / opC;   // 3.33333333333333333
24      }
25  }
```

- The result of dividing an int by an int, is always an int.
- Promoting just one of the operands to a double will result in a double.
- Division by zero will throw an exception

Numeric Data Types – Fractional

C# Type Name	.NET Type Name	Value Range	Precision (digits)	Size (bytes)	Usage
float	System.Single	-3.402823e+038 to 3.402823e+038	8	4	Avoid
double	System.Double	-1.797693e+308 to 1.797693e+308	15	8	Yes
decimal	System.Decimal	-7.922816e+028 to 7.922816e+028	28	16	Financial

Decimal – Always Use for Financial Calculations

```
Console.Title = "Decimal Data Type";

double doublePenny = 0.01d;

double doubleDime = doublePenny + doublePenny + doublePenny + doublePenny + doublePenny
                  + doublePenny + doublePenny + doublePenny + doublePenny + doublePenny;

double doubleDollar = doubleDime + doubleDime + doubleDime + doubleDime + doubleDime
                    + doubleDime + doubleDime + doubleDime + doubleDime + doubleDime;

bool doubleEqual = doubleDollar == 1.00d;
Console.WriteLine("Equal (double): {0}", doubleEqual);


decimal decimalPenny = 0.01m;

decimal decimalDime = decimalPenny + decimalPenny + decimalPenny + decimalPenny + decimalPenny
                    + decimalPenny + decimalPenny + decimalPenny + decimalPenny + decimalPenny;

decimal decimalDollar = decimalDime + decimalDime + decimalDime + decimalDime + decimalDime
                      + decimalDime + decimalDime + decimalDime + decimalDime + decimalDime;

bool decimalEqual = decimalDollar == 1.00m;
Console.WriteLine("Equal (decimal): {0}\n\n", decimalEqual);
```

Decimal Data Type

```
Equal (double): False
Equal (decimal): True
```

- Special type designed to represent fractional values with high precision
- Always use decimal for financial values
- Optimized for base-10 values
- Literals with decimals default to double, so need to specify decimal with “m” suffix

Math Class

```
11  static void Main(string[] args)
12  {
13      Console.Title = "Math Class";
14
15      double result01 = Math.Sin(30 * Math.PI / 180); // 0.4999
16      double result02 = Math.Cos(30 * Math.PI / 180); // 0.8660
17      double result03 = Math.Tan(30 * Math.PI / 180); // 0.5774
18
19      double result04 = Math.Sin(45 * Math.PI / 180); // 0.7071
20      double result05 = Math.Cos(45 * Math.PI / 180); // 0.7071
21      double result06 = Math.Tan(45 * Math.PI / 180); // 0.9999
22
23      double result07 = Math.Sin(60 * Math.PI / 180); // 0.8660
24      double result08 = Math.Cos(60 * Math.PI / 180); // 0.5000
25      double result09 = Math.Tan(60 * Math.PI / 180); // 1.7321
26
27      double result10 = Math.Sqrt(2);                 // 1.4142
28      double result11 = Math.Abs(15);                 // 15
29      double result12 = Math.Abs(-15);                // 15
30
31      double result13 = Math.Pow(3.446, 2.11245);     // 13.647454299617072
32      int result14 = (int)Math.Pow(4, 2);             // 16 is this the best way to calculate
33                                                      //    the square of a number?
34  }
```

- A set of static methods to perform mathematical calculations.
- Is Math.Pow the best way to calculate the square of a number?

Operators and Precedence

$$1 + 2 \times 3 = ?$$

Operators and Precedence

1 + 2 X 3 = ?

The answer is 7

Why?

Operator precedence - Multiplication Before Addition
Step 1 – Multiply 2 X 3 = 6
Step 2 – Add 1 + 6 = 7

To force an alternate order – use parentheses ()

(1 + 2) X 3

Step 1 – Add 1 + 2 = 3
Step 2 – Multiply 3 X 3 = 9

Operators and Precedence

$$8 \div 2(2+2) = ?$$

$8 \div 2(2+2) = ?$

The answer is 16

Why?

Operator precedence Parentheses first
Step 1 – Add (2 + 2) = 4
Step 2 – Calculate 8 ÷ 2 x 4
Step 3 – Calculate 4 x 4 = 16

The author probably meant 8 / 2(2+2), which would equal 1. This could be better written as (8/2)(2+2).
Moral of the story? **Err on the side of clarity**

Operators and Precedence

Category	Operators	Associativity
Primary	x.y f(x) a[x] x++ x-- new typeof checked unchecked	Left to right
Unary	+ - ! ~ ++x --x (T)x	Left to right
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational and type testing	< > <= >= is as	Left to right
Equality	== !=	Left to right
Logical AND	&	Left to right
Logical XOR	^	Left to right
Logical OR	|	Left to right
Conditional AND	&&	Left to right
Conditional OR	||	Left to right
Conditional	?:	Right to left
Assignment	= *= /= %= += -= <<= >>= &= ^= |=	Right to left

Console Class – Write and WriteLine Methods

