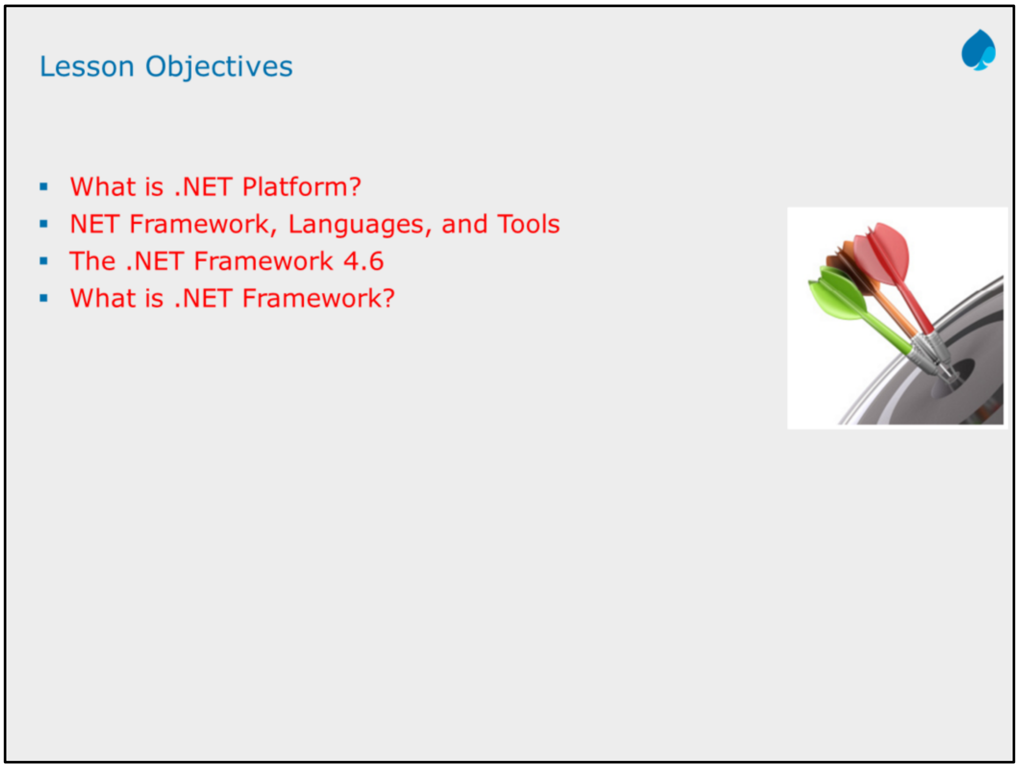
C# 7.0

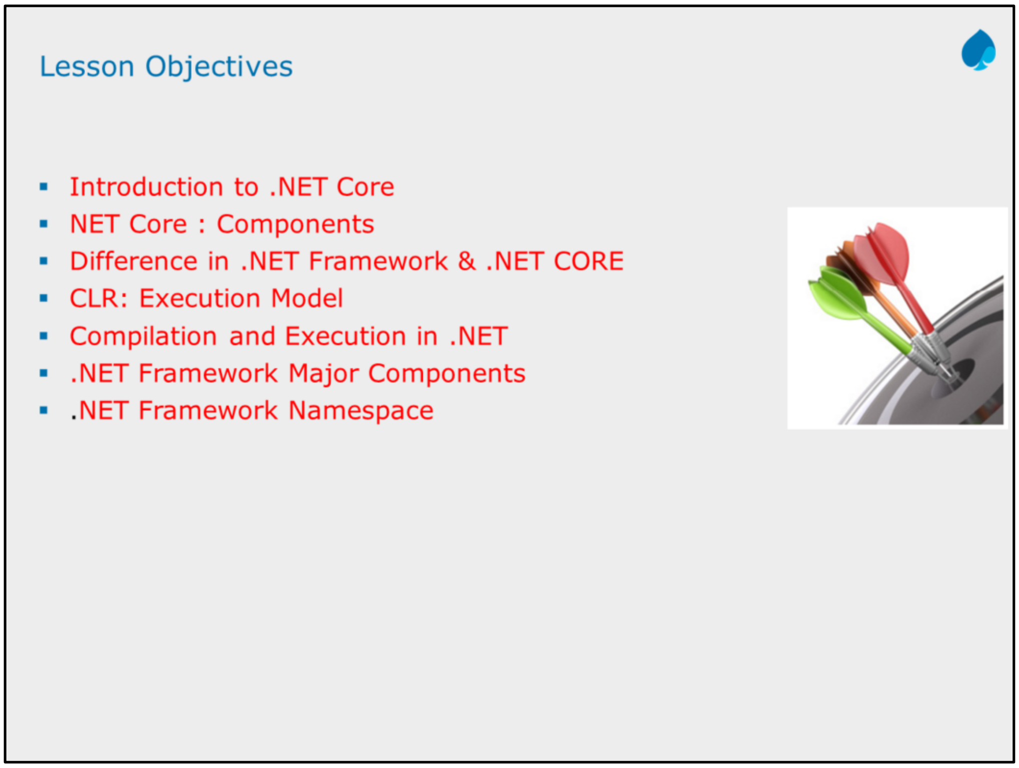
Lesson 01 : Introduction to .NET Framework 4.6

