COMP4701: Web Application Development

Fall 2025

Dr. Abdullah Al-Hamdani

1- .Net Framework and C# Basics

- .NET Frameworks
- .Net Programming Language
- Common Language Runtime (CLR)
- Intermediate Language (IL)
- .Net History

Web Pages

- **Static Web Pages:**
  - **Content doesn't change when requested by a user**
- **Dynamic Web Pages:**
  - **Content changes depending on the user's request and preferences**
    - **Client-Side: Executed in the browser**
    - **Server-Side: Executed in the server**

HTML

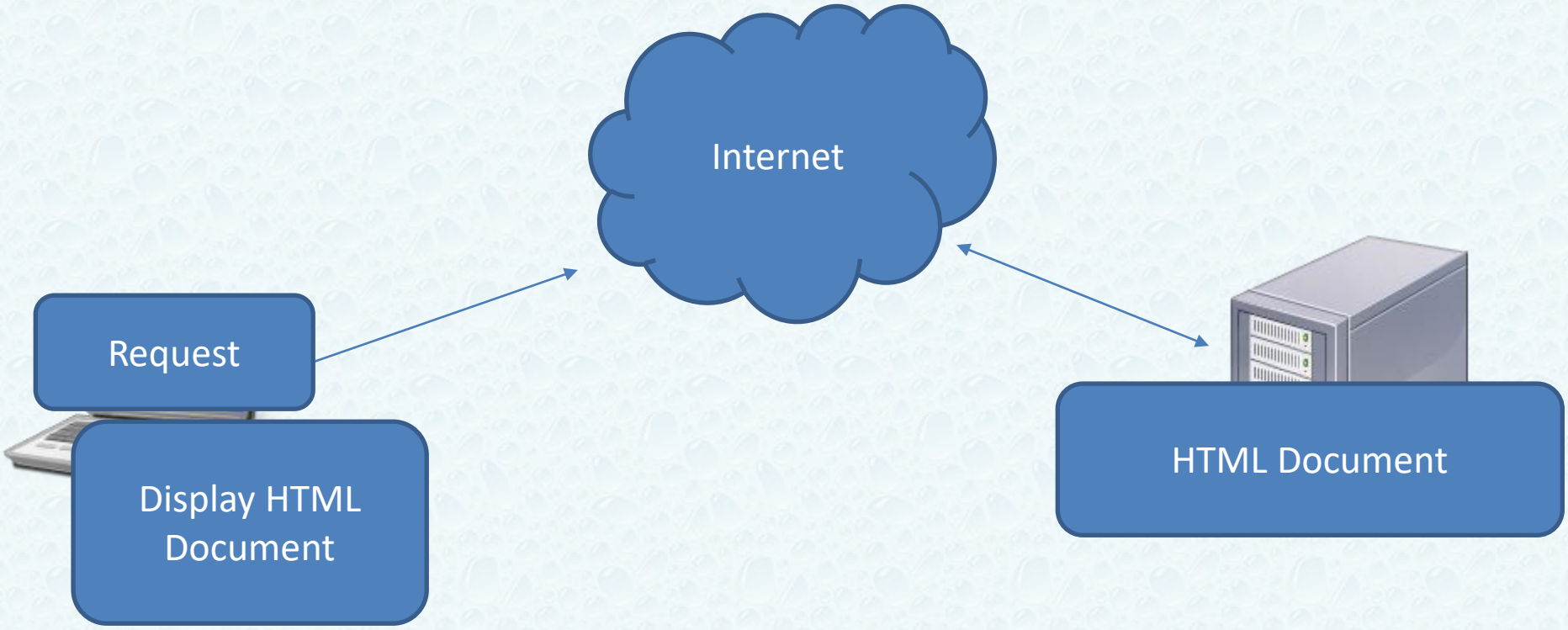
- Early websites were not *web applications*
- HTML only displays the information
- HTML has two types of contents: Text and Tags
- Saved in a fixed file (for example index.html)
- No interactivity – needed to be updated by hand
- Doesn’t require a web server

HTML Example

```
<html>
    <head>
        <title>Sample Web Page</title>
    </head>
    <body>
        <h1>Sample Web Page Heading</h1>
        <p>This is a sample web page.</p>
    </body>
</html>
```

HTML Example

An HTML Request



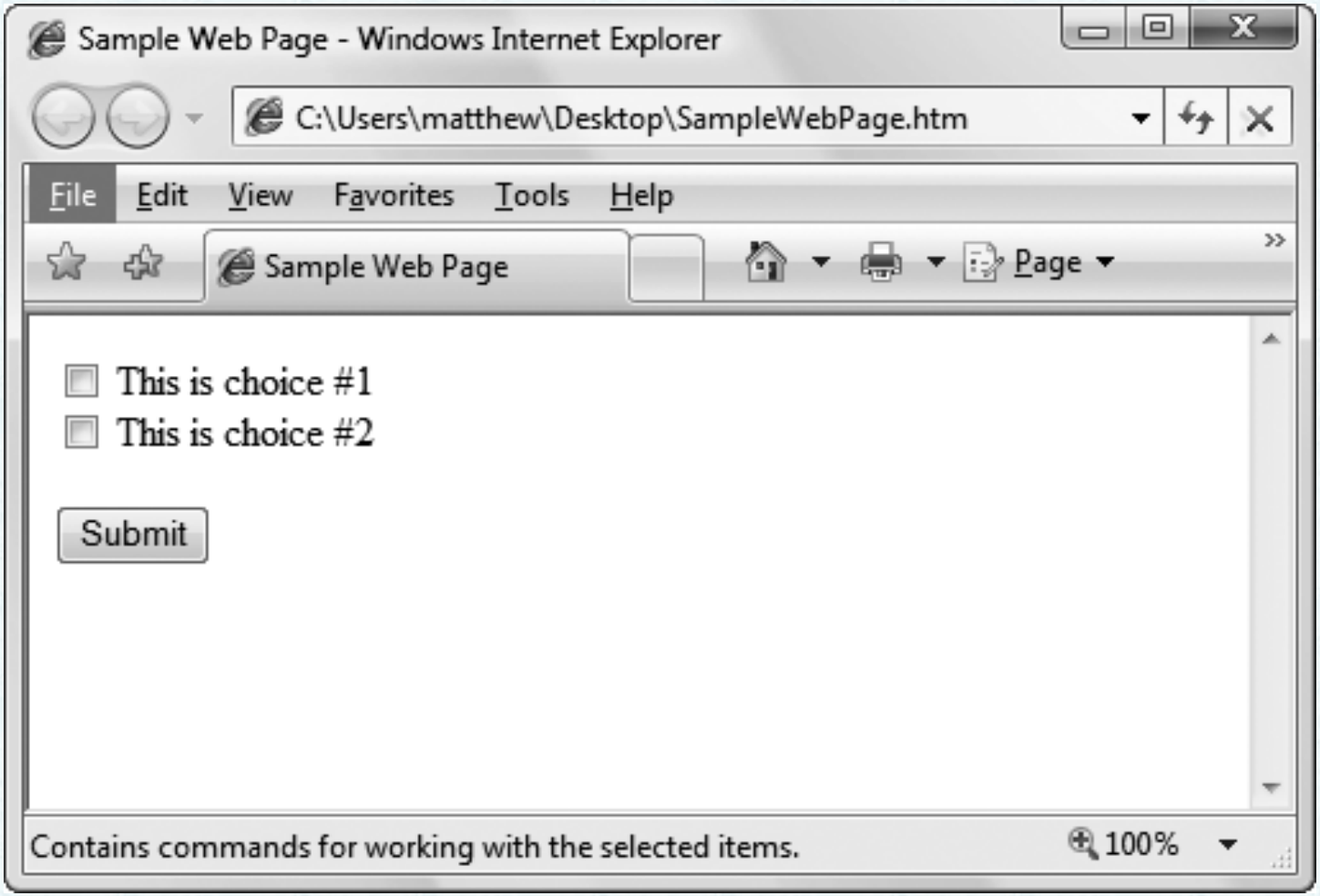
Browsers

- Examples: Internet Explorer, Firefox, Opera, Google Chrome, Safari, Netscape, Camino, etc.
- Browsers display HTML pages
- Different browsers display the same page differently
- Incorrect HTML may be displayed weirdly or not at all in some browsers
- There are different versions of browsers for mobile devices too

HTML Forms

- Introduced by HTML 2.0
- Graphical widgets or *controls* (text boxes, buttons, drop-down lists, etc.) are added
- These controls should be put between <form> and </form> tags
- Standard input pages can be designed
- Input data are sent to web server
- Web server processes the data

HTML Forms Example



HTML Forms Example

```
<html>
   <head>
      <title>Sample Web Page</title>
   </head>
   <body>
      <form>
         <input type="checkbox" /> This is choice #1<br />
         <input type="checkbox" /> This is choice #2<br /><br />
         <input type="submit" value="Submit" />
      </form>
   </body>
</html>
```

Methods of Sending Information

- The Form object may be used to send the data to another page using "action" attribute:

```
<form action="Another_Page.php">
  <input type="checkbox" /> This is choice #1<br />
  <input type="checkbox" /> This is choice #2<br /><br />
  <input type="submit" value="Submit" />
</form>
```

- Form object can send values in two separate methods: POST and GET:

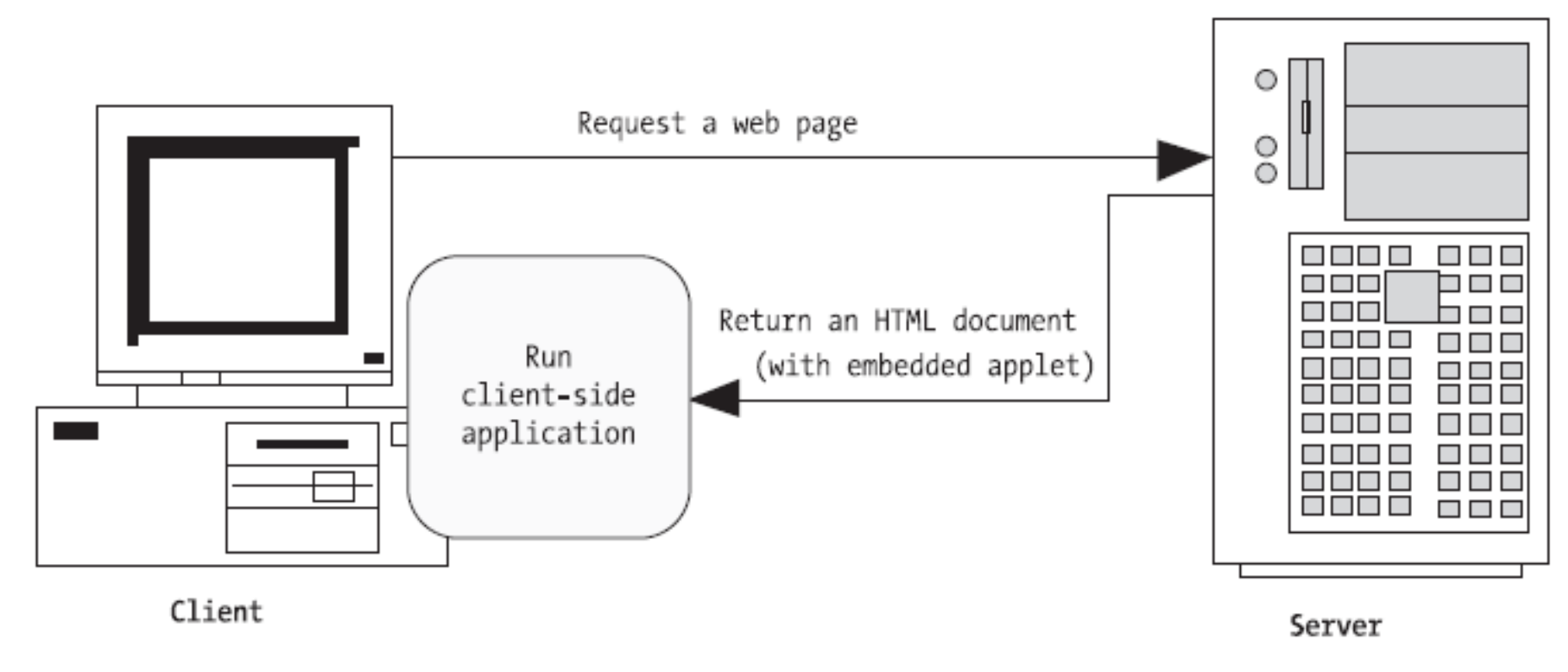
```
<form method="GET">...</form>
<form method="POST">...</form>
```