```
13
14      Console.Title = "Console IO";
15
16      Console.Write("Enter your name: ");
17      string name = Console.ReadLine();
18
19      Console.WriteLine("You entered {0}.", name);
20
```

- Static methods
- Returns void
- Prints characters to the Console
- WriteLine moves the cursor to the next line
- Write leaves the cursor on the current line

Console Class – ReadLine Method

```
13
14      Console.Title = "Console IO";
15
16      Console.Write("Enter your name: ");
17      string name = Console.ReadLine();
18
19      Console.WriteLine("You entered {0}.", name);
20
```

- Static method
- Returns a string
- Takes no arguments
- Reads a line of characters entered in the Console

Console Title

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.Title = "Numeric Data Types";

        string formatStringHeading = "{0, -8} {1, 28} {2,
        string formatStringIntegral = "{0, -8} {1, 28:N0}
        string formatStringFloating = "{0, -8} {1, 28:E} {

        Console.WriteLine(formatStringHeading,
                "C# Data",
                "Minimum",
                "Maximum",
                 Size");
```

- Console.Title
- Static property

```
Numeric Data Types
C# Data                          Minimum                          Maximu
Type                               Value                            Valu
-------------------------------------------------------------------------
byte                                   0                              25
sbyte                               -128                              12
```

Composite Strings

```
13
14      string name = "Jennifer";
15      int messageCount = 1;
16
17      Console.WriteLine("Hello {0}, you have {1} messages.", name, messageCount);
18
19
```

- Combines variable data with static text in a convenient and easily controlled way.
- *Is this code ready for deployment?*

Composite Strings

```
13
14      string name = "Jennifer";
15      int messageCount = 1;
16
17      Console.WriteLine("Hello {0}, you have {1} message(s).", name, messageCount);
18
19
```

- No, the output is grammatically incorrect when there is one message
- A common solution is shown above
- Is this an appropriate way to solve this problem?

Composite Strings

```
13
14      string name = "Jennifer";
15      int messageCount = 1;
16
17      Console.WriteLine("Hello {0}, you have {1} message{2}.",
18                        name,
19                        messageCount,
20                        messageCount == 1 ? "" : "s");
21
```

- Here is a better solution
- The additional effort required for this solution is minimal
- This is a small detail; software development is all about details
- The work required to do things properly is no more than the work required to do things improperly

Composite Strings

```
38      // Create a literal, properly punctuated string first
39      Console.WriteLine("Hello Gillian, you have 3 messages.");
40
41      // Substitute variable text with placeholders
42      Console.WriteLine("Hello {0}, you have {1} messages.", name, numMessages);
43
```

- To make things easier:
  - Create a literal, properly punctuated string first
  - *Then* substitute the variable data positions with placeholders

Composite Strings – Column Widths and Alignment

```
28
29      foreach (SystemUser user in users)
30      {
31          Console.WriteLine("{0, -10} {1, 10}  {2}",
32                            user.Name,
33                            user.AverageLoginsPerWeek,
34                            user.Type);
35      }
36
```

Composite String Formatting

```
Average Logins Per Week
-----------------------------------
Name          Average  User Type
-----------------------------------
Geoff            15.2  Administrator
Bridgette        3.09  Administrator
Andrew          44.45  PowerUser
Brenda              1  User
Natalie             0  User
```

- {index [, alignment]}
  Example: **{0, 10}**
- Alignment is an int value indicating the preferred width of the column in characters
- Positive alignment values right justify
- Negative alignment values left justify

Textual data are usually left justified
Numeric data are usually right justified

Composite Strings – Formatting

```
28
29      foreach (SystemUser user in users)
30      {
31          Console.WriteLine("{0, -10} {1, 10:F2}  {2}",
32                            user.Name,
33                            user.AverageLoginsPerWeek,
34                            user.Type);
35      }
36
```

Composite String Formatting

```
Average Logins Per Week
-----------------------------------
Name          Average  User Type
-----------------------------------
Geoff           15.20  Administrator
Bridgette        3.09  Administrator
Andrew          44.45  PowerUser
Brenda           1.00  User
Natalie          0.00  User
```