Capgemini



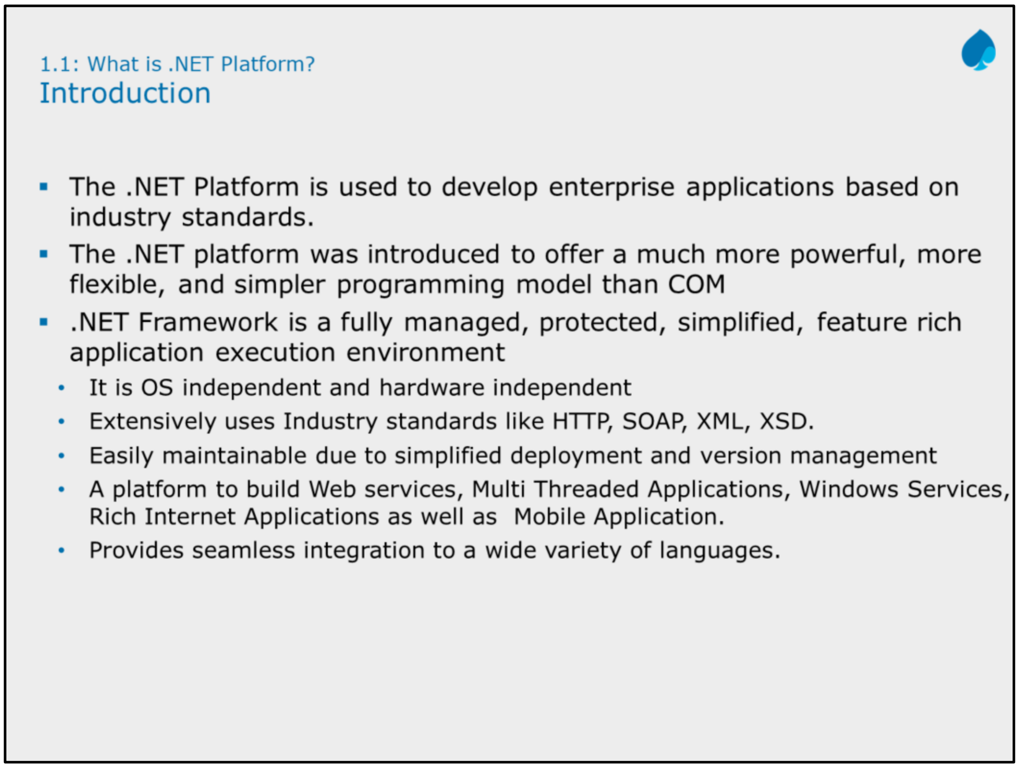
Lesson Objectives

- What is .NET Platform?
- NET Framework, Languages, and Tools
- The .NET Framework 4.6
- What is .NET Framework?



Lesson Objectives

- Introduction to .NET Core
- NET Core : Components
- Difference in .NET Framework & .NET CORE
- CLR: Execution Model
- Compilation and Execution in .NET
- .NET Framework Major Components
- .NET Framework Namespace

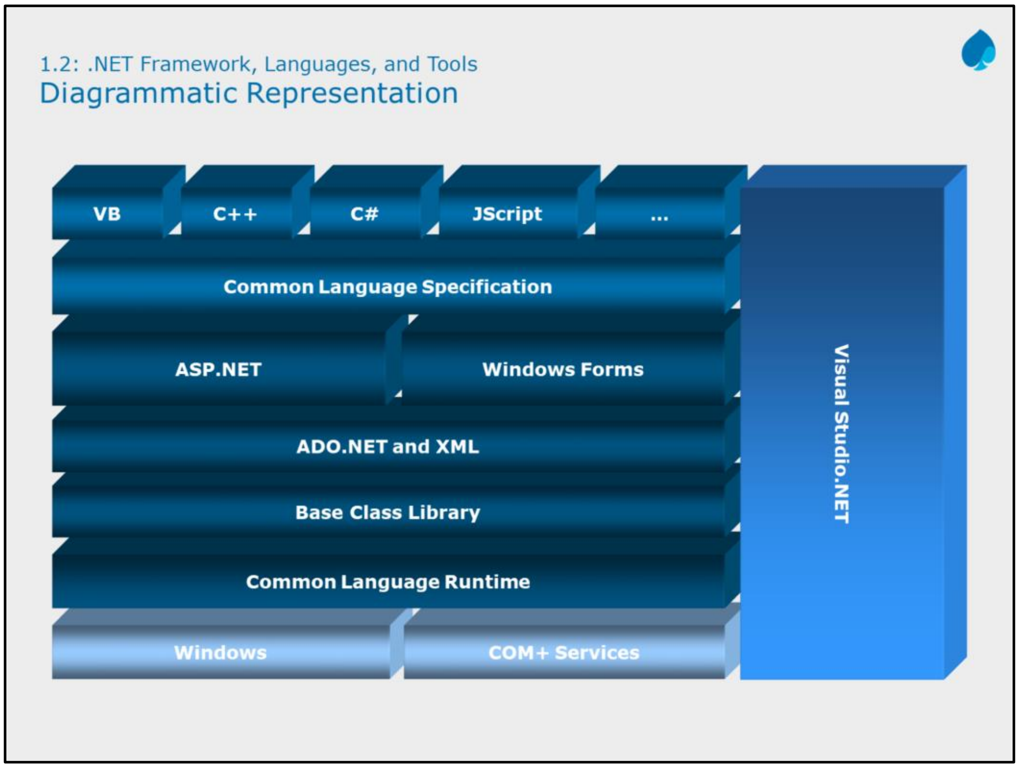

1.1: What is .NET Platform?

Introduction

- The .NET Platform is used to develop enterprise applications based on industry standards.
- The .NET platform was introduced to offer a much more powerful, more flexible, and simpler programming model than COM
- .NET Framework is a fully managed, protected, simplified, feature rich application execution environment
  - It is OS independent and hardware independent
  - Extensively uses Industry standards like HTTP, SOAP, XML, XSD.
  - Easily maintainable due to simplified deployment and version management
  - A platform to build Web services, Multi Threaded Applications, Windows Services, Rich Internet Applications as well as Mobile Application.
  - Provides seamless integration to a wide variety of languages.

<u>What is .NET Platform</u>?

According to the Microsoft, .NET Framework is: A platform for building, deploying, and running web services and applications. The .NET Framework provides a highly productive, standards-based, multi-language environment for integrating existing investments with next-generation applications and services as well as the ability to solve the challenges of deployment and operation of Internet-scale applications.