Methods of Sending Information

- **POST**
  - Default method
  - User data is combined in a special form and sent to the server
- **GET**
  - User data is added to the end of the URL address
  - Data is sent as an encoded stream
  - Example:

https://www.google.com/search?q=oman&hl=en

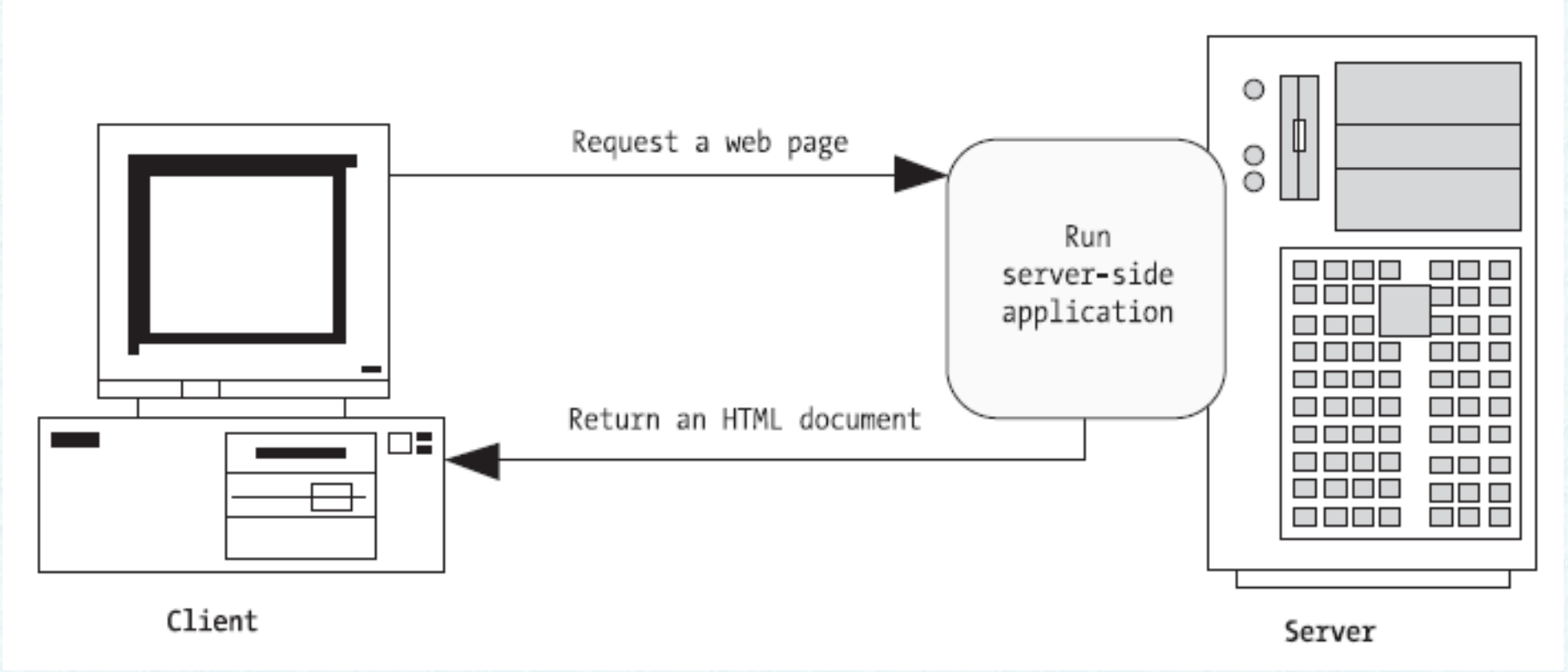
A Client-Side Web Application



Server-Side Programming

- Executed on the server side
- Early web development technologies used CGI
  - In CGI, web server launch a completely separate instance of the application for each web request
- If the website is popular, the web server may be down easily
- If higher-level features are needed (e.g. user authentication), the pages of codes should be written from scratch
- Building web Application, this way is boring and error-prone
- To counter these problems, high-level development platforms are created, such as PHP, JSP, ASP, ASP.NET and ASP.NET Core

A Server-Side Web Application



Classic ASP Example

```
<html>
  <body>

    <% Response.Write(“<p>Hello World!</p>") %>

    <% Response.Write("<p
    style='color:#0000ff'>Hello World!</p>") %>

  </body>
</html>
```

PHP Example

```
<html>
  <body>

    <?php echo “</p>Hello World!</p>"; ?>

    <?php echo "<p style='color:#0000ff'>Hello
    World!</p>"; ?>

  </body>
</html>
```

Classic ASP vs. ASP.NET

- ASP is a script-based programming language that requires a thorough understanding of HTML and a good deal of painful coding

- ASP.NET is an object-oriented programming model that lets developers to develop websites as easily as Windows applications

ASP.NET

- A server-side technology
- All ASP.NET codes are executed on the web server
- When the code is finished executing, the user receives an ordinary HTML page, which can be viewed in any browser

Why Use Server-Side Programming Instead of Client-Side?

- **Isolation:** Client-side code can't access server-side resources
- **Security:** End users can view client-side code
- **Thin clients:** Mobile phones, PDAs, etc. don't support all features of a traditional browser

Best Solution?

- Combining server-side and client-side programming is the best solution
- ASP.NET controls can detect capabilities of the client browser
- If browser supports JavaScript, these controls return a web page that incorporates JavaScript
- Ajax technology can be used with ASP.NET effectively
- However, ASP.NET codes are always executed on the server

The .NET Framework

- .NET Framework is composed of a cluster of several technologies:
  - .NET Languages:

    C#, Visual Basic.NET, JScript.NET, J#, C++, Java, ……
  - The Common Language Runtime (CLR):

    The engine that executes all .NET programs
  - The .NET Framework Class Library:

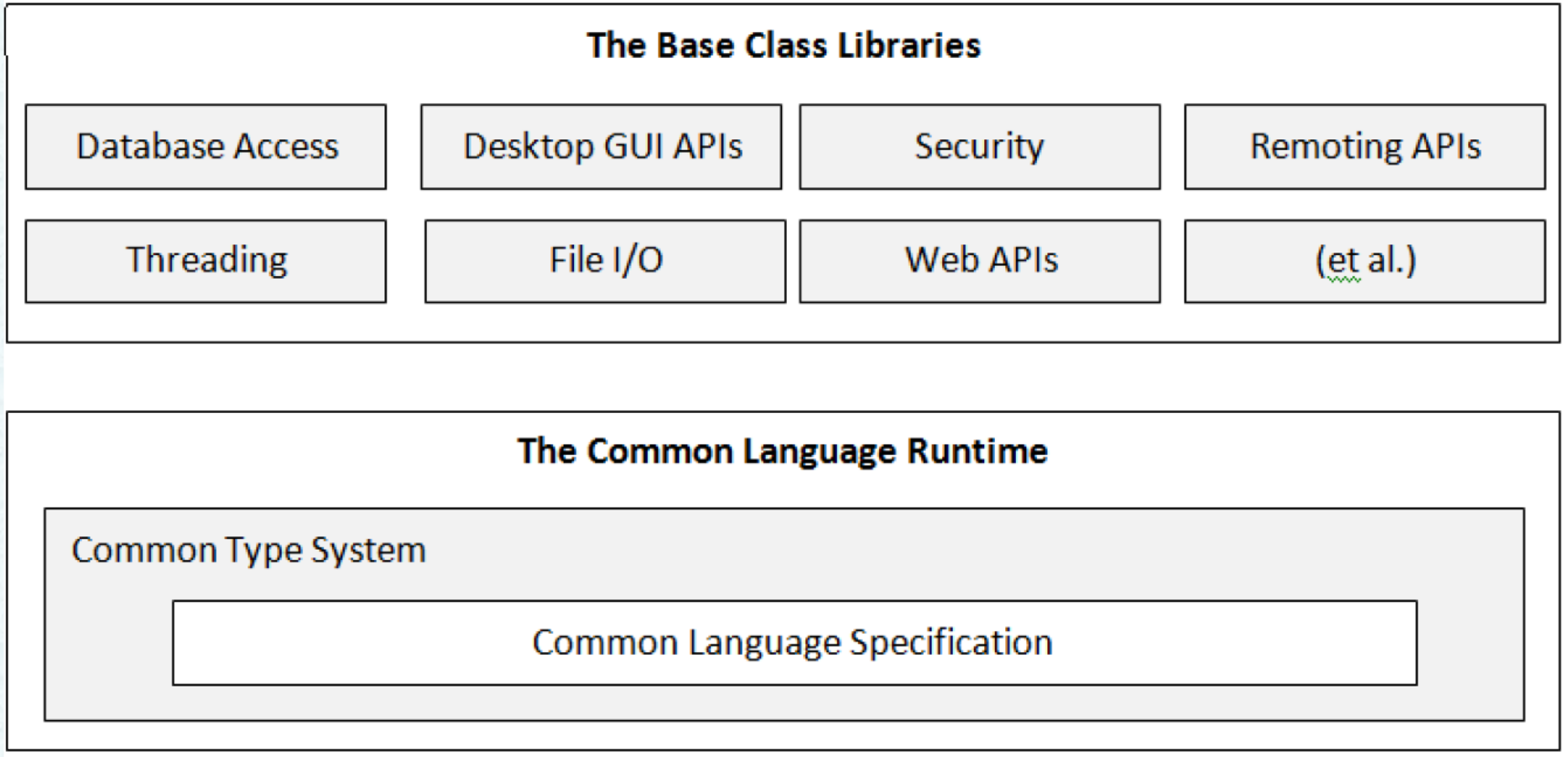
    The class library for thousands of prebuilt functionality (e.g. ADO.NET, Windows Forms, etc.) – 13000 classes
  - ASP.NET: engine that hosts web applications and includes a set of web specific services
  - Visual Studio.NET