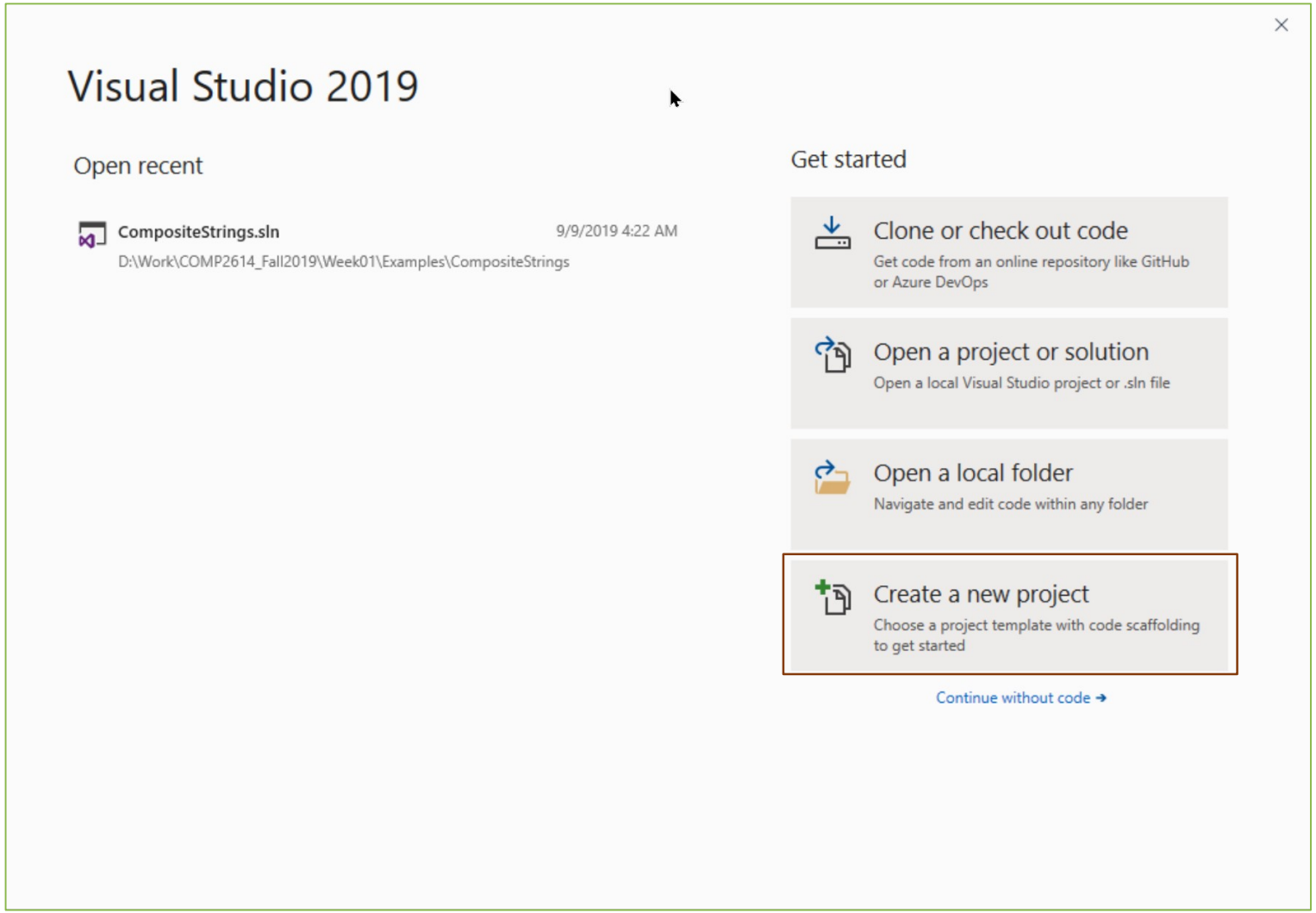
- {index [, alignment] [:format]}
  Example: **{0, 10:F2}**
- Format is a symbol or string indicating how the value should be formatted
- Alignment is optional:
  Example: **{0:F2}**

("{0, -10} {1, 10:F2} {2:F2} {3}")

C or c	currency (shows symbol)
D or d	base 10 (decimal) integers
E or e	scientific (exponential)
F or f	fixed point (to fix floating point)
N or n	basic formatting (with commas)