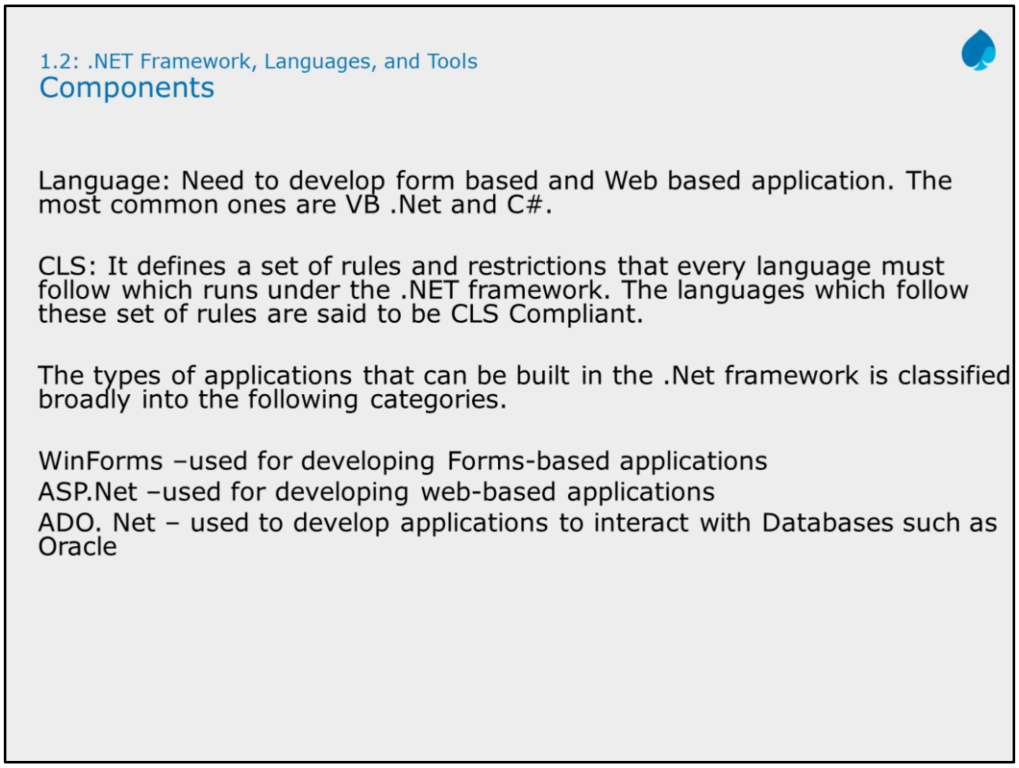
1.2: .NET Framework, Languages, and Tools

Diagrammatic Representation

.NET Framework, Languages, and Tools:

The common language runtime provides a code-execution environment that manages code targeting the .NET Framework. Code management can take the form of memory management, thread management, security management, code verification and compilation, and other system services.

Managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.



1.2: .NET Framework, Languages, and Tools

Components

Language: Need to develop form based and Web based application. The most common ones are VB .Net and C#.

CLS: It defines a set of rules and restrictions that every language must follow which runs under the .NET framework. The languages which follow these set of rules are said to be CLS Compliant.

The types of applications that can be built in the .Net framework is classified broadly into the following categories.

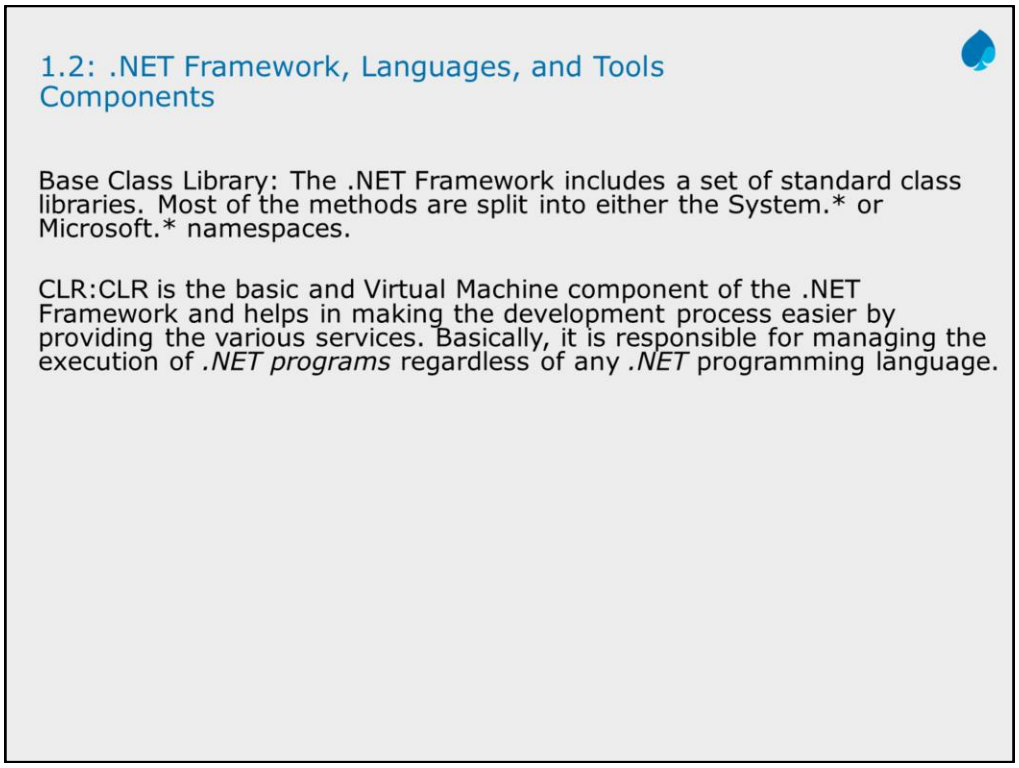
WinForms –used for developing Forms-based applications
ASP.Net –used for developing web-based applications
ADO. Net – used to develop applications to interact with Databases such as Oracle

Language:The .Net framework can be used to create both - **Form-based** and **Web-based** applications. Web services can also be developed using the .Net framework. The framework also supports various programming languages such as Visual Basic and C#. So developers can choose and select the language to develop the required application.

WinForms – This is used for developing Forms-based applications, which would run on an end user machine. Notepad is an example of a client-based application.
ASP.Net – This is used for developing web-based applications, which are made to run on any browser such as Internet Explorer, Chrome or Firefox.

The Web application would be processed on a server, which would have Internet Information Services Installed.
Internet Information Services or IIS is a Microsoft component which is used to execute an Asp.Net application.
The result of the execution is then sent to the client machines, and the output is shown in the browser.

ADO.Net – This technology is used to develop applications to interact with Databases such as Oracle or Microsoft SQL Server.
Microsoft always ensures that .Net frameworks are in compliance with all the supported Windows operating systems.