.NET Framework

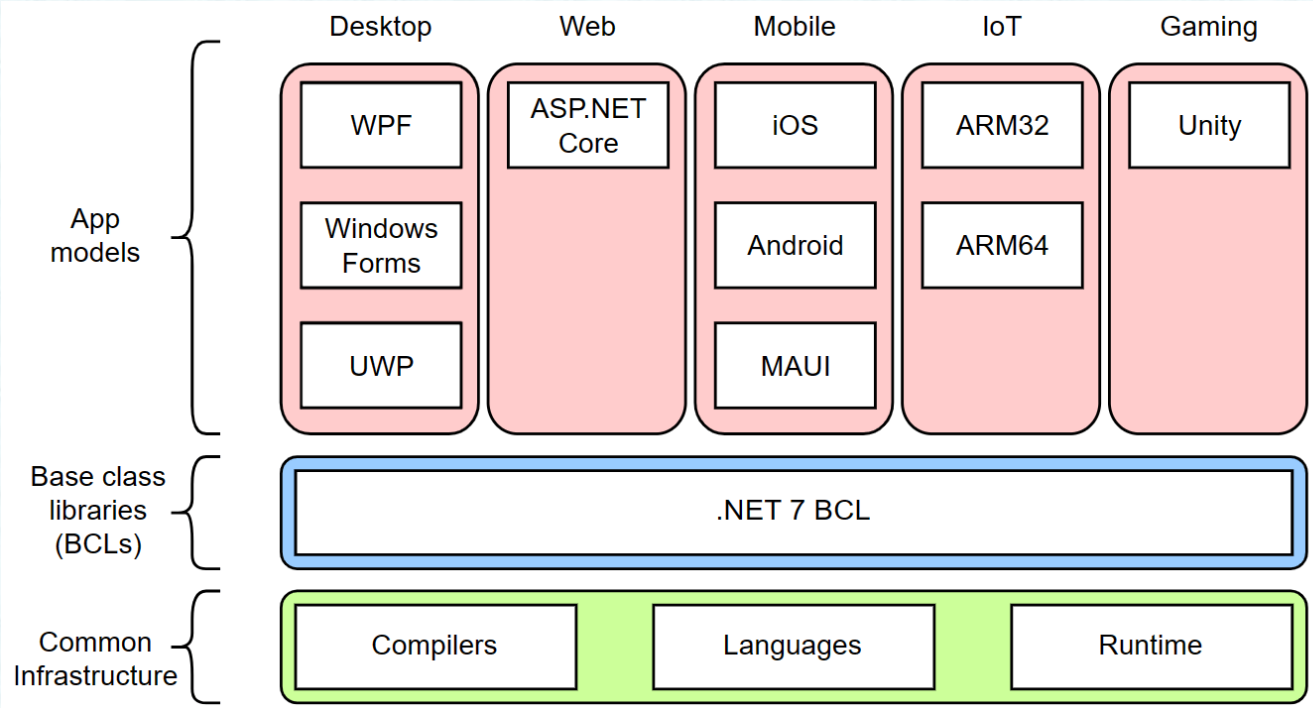
- The .NET Framework Class Library provides many capabilities that you can use to build substantial C# applications quickly and easily.
- It contains *thousands* of valuable *prebuilt* classes that have been tested and tuned to maximize performance.
- You should *re-use* the .NET Framework classes whenever possible to speed up the software-development process, while enhancing the quality and performance of the software you develop.

.NET Framework

- In addition to the CLR and CTS/CLS specifications, the .NET platform provides a base class library that is available to all .NET programming languages.

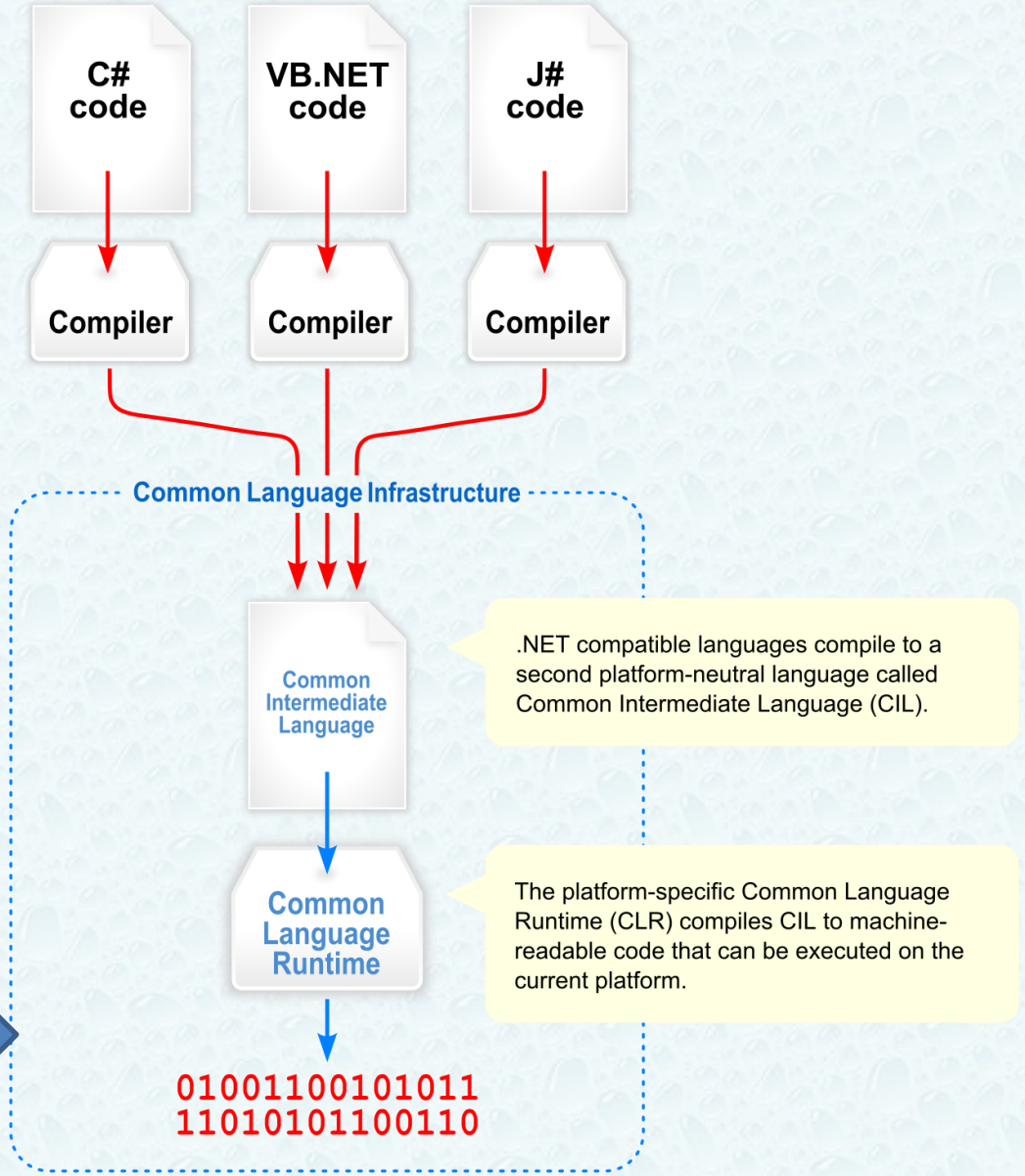


.NET Core Framework



Starting with .NET 5 and continuing in .NET 7 there is a single .NET platform which provides the BCL across all app models. All app models are based on the same underlying BCL.

Common Language Infrastructure (CLI)