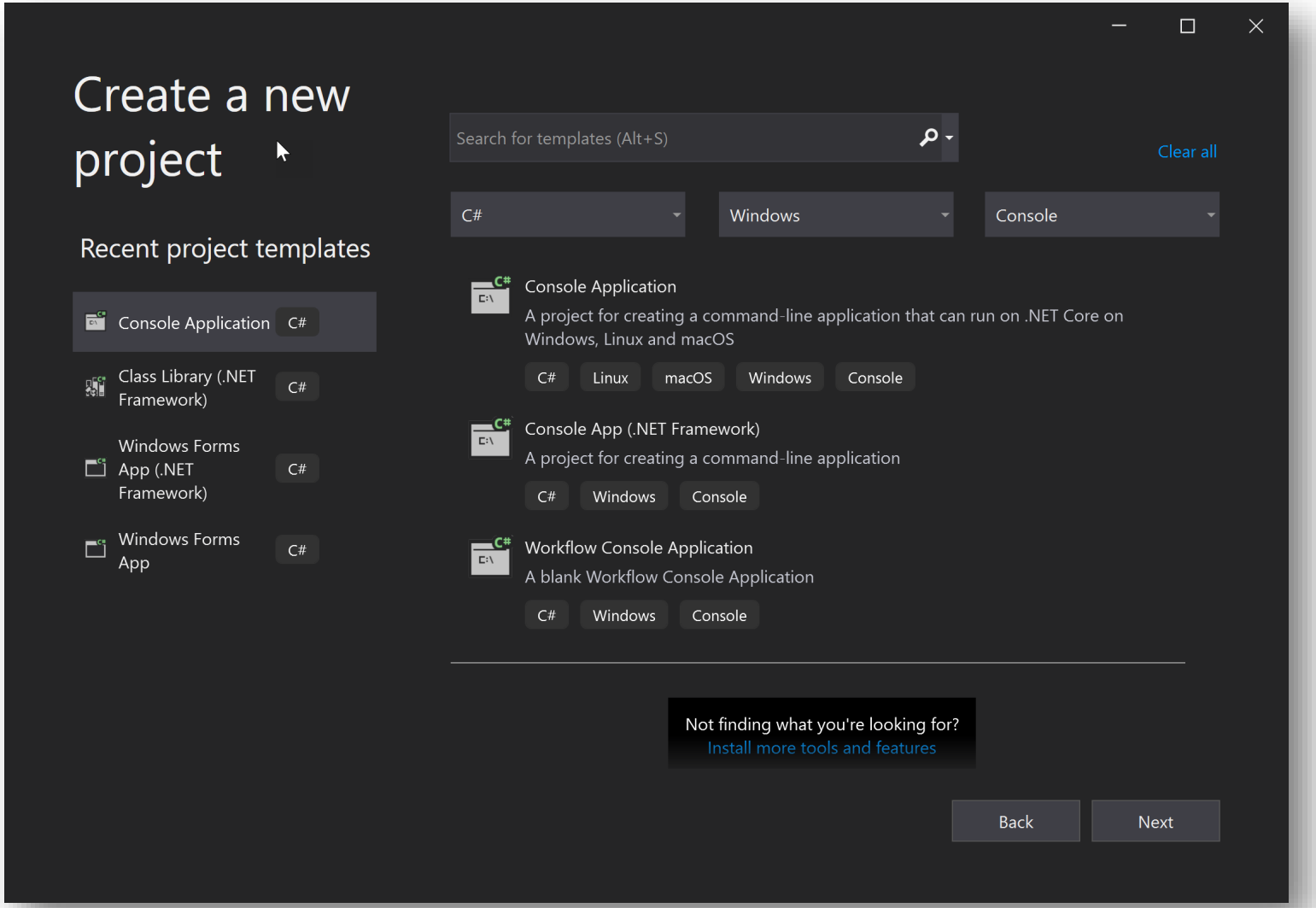
(see Online Help for others)