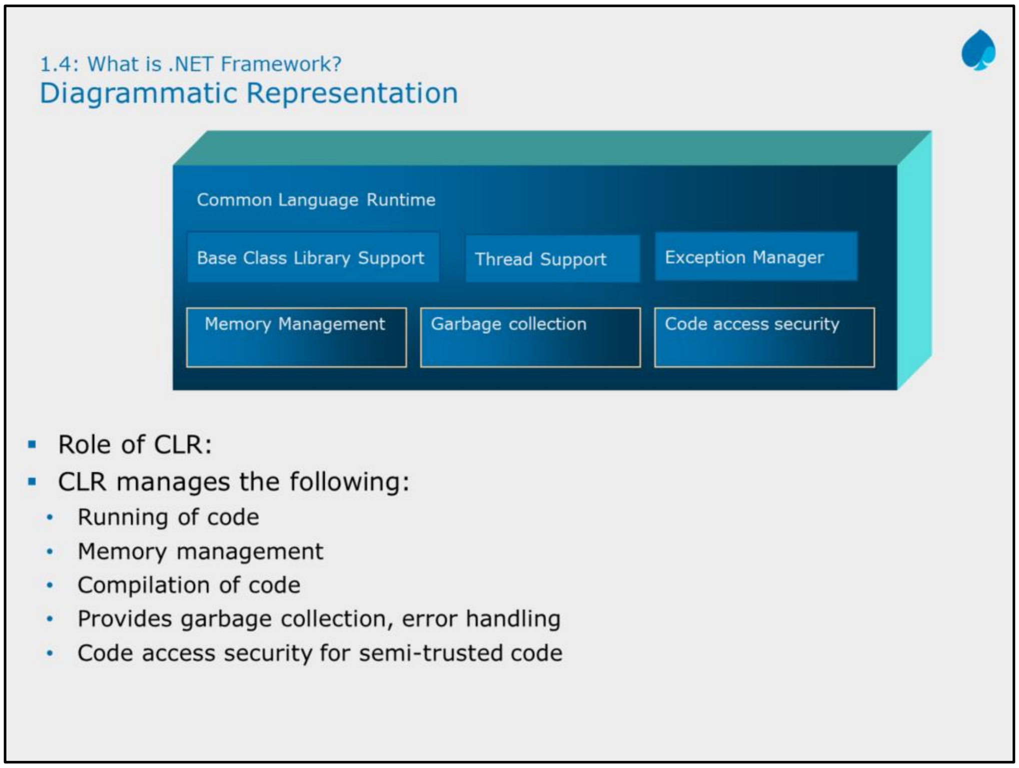


1.2: .NET Framework, Languages, and Tools Components

Base Class Library: The .NET Framework includes a set of standard class libraries. Most of the methods are split into either the System.* or Microsoft.* namespaces.

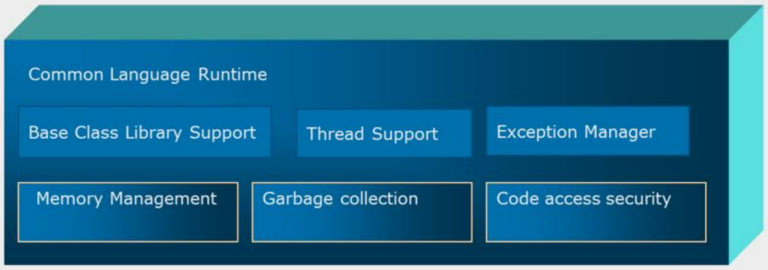
CLR:CLR is the basic and Virtual Machine component of the .NET Framework and helps in making the development process easier by providing the various services. Basically, it is responsible for managing the execution of *.NET programs* regardless of any *.NET* programming language.

CLR is the basic and Virtual Machine component of the **.NET Framework**. It is the **run-time enviornment in the .NET Framework** that runs the codes and helps in making the development process easier by providing the various services. Basically, it is responsible for managing the execution of *.NET programs* regardless of any *.NET* programming language. Internally, CLR implements the *VES(Virtual Execution System)* which is defined in the Microsoft's implementation of the *CLI(Common Language Infrastructure).*
The code that runs under the Common Language Runtime is termed as the Managed Code. In other words, you can say that CLR provides a managed execution enviornment for the *.NET* programs by improving the security, including the cross language integration and a rich set of class libraries etc. CLR is present in every .NET framework verison. Below table illustrate the CLR version in .NET framework.



1.4: What is .NET Framework?

Diagrammatic Representation



- Role of CLR:
- CLR manages the following:
  - Running of code
  - Memory management
  - Compilation of code
  - Provides garbage collection, error handling
  - Code access security for semi-trusted code

.NET Framework Major Components (Cont.)

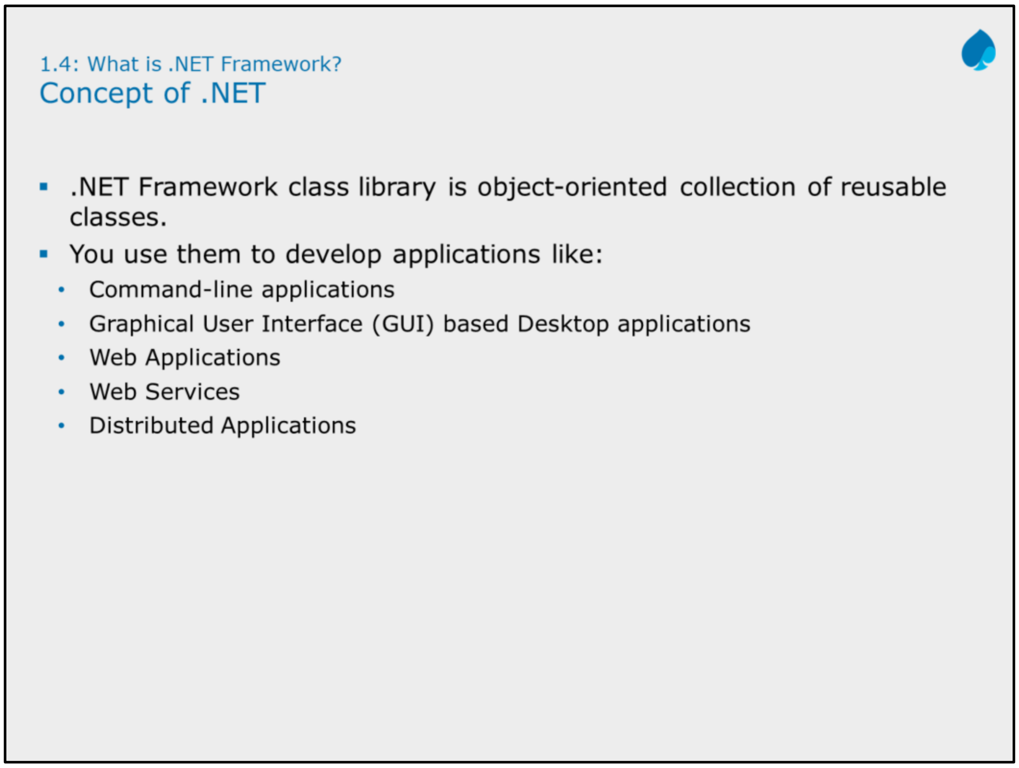
The Common Language Runtime in the .NET Framework is the Virtual Machine component that handles program execution for various languages such as C#, F#, Visual Basic .NET, etc. The managed execution environment is provided by giving various services such as memory management, security handling, exception handling, garbage collection, thread management, etc.