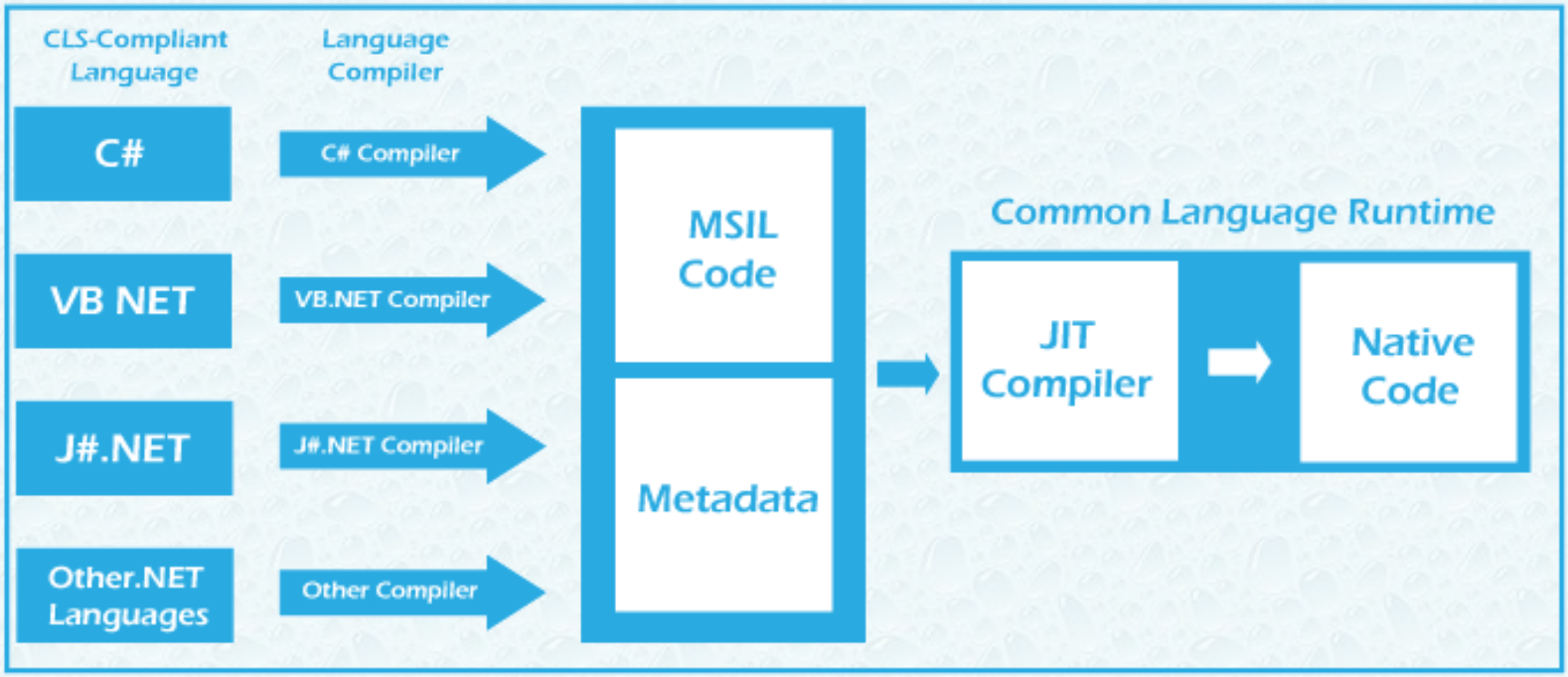
.Net:
Windows only

.Net Core:
any OS

Common Language Runtime

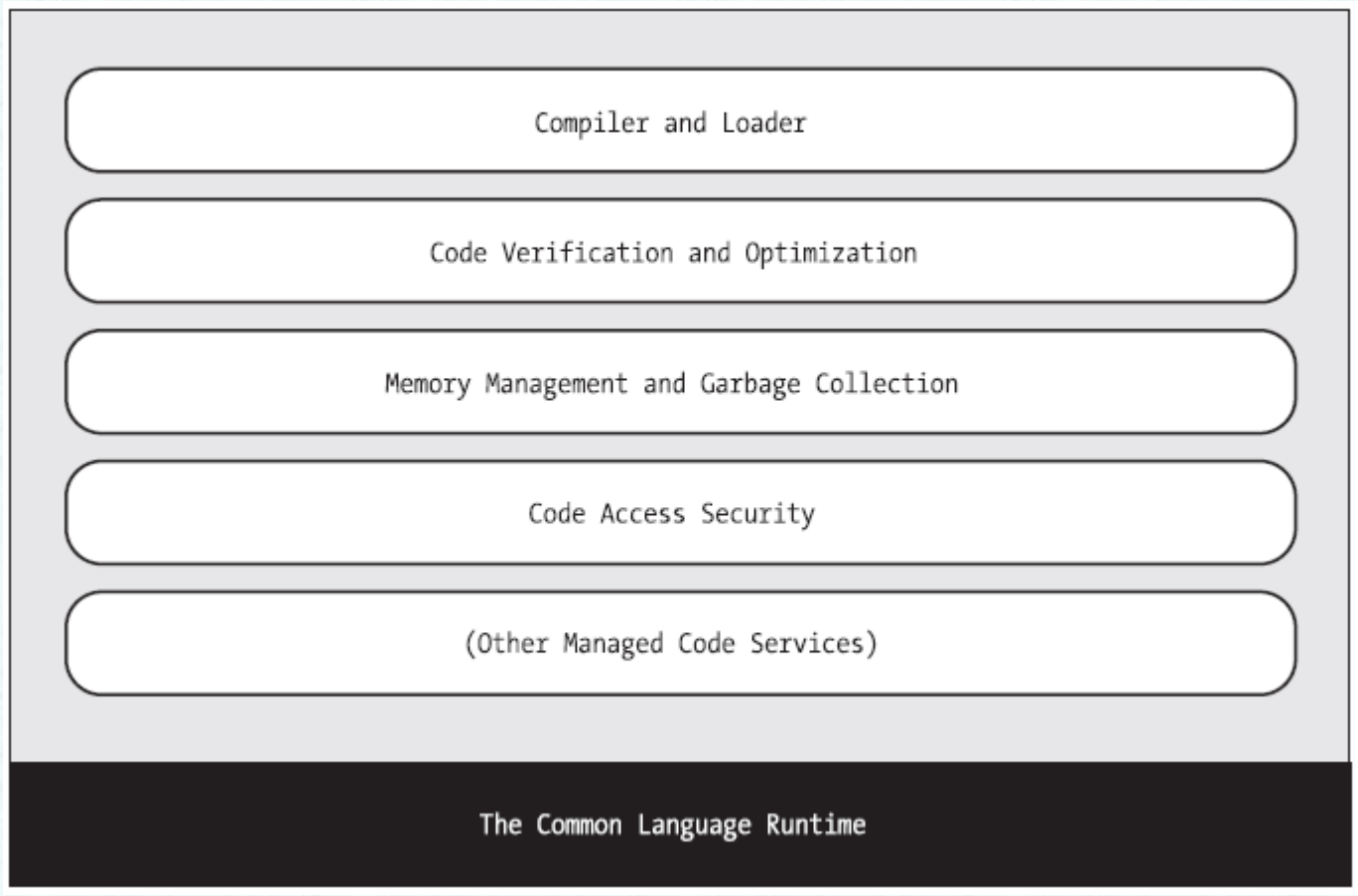
- The Common Language Runtime (CLR) executes .NET programs and provides functionality to make them easier to develop and debug.
- The CLR is a virtual machine (VM)—software that manages the execution of programs and hides from them the underlying operating system and hardware.
- The source code for programs that are executed and managed by the CLR is called managed code.

Common Language Infrastructure (CLI) - .Net Core Framework



Exceution of a .NET Application

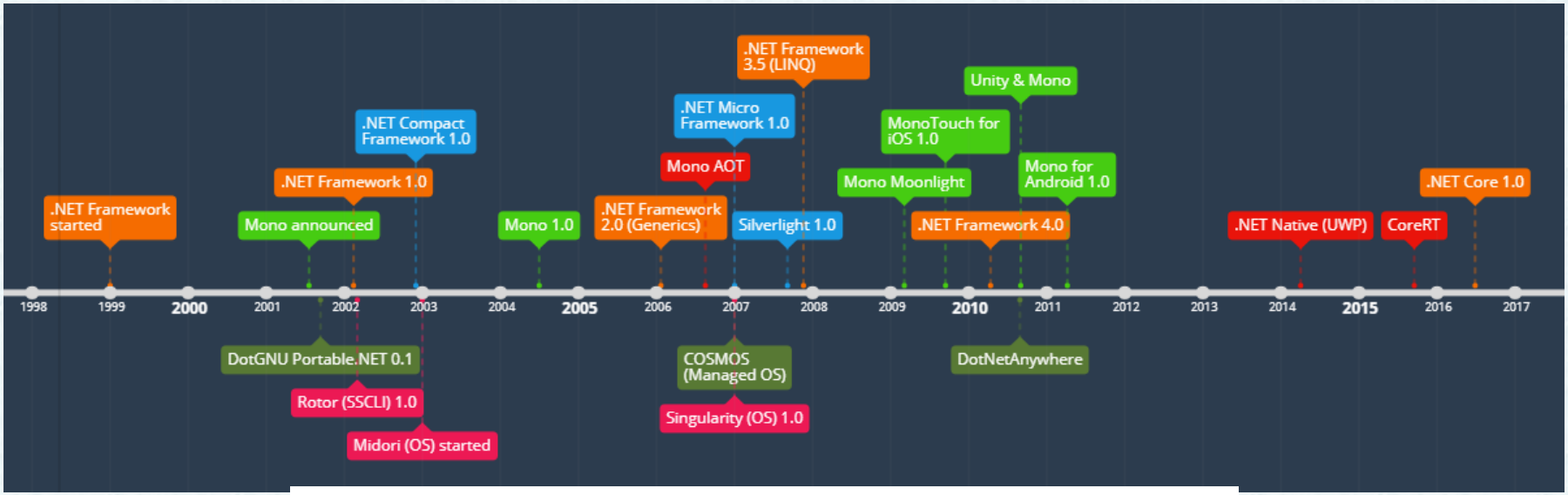
Common Language Runtime (CLR)



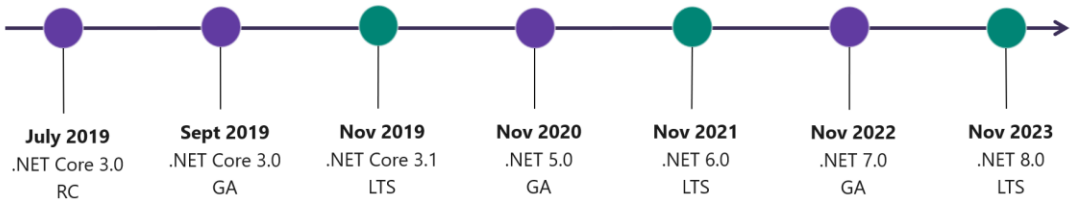
Intermediate Language (IL)

- All .NET languages are compiled into another low-level language before the code is executed
- This lower-level language is the Common Intermediate Language (CIL, or just IL)
- Because of this feature, C# programs can use VB.NET (or any other .NET language) components
- Every EXE or DLL file built with a .NET language contains IL code
- .NET Framework should be installed on the computer that the .NET program is deployed

History of .NET

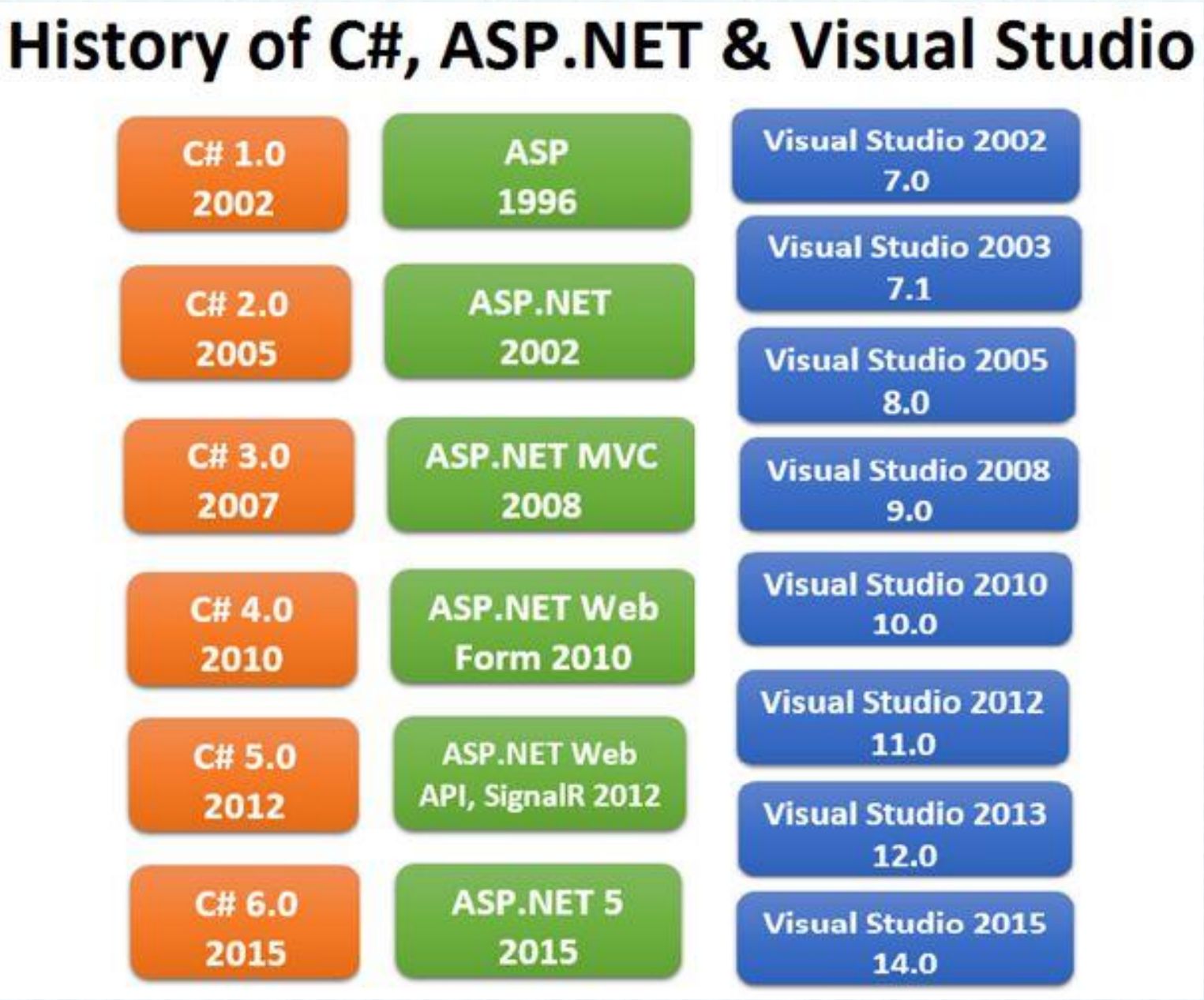


.NET Schedule



- .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

History of C#, ASP.NET & Visual Studio



Application Types

- Use to develop the following types of apps and services:
  - Console apps. (black screen)
  - Windows GUI apps (Windows Forms).
  - ASP.NET apps.
  - Windows services.
  - Windows Presentation Foundation (WPF) apps.
  - Service-oriented apps using Windows Communication Foundation (WCF).
  - Workflow-enabled apps using Windows Workflow Foundation (WF).