Visual Studio 2019 – Create New Project



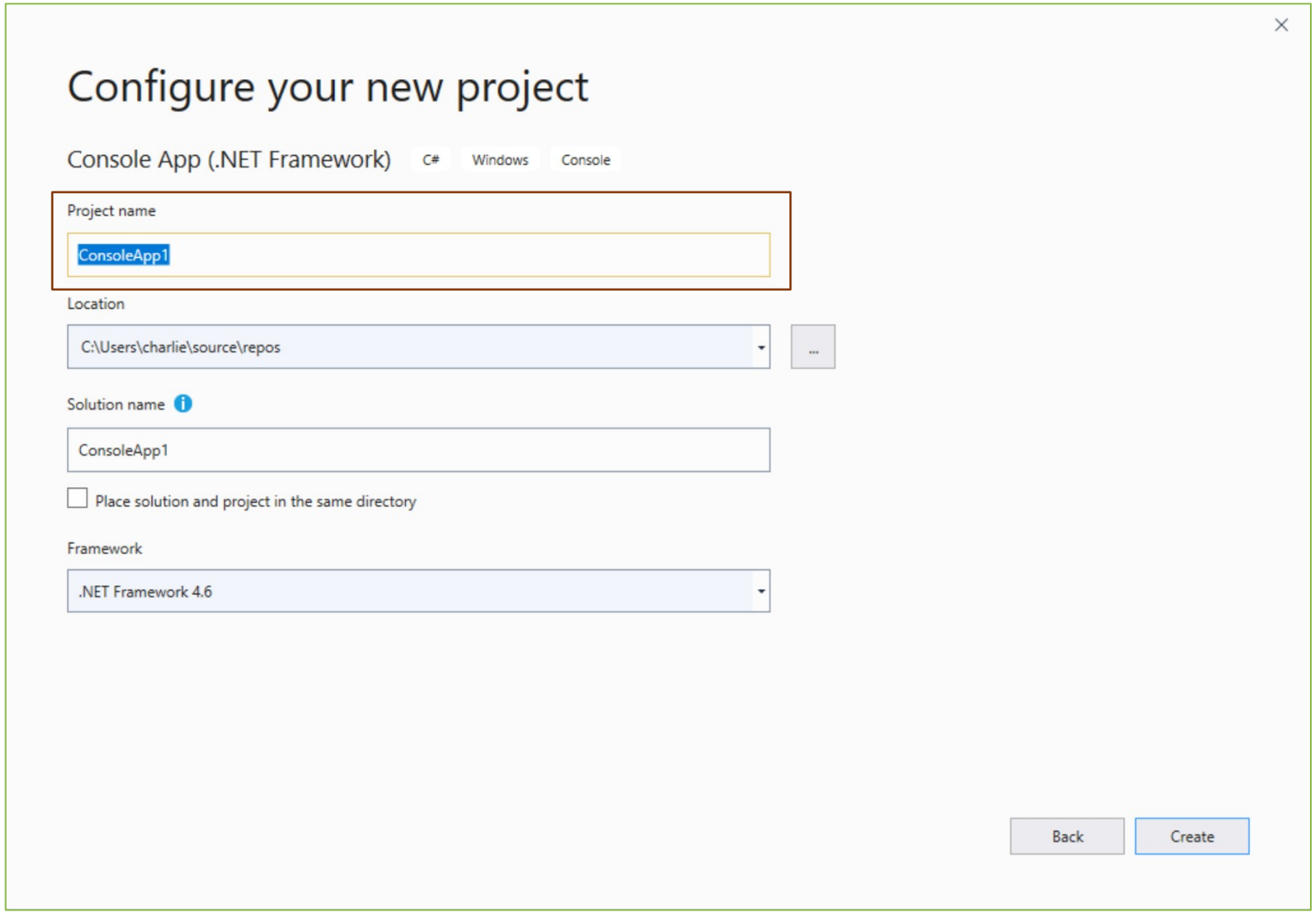
- Click Create a new project

Visual Studio 2019 – Create New Project



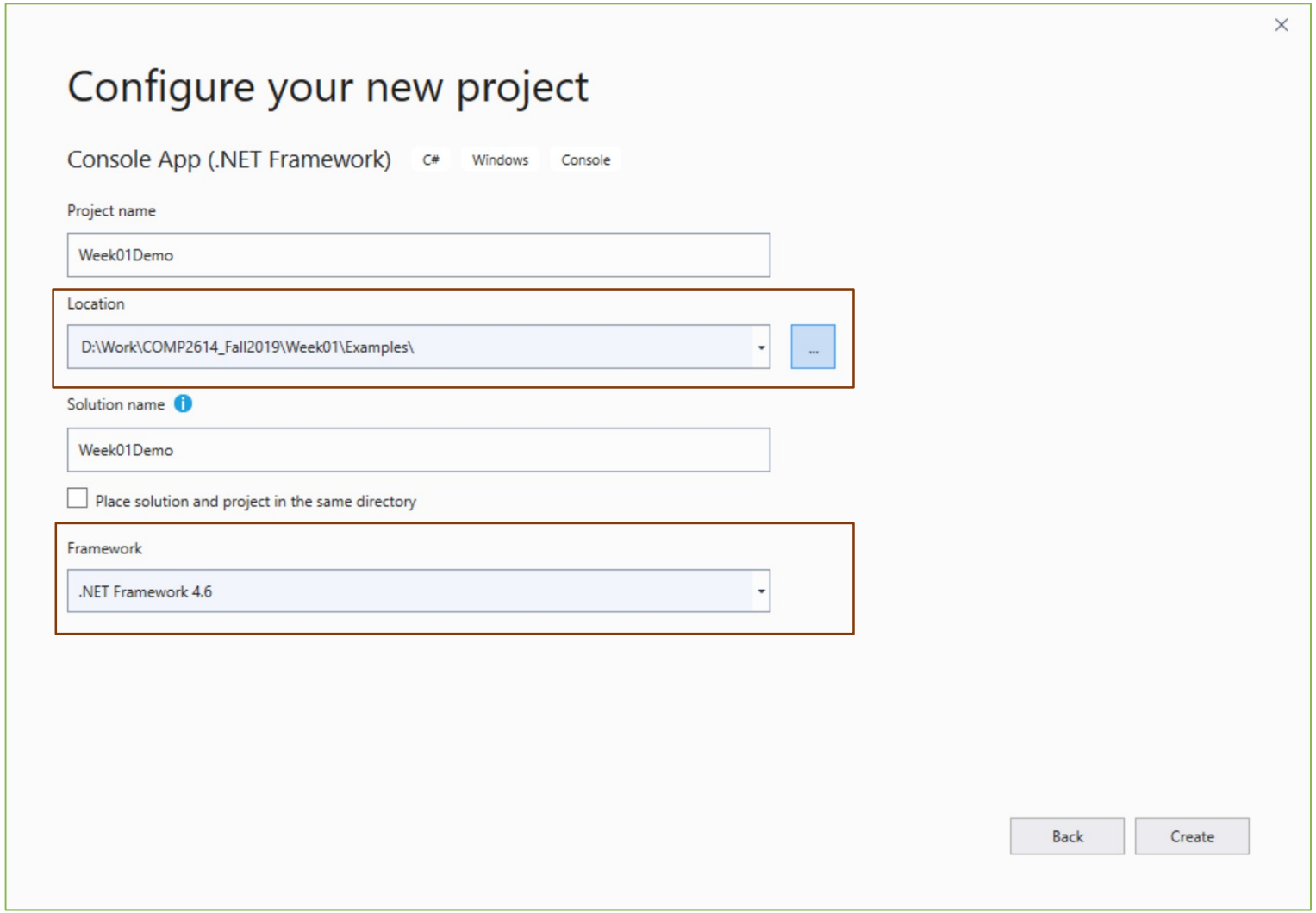
- Either search or use the filters to narrow the selection down to Console App
- .Net Framework is recommended, but .Net Core/5 are also OK until Week07.
- Click Next