The Common Language Runtime implements the VES (Virtual Execution System) which is a run time system that provides a managed code execution environment. The VES is defined in Microsoft's implementation of the CLI (Common Language Infrastructure).

At the bottom is the CLR (Common Language Runtime). It is considered as the heart of .NET Framework. Following are the functionalities of CLR

1. Memory management.
2. **Base Class Library Support:**The Common Language Runtime provides support for the base class library. The BCL contains multiple libraries that provide various features such as *Collections*, *I/O*, *XML*, *DataType definitions*, etc. for the multiple *.NET* programming languages.
3. **Thread Support:** The CLR provides thread support for managing the parallel execution of multiple threads. The *System.Threading class* is used as the base class for this.

Exception Manager: The exception manager in the CLR handles the exceptions regardless of the *.NET Language* that created them. For a particular application, the catch block of the exceptions are executed in case they occur and if there is no catch block then the application is terminated.
Security Engine: The security engine in the CLR handles the security permissions at various levels such as the code level, folder level, and machine level. This is done using the various tools that are provided in the *.NET* framework.
Garbage Collector: Automatic memory management is made possible using the garbage collector in CLR. The garbage collector automatically releases the memory space after it is no longer required so that it can be reallocated.
Other components of CLR includes:
Debug Engine
JIT Compiler:
Code Manager:
CLR Loader:



1.4: What is .NET Framework?

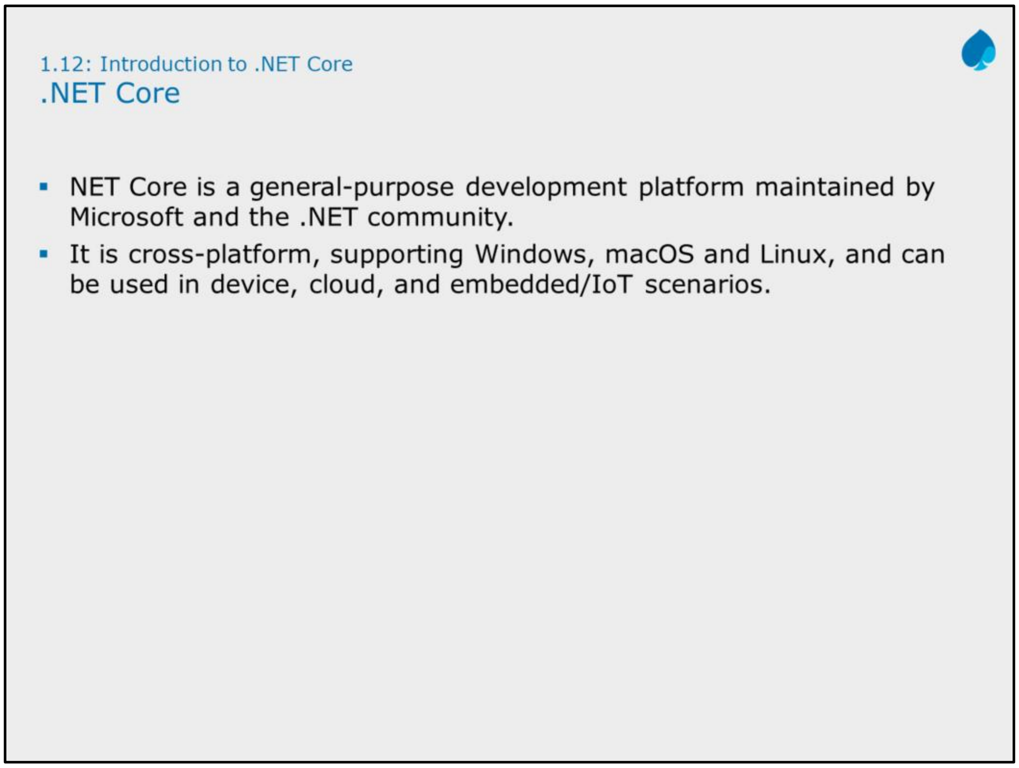
Concept of .NET

- .NET Framework class library is object-oriented collection of reusable classes.
- You use them to develop applications like:
  - Command-line applications
  - Graphical User Interface (GUI) based Desktop applications
  - Web Applications
  - Web Services
  - Distributed Applications

What is the .NET Framework?

The .NET Framework class library is a comprehensive, object-oriented collection of reusable classes that you can use to develop applications.

The applications can range from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET and Web Services.