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

ASP.Net Core Frameworks

- ASP.NET offers four primary development frameworks
  - ASP.NET Web Forms
    - Declarative and control-based programming similar to Windows Forms.
    - Build a web application without having experience in HTML and JavaScript.
  - ASP.NET MVC
    - Use patterns and principles like test-driven development and separation of concerns.
    - Separating the business logic layer of a web application from its presentation layer.
  - ASP.NET Web Pages
    - In the Web Pages model, you create HTML pages and then add server-based code to the page in order to dynamically control how that markup is rendered.
      - Using HTML controls for existing HTML pages
  - ASP.NET Single Page Application
    - ASP.NET Single Page Application (SPA) helps you build applications that include significant client-side interactions using HTML 5, CSS 3 and JavaScript.
    - New template for building single page applications using **ASP.NET Web API**.

.NET Languages

- The .NET Framework is a language-independent platform that allows interoperability among the supported programming languages.
- The framework supports these programming languages:
  - C#:
    - object-oriented programming language
    - Offers type-safety, scalability support, garbage collection, and other productivity-enhancing features.
    - Easy to use and can reduce application development time.
  - F#:
    - Open-source, cross-platform language with object-oriented and imperative programming capabilities.
    - Core functional programming language for .NET.
  - Visual Basic:
    - build object-oriented apps.
    - Type-safety and uses simple syntax.
  - Developers can also use managed C++, Python, COBOL, Ruby and many other languages found in the Visual Studio Languages to code in .NET.

.NET Languages

- C# is designed for .NET technology
- C# and VB.NET uses the same .NET library
- C# and VB.NET codes are very similar
- Both C# and VB.NET are object-oriented languages
- A C# programmer can easily understand and convert VB.NET programs

Main C# features

- .NET 1.0 (released 2001) - C# version 1.0
  - No pointers required.
  - Automatic memory management through garbage collection.
    - C# does not support a delete keyword.
  - Formal syntactic constructs for classes, interfaces, structures, enumerations, and delegates.
  - Overload operators for a custom type

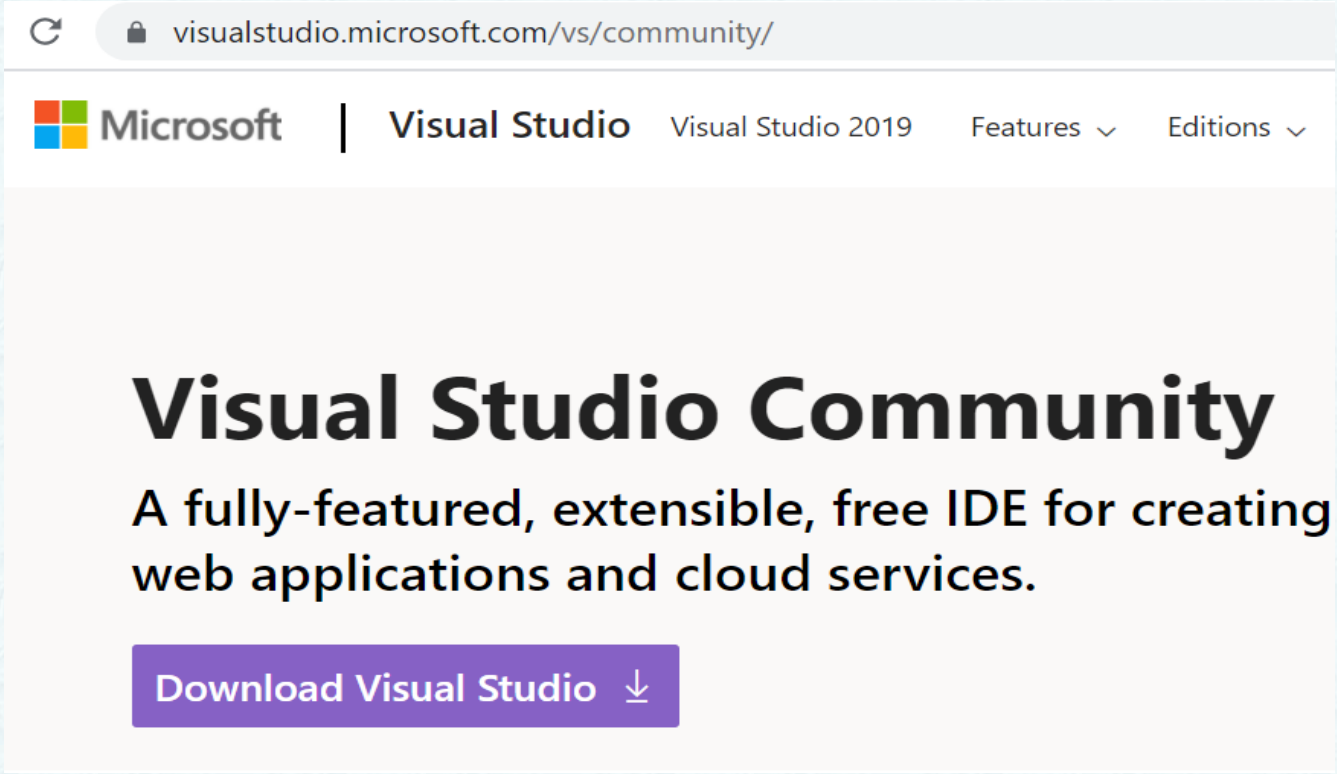
- .NET 2.0 (released 2005) - C# version 2.0
  - Generics - defer the specification of types until the class or method is declared
  - Anonymous methods - inline function anywhere a delegate type is required.
  - Partial keyword - single class across multiple code files.
  - Nullable value types – adding null values to value types
  - Iterators are used to step through collections such as lists and arrays
  - Covariance and contravariance - implicit reference conversion for array types and delegate types.

Main C# features

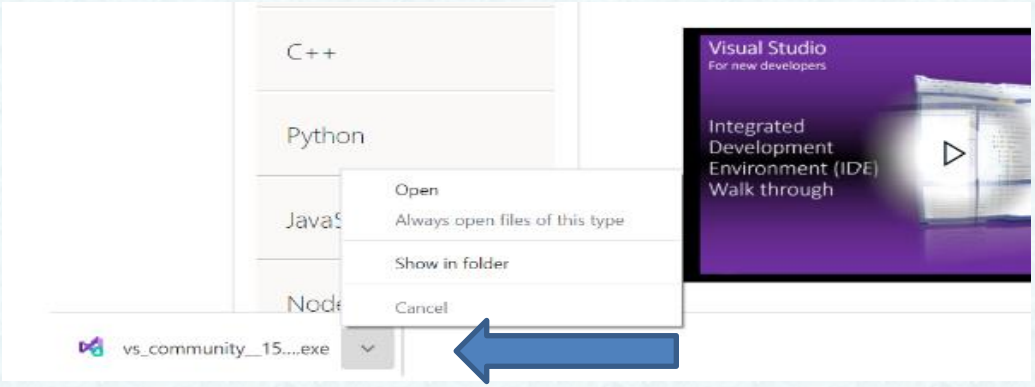
- .NET 3.5 (released 2008) - C# version 3.0
  - Auto-implemented properties - no additional logic is required in the property accessors.
  - Strongly typed queries (e.g., LINQ) - set of instructions that describes what data to retrieve
  - Anonymous types read-only properties into a single object without having to define a type
  - Extension methods - extend the functionality of an existing type (without subclassing)
  - Lambda operator (=>) further simplifies working with .NET delegate types.
- .NET 4.0 (released 2010) - C# version 4.0
  - Optional method parameters, as well as named method arguments.
  - dynamic binding - dynamic lookup of members at runtime.
- .NET 4.5. (Released 2012) - C# version 5.0
  - async and await, which massively simplify multithreaded and asynchronous programming.
- .NET 4.6 (Released 2015) - C# version 6.0
  - Release many smaller features
    - Static imports, Exception filters, Auto-property initializers, Expression bodied members, Null propagator, String interpolation, nameof operator, Index initializers.
- .NET Core (Released 2017) - C# version 7.0 – Cross Platforms (working in different OS)
  - Azure Support, AI Support
  - Improved performance and productivity
    - Out variables, Tuples and deconstruction, Pattern matching, Local functions, Expanded expression bodied members, Ref locals and returns