Visual Studio 2019 – Create New Project



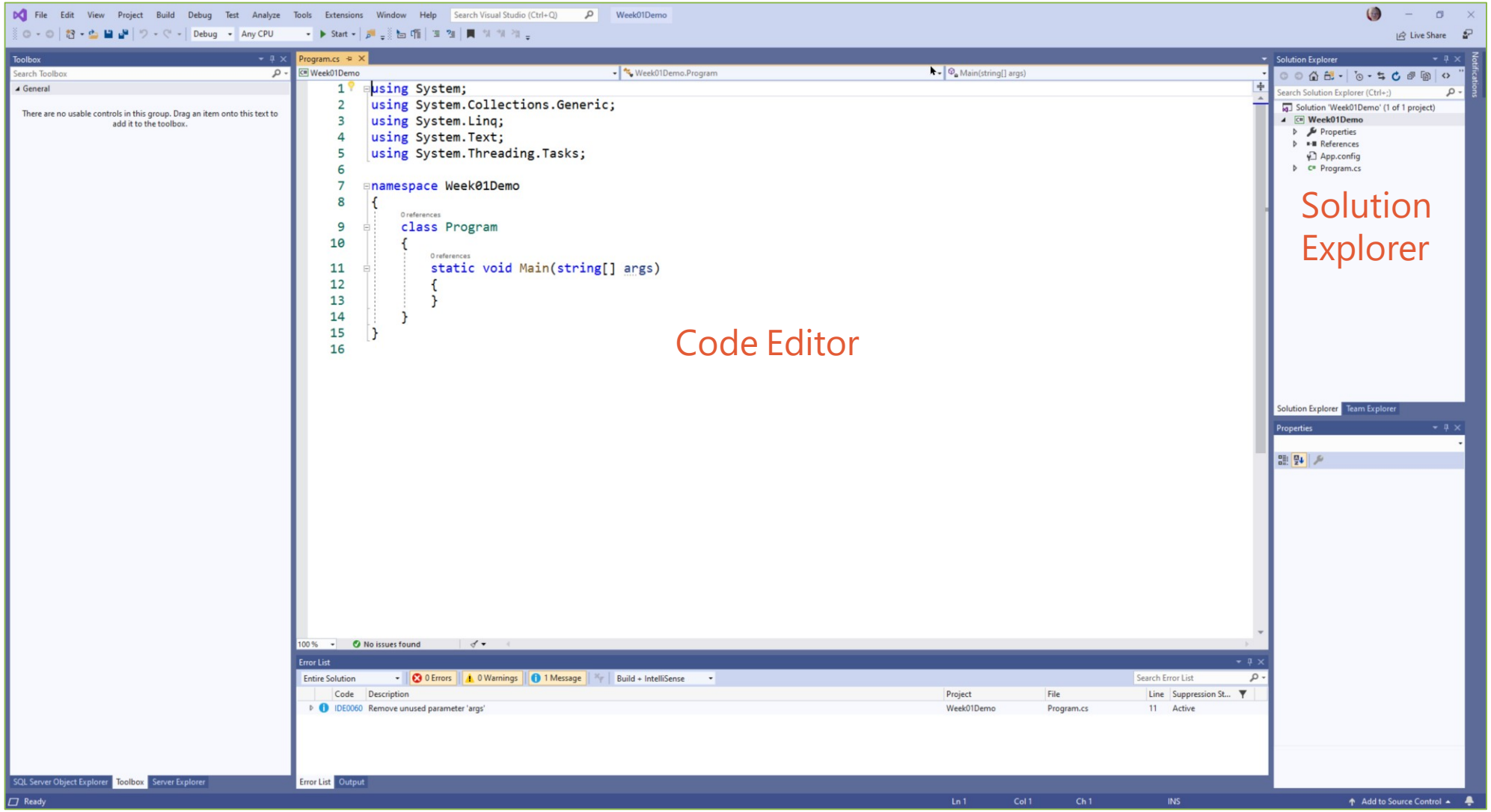
- Give your project a meaningful name (Pascal Case)
- Solutions with only one project have matching names by convention
- Never check the box to “Place solution and project in the same directory”