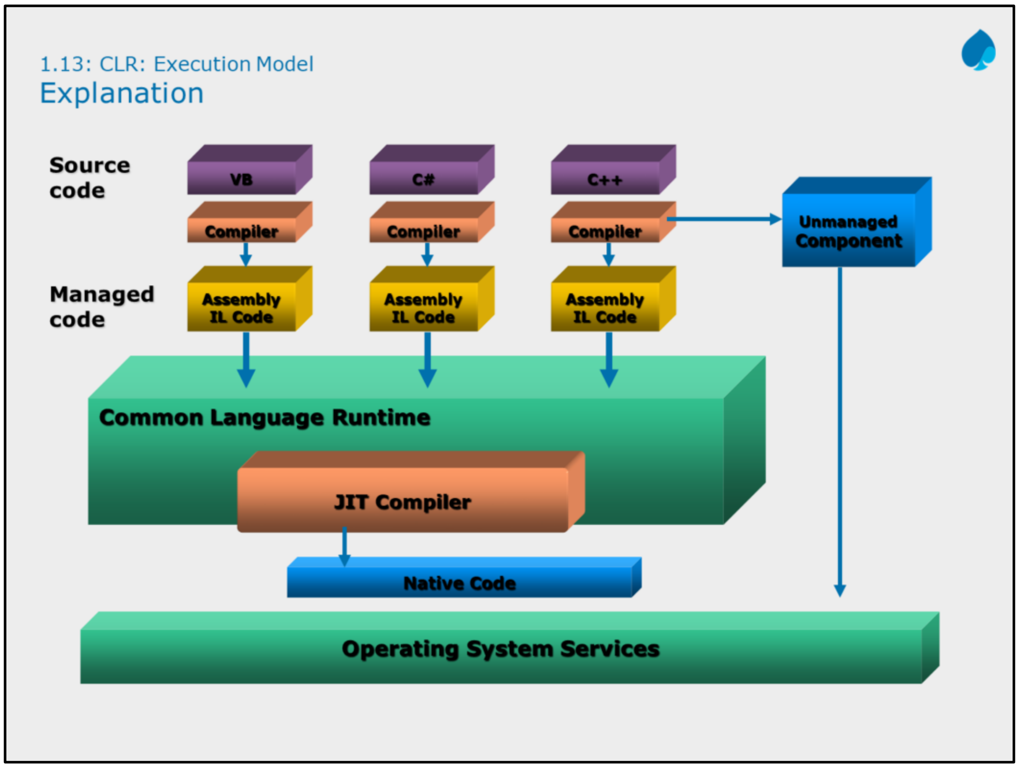


1.12: Introduction to .NET Core

.NET Core

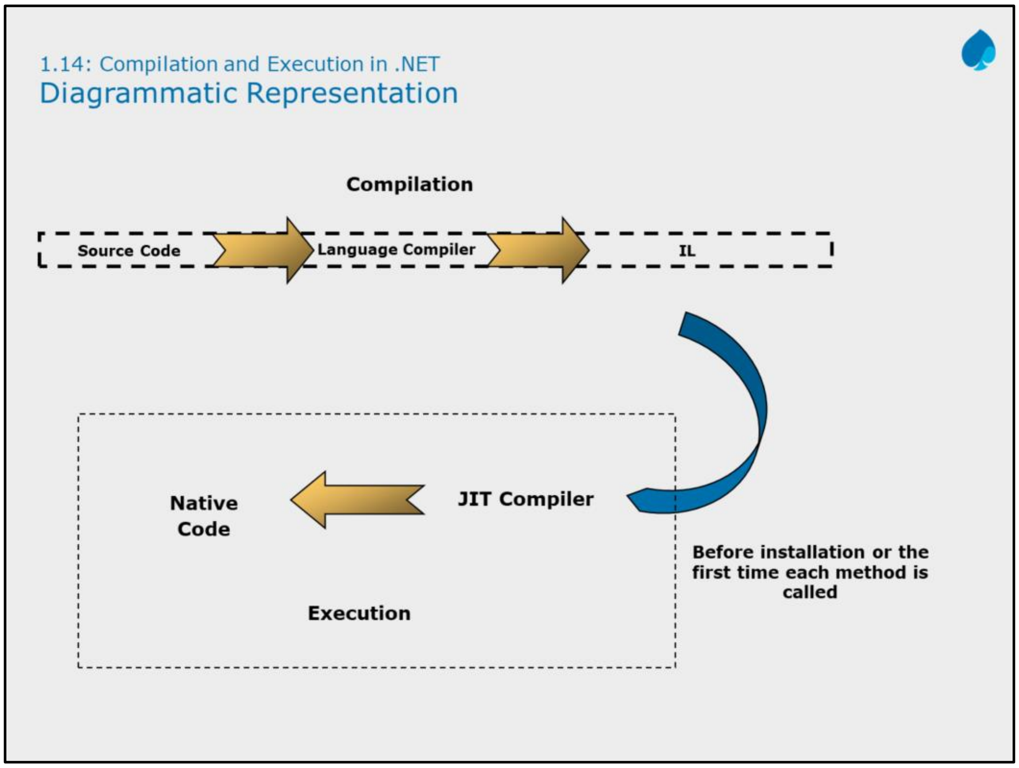
- NET Core is a general-purpose development platform maintained by Microsoft and the .NET community.
- It is cross-platform, supporting Windows, macOS and Linux, and can be used in device, cloud, and embedded/IoT scenarios.

Following are the Features :-
Flexible deployment: Can be included in your app or installed side-by-side user- or machine-wide.
Cross-platform: Runs on Windows, macOS and Linux; can be ported to other operating systems.
Command-line tools: All product scenarios can be exercised at the command-line.
Compatible: .NET Core is compatible with .NET Framework, Xamarin and Mono, via the .NET Standard
Open source: The .NET Core platform is open source, using MIT and Apache 2 licenses.