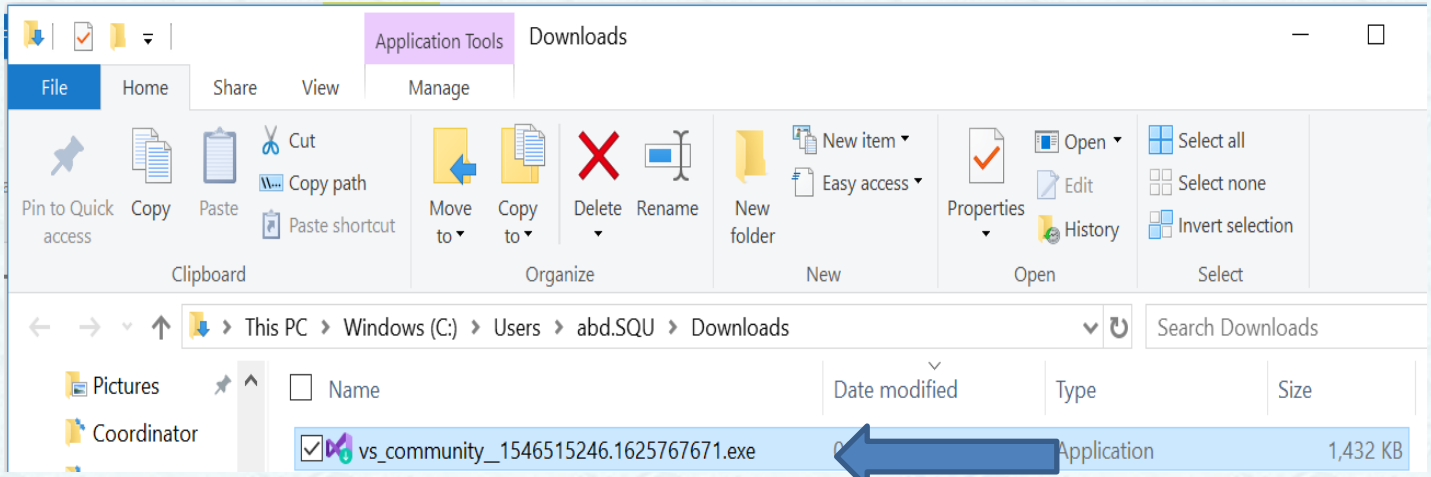
Installing Visual Studio 2019

- **Website: https://visualstudio.microsoft.com/vs/community**

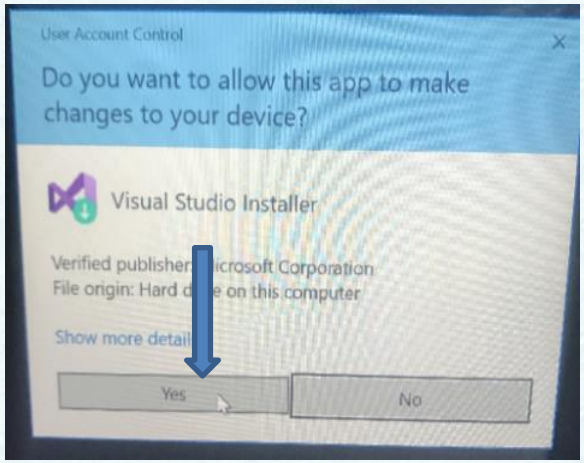


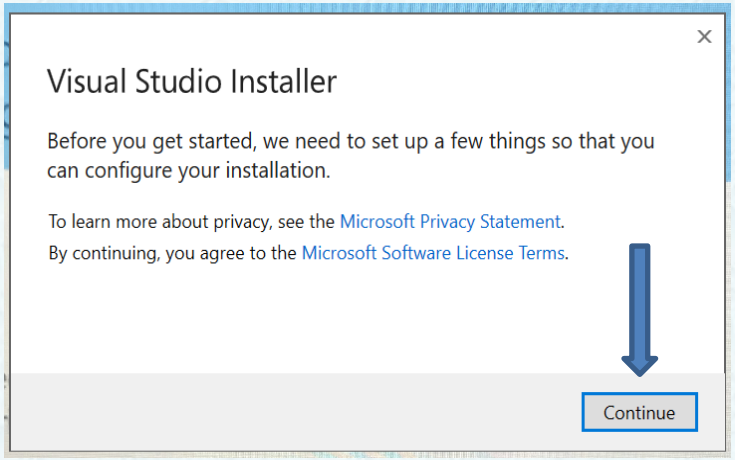
Installing Visual Studio 2019



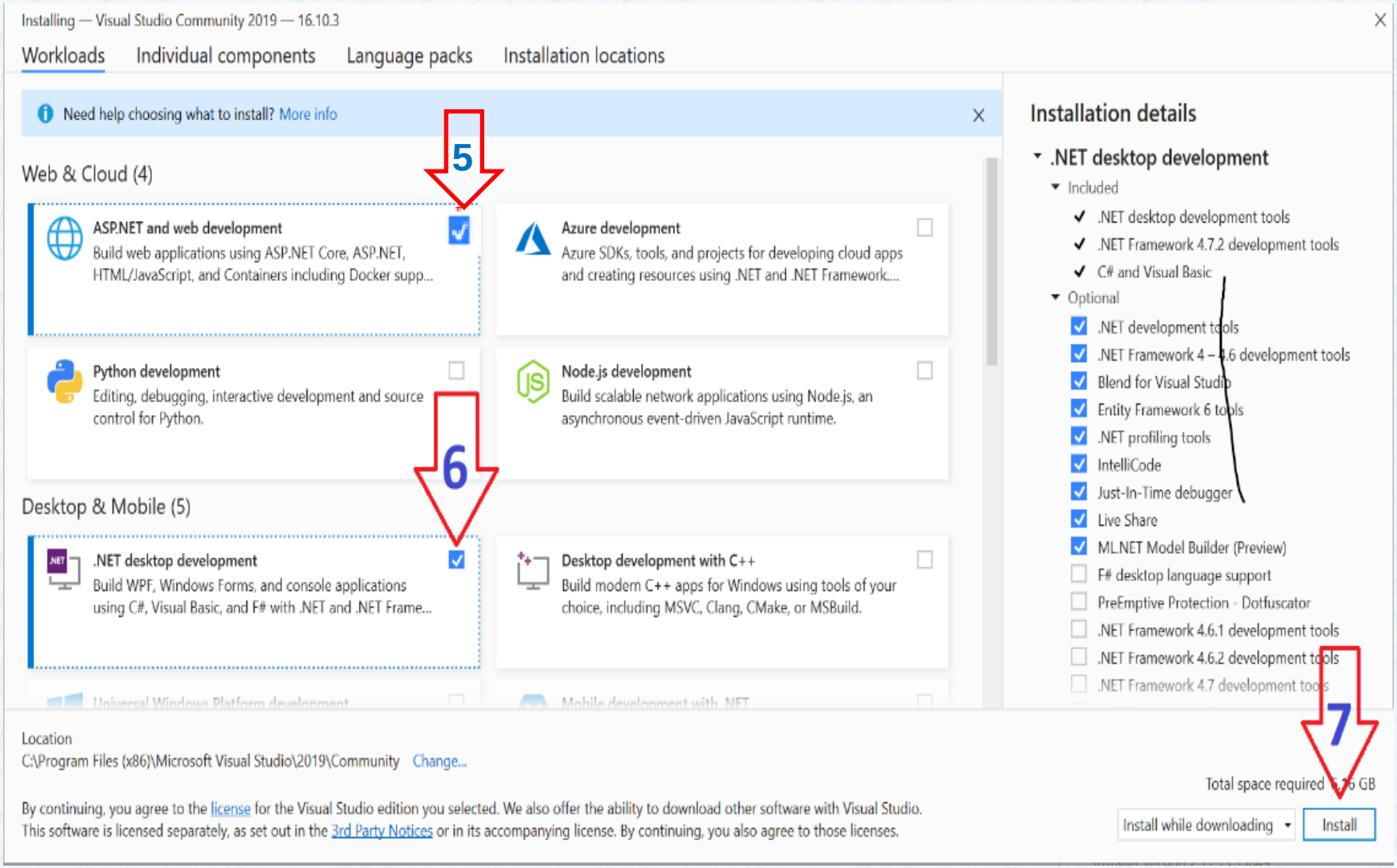


Installing Visual Studio 2019