Visual Studio 2019 – Create New Project

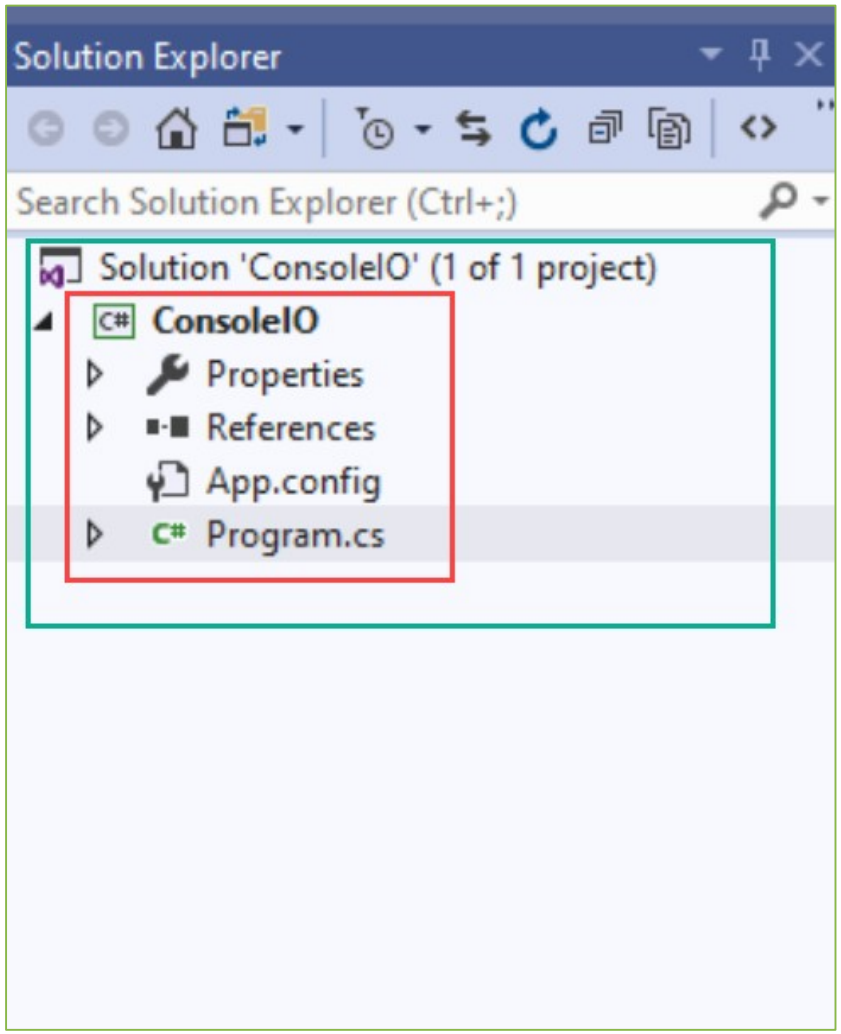


- Navigate to a predetermined location for your project
- Leave Solution name (Defaults to project name)
- Select .NET Framework 4.7 (or whatever is most current)
- Click Create