1.13: CLR: Execution Model

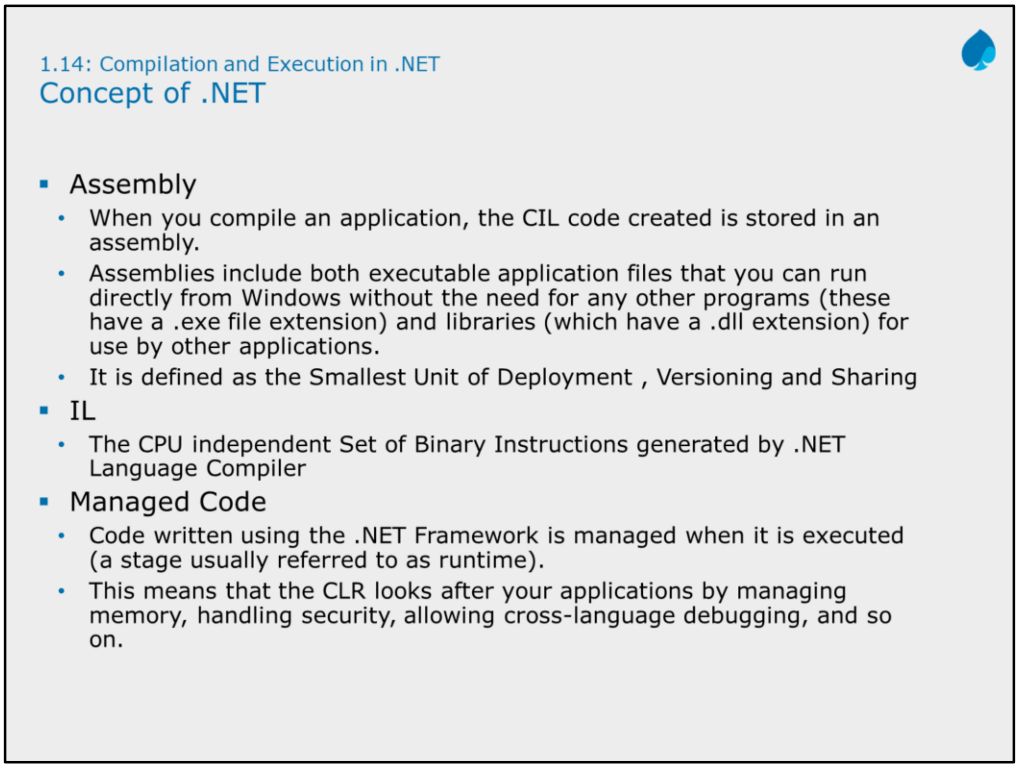
Explanation



1.14: Compilation and Execution in .NET

Diagrammatic Representation

Compilation and Execution in .NET:
The managed execution process includes the following steps:
Designing and writing your source code
To obtain the benefits provided by the common language runtime, you must use one or more language compilers that target the runtime.
Compiling your code to Intermediate language (IL)
Compiling translates your source code into IL and generates the required metadata.
When compiling to managed code, the compiler translates your source code into intermediate language (IL), which is a CPU-independent set of instructions that can be efficiently converted to native code. IL includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations. Before code can be executed, MSIL must be converted to CPU-specific code by a just in time (JIT) compiler. Since the runtime supplies one or more JIT compilers for each computer architecture it supports, the same set of MSIL can be JIT-compiled and executed on any supported architecture.

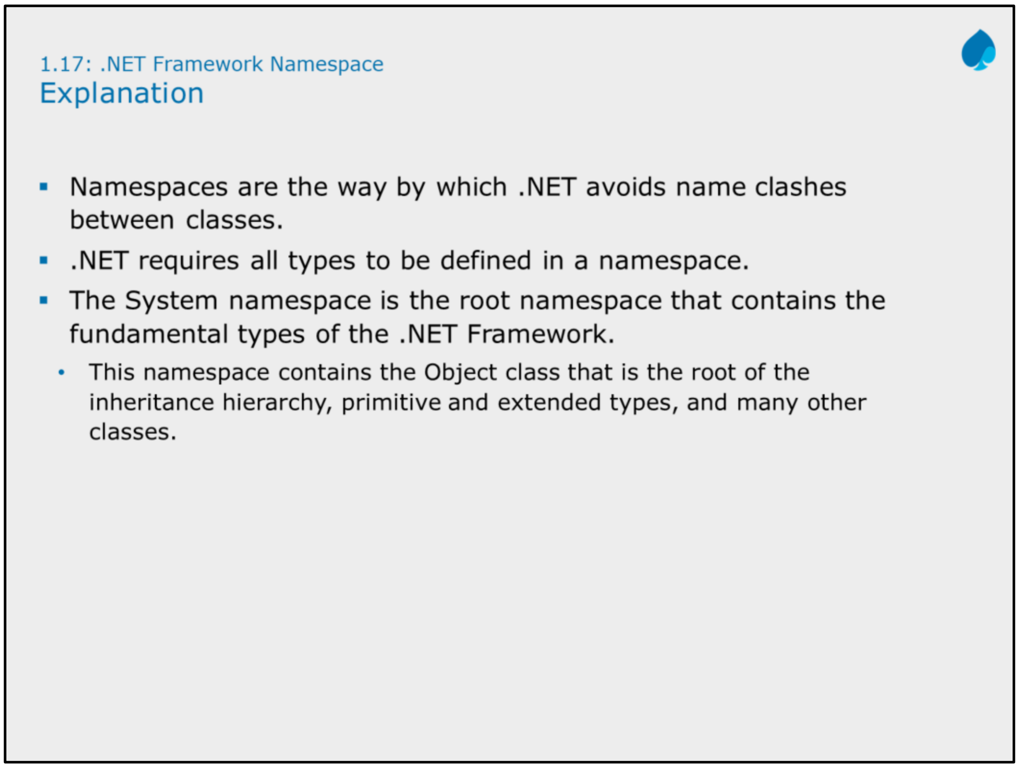


1.14: Compilation and Execution in .NET

Concept of .NET

- Assembly
  - When you compile an application, the CIL code created is stored in an assembly.
  - Assemblies include both executable application files that you can run directly from Windows without the need for any other programs (these have a .exe file extension) and libraries (which have a .dll extension) for use by other applications.
  - It is defined as the Smallest Unit of Deployment , Versioning and Sharing
- IL
  - The CPU independent Set of Binary Instructions generated by .NET Language Compiler
- Managed Code
  - Code written using the .NET Framework is managed when it is executed (a stage usually referred to as runtime).
  - This means that the CLR looks after your applications by managing memory, handling security, allowing cross-language debugging, and so on.

Compilation and Execution in .NET:
The managed execution process includes the following steps:
Designing and writing your source code
To obtain the benefits provided by the common language runtime, you must use one or more language compilers that target the runtime.
Compiling your code to Intermediate language (IL)
Compiling translates your source code into IL and generates the required metadata.
When compiling to managed code, the compiler translates your source code into intermediate language (IL), which is a CPU-independent set of instructions that can be efficiently converted to native code. IL includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations. Before code can be executed, MSIL must be converted to CPU-specific code by a just in time (JIT) compiler. Since the runtime supplies one or more JIT compilers for each computer architecture it supports, the same set of MSIL can be JIT-compiled and executed on any supported architecture.