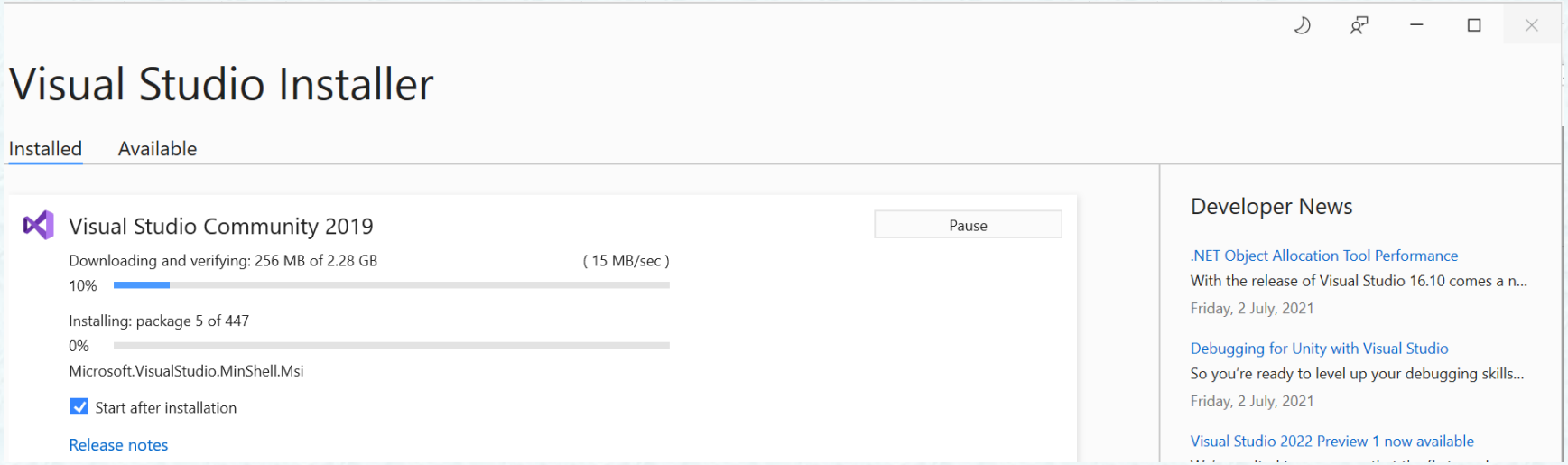


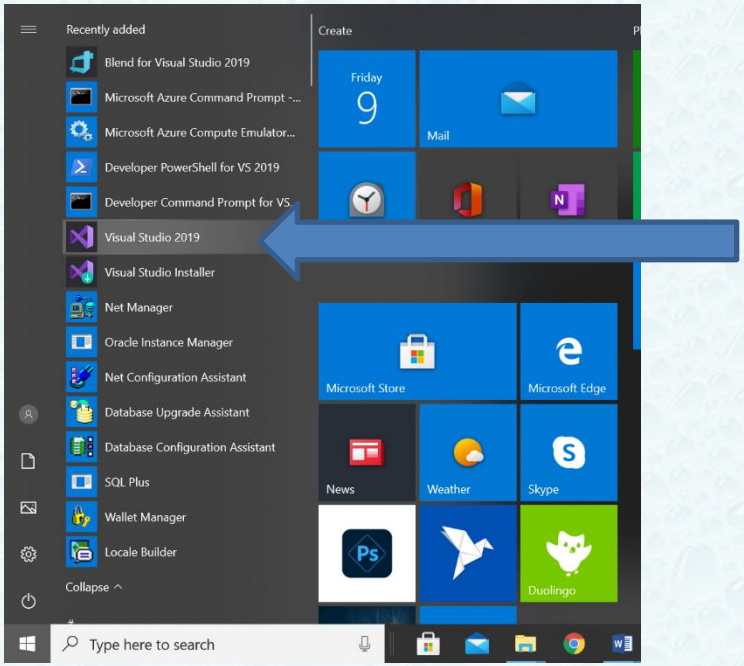


Installing Visual Studio 2019



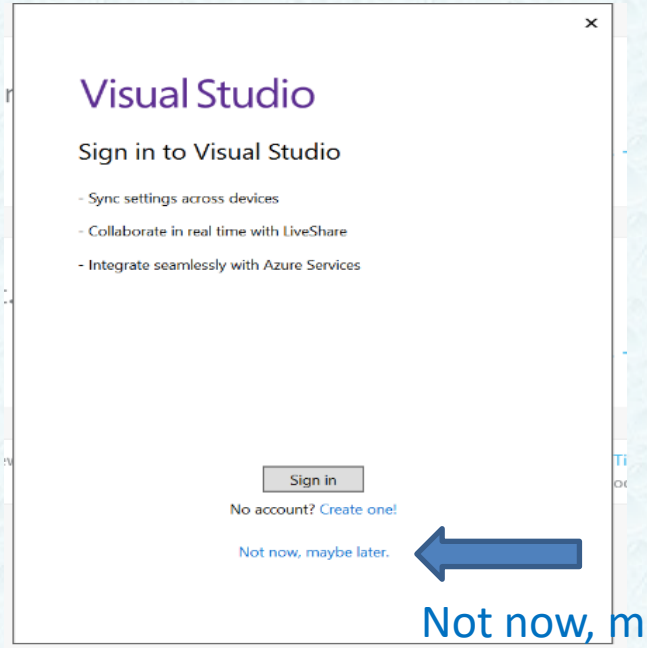
Installing Visual Studio 2019



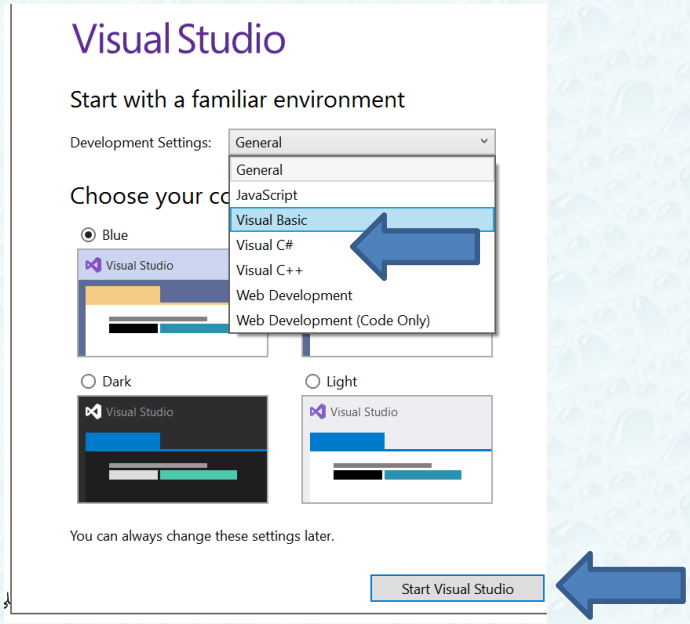


Launching Visual Studio

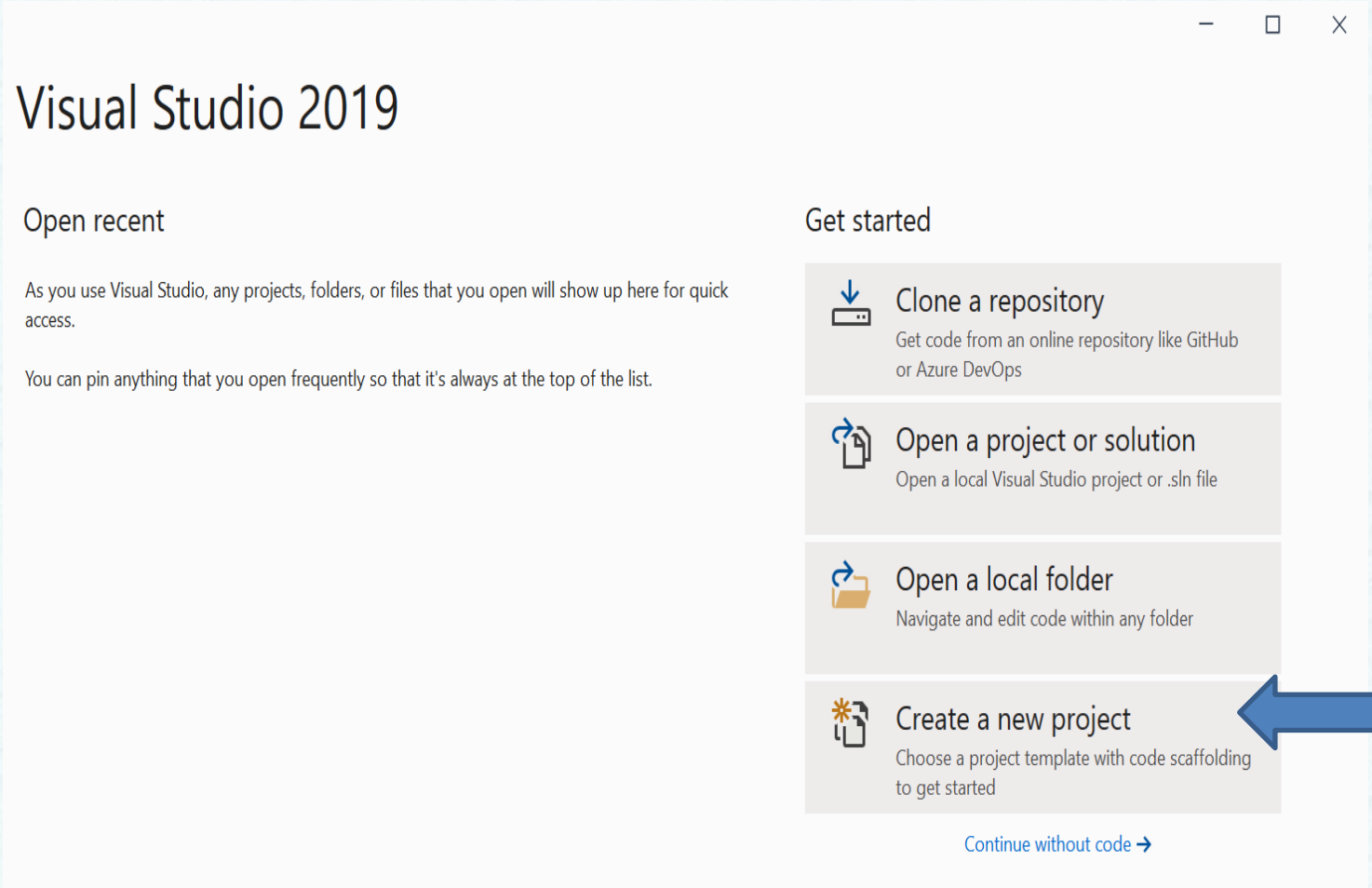
Opening Visual Studio 2019 for the first time