Visual Studio 2019 – Create New Project

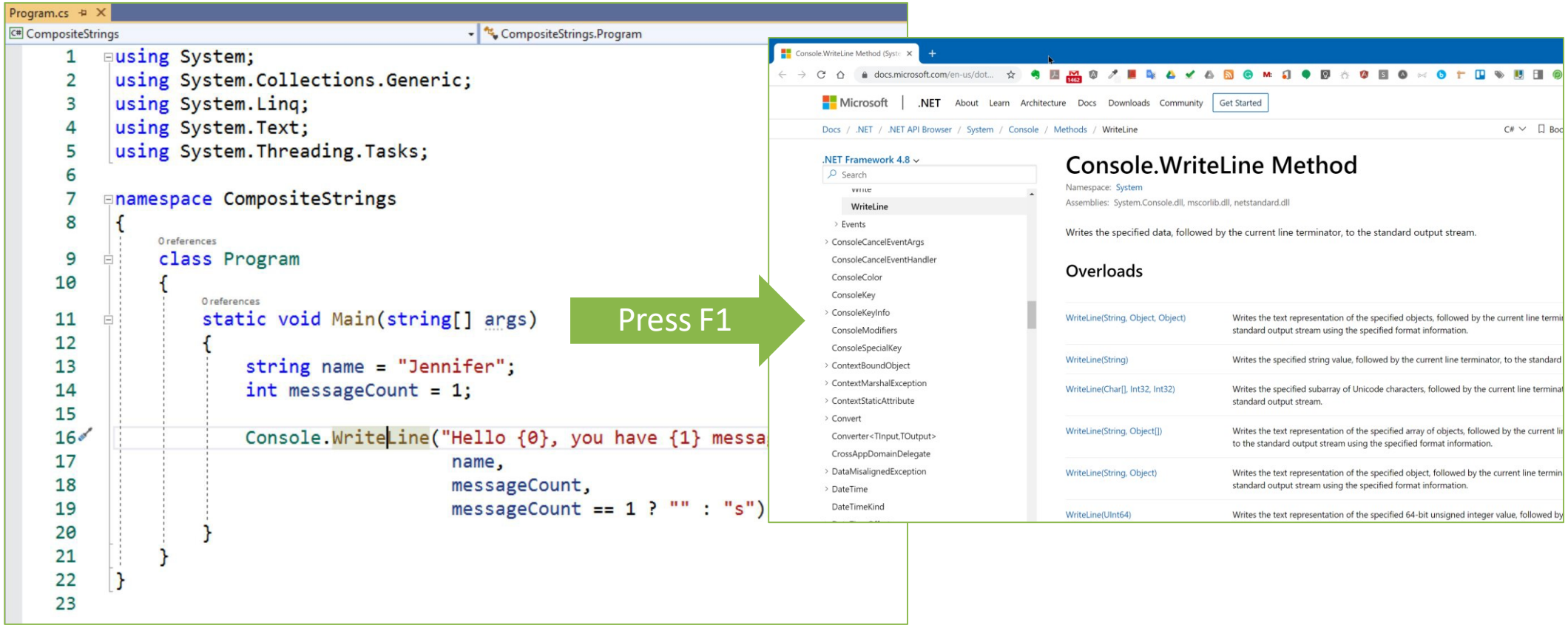


Visual Studio Solutions and Projects



- A Solution is an outer container which can hold multiple projects
- A Project is a container which holds all the assets of a program
  (Source Files, Images, etc.)
- Projects are nested inside a Solution
- When a Solution is opened, all nested Projects are opened as well
- A Solution can be opened by accessing its .sln file

Visual Studio 2019 – Online Help



1) Select a keyword in the code editor
2) Press F1
3) Select Online if prompted for a preference

Inline Comments

- ***Rule of thumb:***
- Comment what the code ***should*** be doing
- Not what it ***is*** doing (unless it is unintuitive/complex)

Files

Extension	Description
.exe	Compiled, executable code in MSIL
.dll	Dynamic link library – compiled code in MSIL (class library)
.sln	Solution file – XML file that stores settings and configuration of the solution
.suo	Solution user options file – stores user-specific settings for the solution
.csproj	C# project file – xml file that stores settings and configuration for the project
.config	XML configuration file that is read/writable to your application (with helpful framework library for this purpose)
.pdb	Program database – debugging information created for debug builds
.resx	Resources file – used for multiligual apps

Folders

Folder Path	Description	Add to Source Control
…/repos/<Solution>/	Main solution directory	Yes
…/repos/<Solution>/.**vs**/	User and environment specific settings and working files	No
…/repos/<solution>/<project>/**bin**/	Compiler output directory	Yes
…/repos/<solution>/<project>/bin/<build type – e.g. **debug**>/	Compiler output directory for a given build configuration. This is where compiled .exe and .dll files go.	No
…/repos/<solution>/<project>/**obj**/	Compiler working directory	No
…/repos/<solution>/<project>/**properties**/	Properties for the compiled assembly	Yes