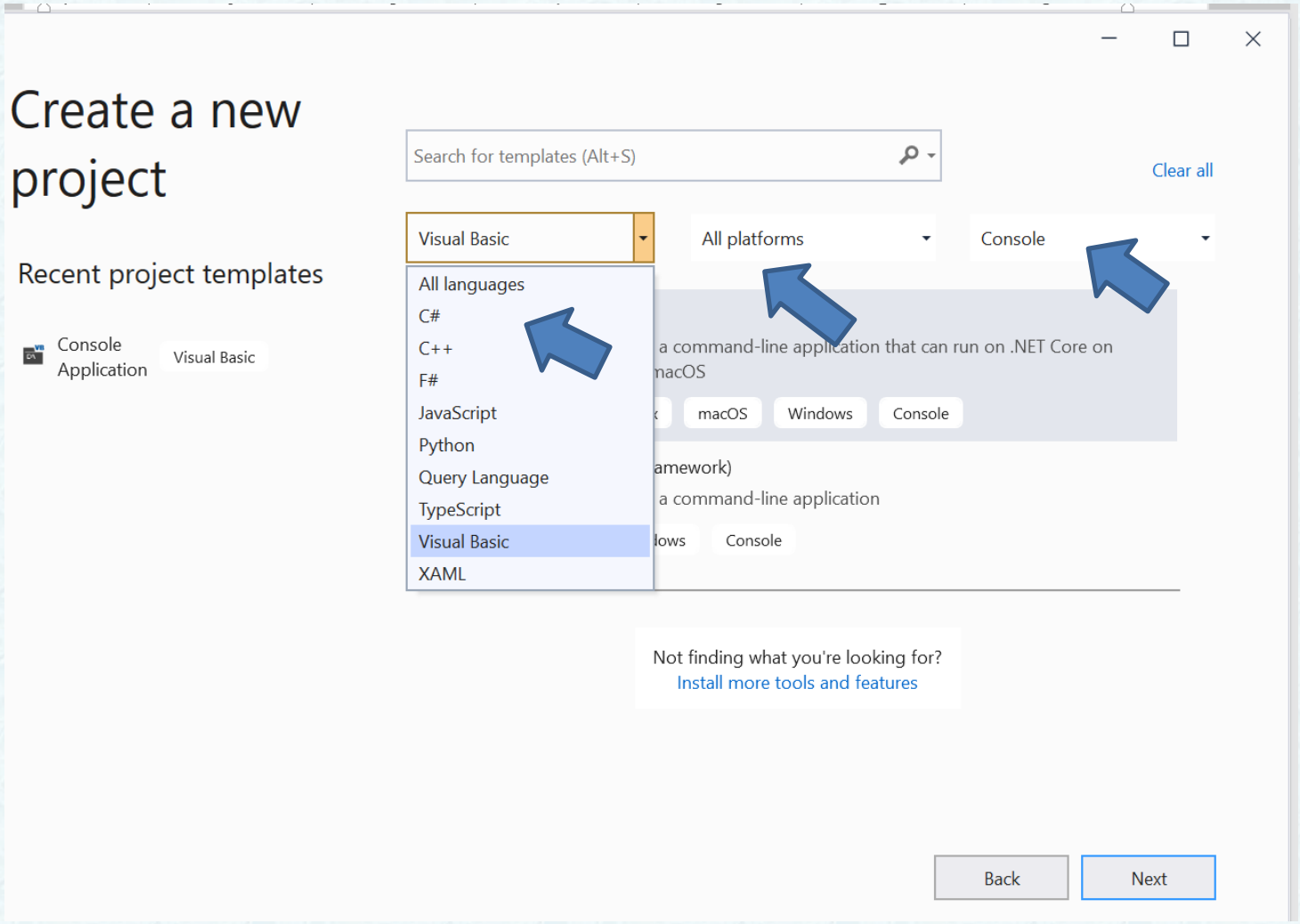
Not now, maybe later



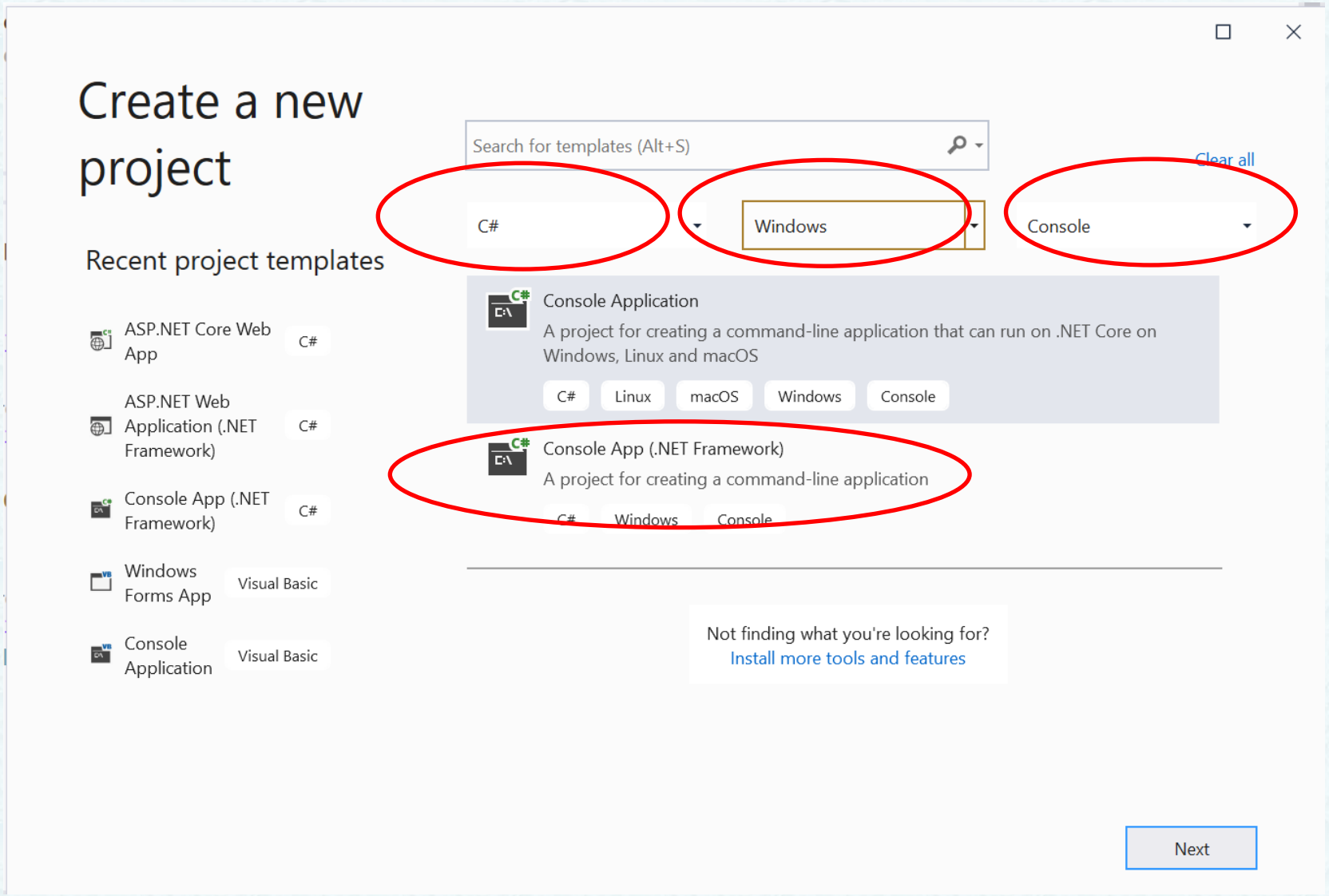
Create a New Project



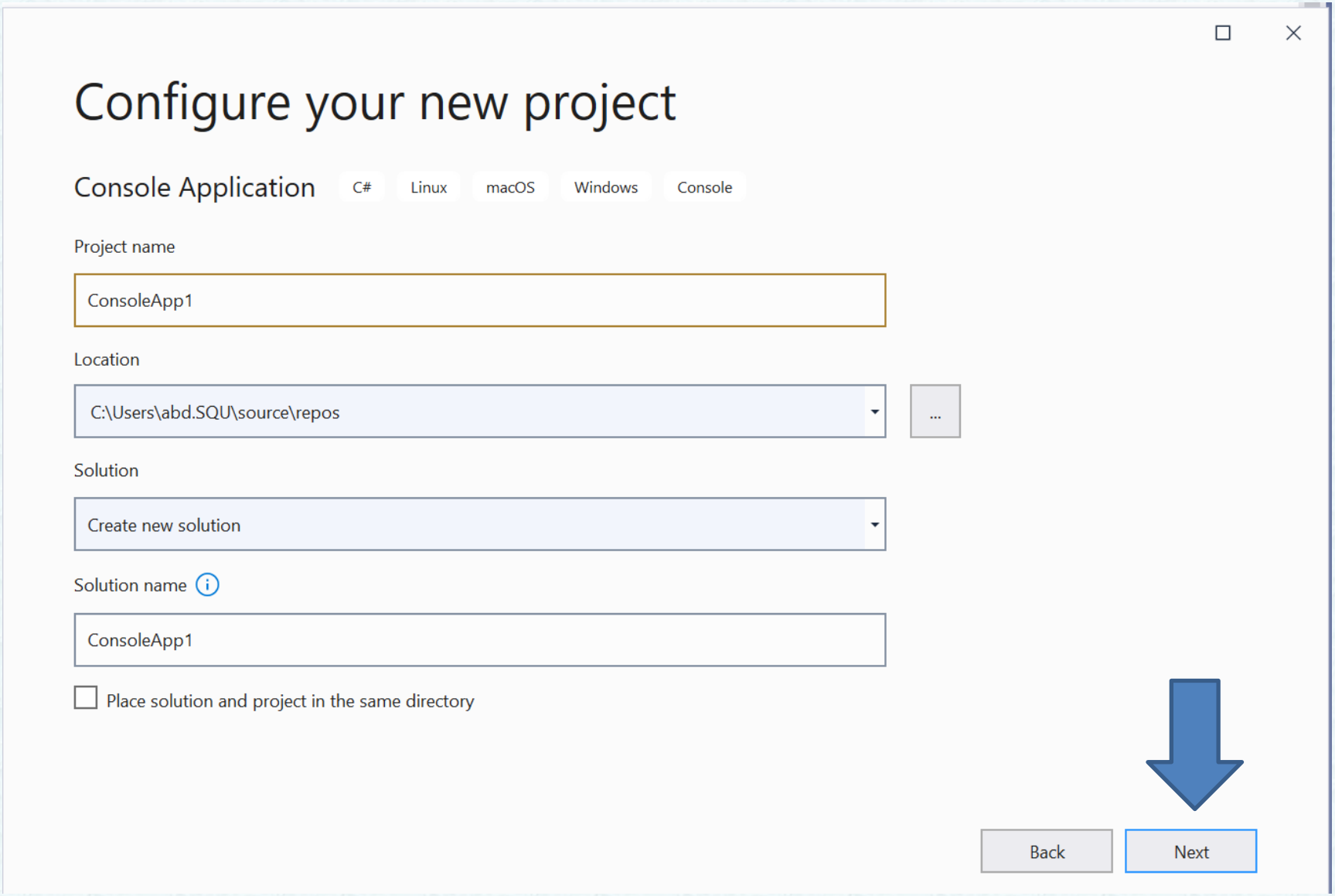
Choosing the application Type



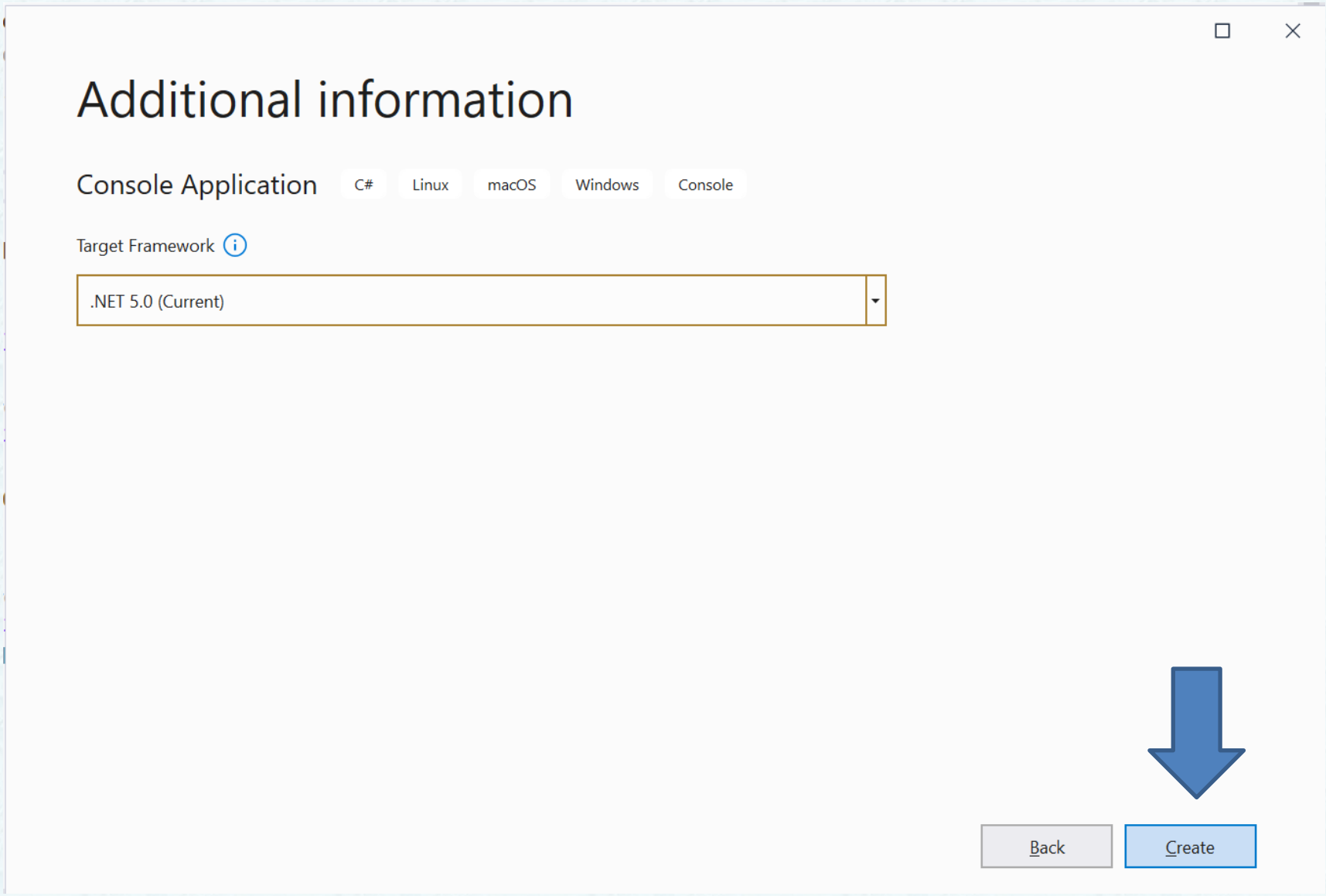
Creating C# Application



Creating C# Application



Creating C# Application



FIRST C# Application