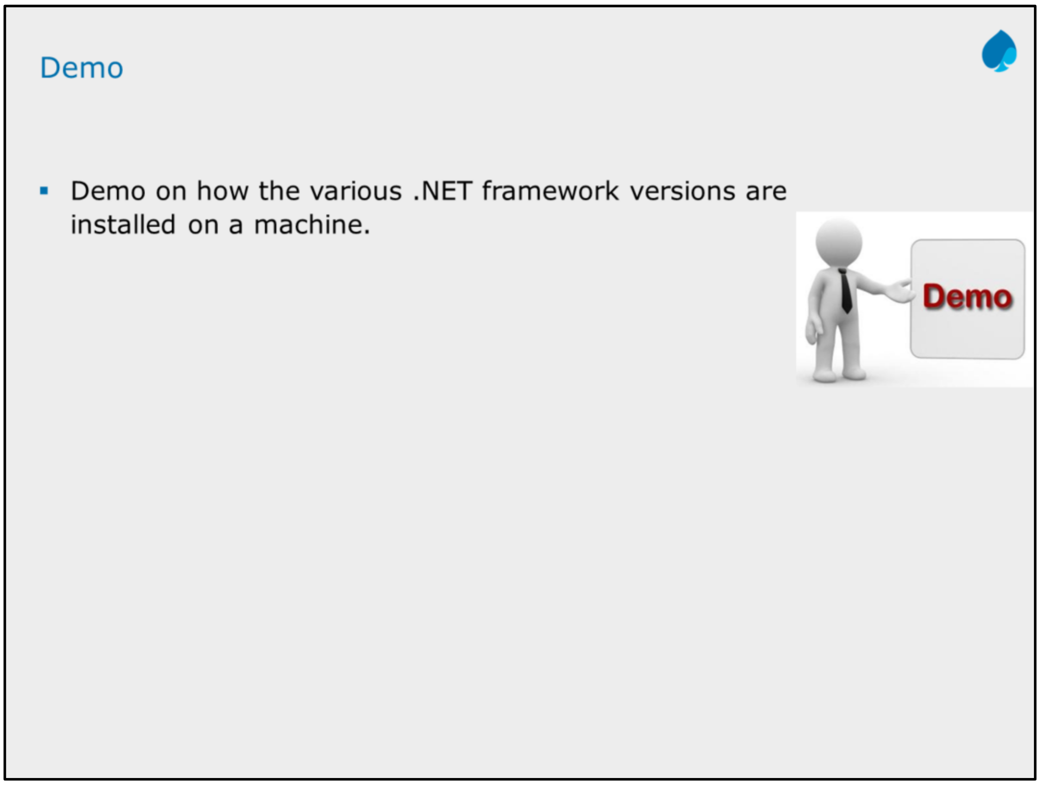
1.17: .NET Framework Namespace

Explanation

- Namespaces are the way by which .NET avoids name clashes between classes.
- .NET requires all types to be defined in a namespace.
- The System namespace is the root namespace that contains the fundamental types of the .NET Framework.
  - This namespace contains the Object class that is the root of the inheritance hierarchy, primitive and extended types, and many other classes.

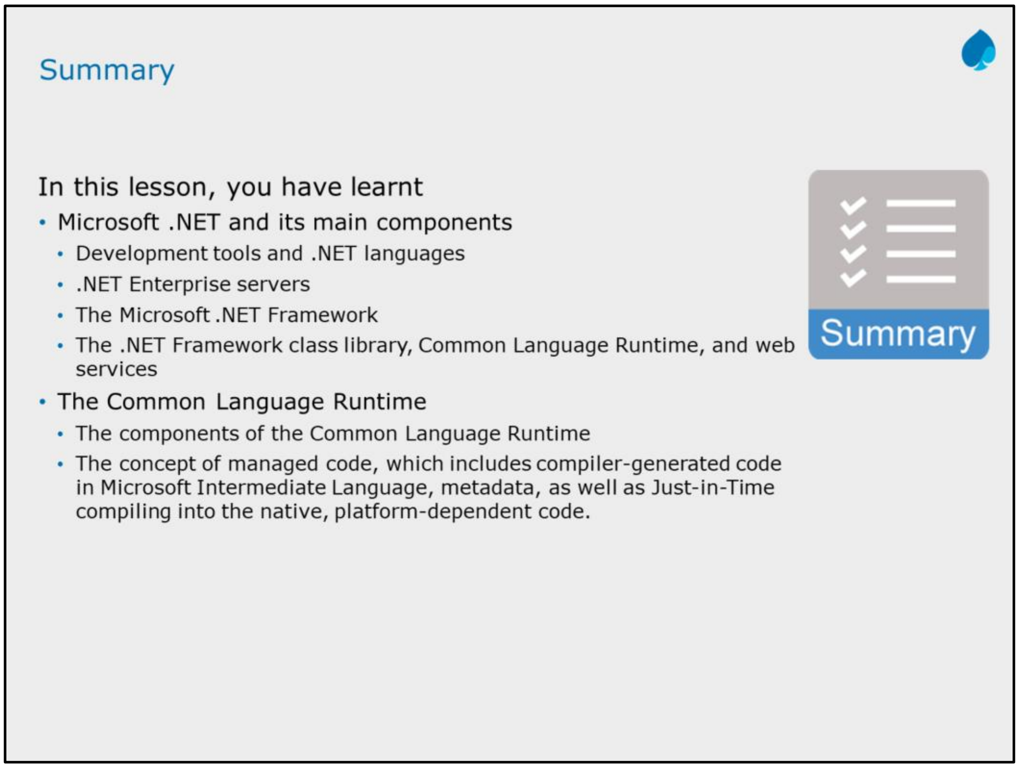
.NET Framework Namespace:

Namespaces are the way by which .NET avoids name clashes between classes. They are designed to avoid the situation in which you define a class to represent a customer, name your class Customer, and then someone else does the same thing (a likely scenario - the proportion of businesses that have customers seems to be quite high).
A namespace is no more than a grouping of data types, but it has the effect that the names of all data types within a namespace are automatically prefixed with the name of the namespace. It is also possible to nest namespaces within each other.
For example: Most of the general-purpose .NET base classes are in a namespace called System. The base class Array is in this namespace, so its full name is System.Array.
.NET requires all types to be defined in a namespace;
For example: You can place your Customer class in a namespace called YourCompanyName. This class will have the full name YourCompanyName.Customer.
The System namespace is the root namespace that contains the fundamental types of the .NET Framework. This namespace contains the Object class that is the root of the inheritance hierarchy, primitive and extended types, and many other classes. Indeed, there are almost 100 classes to handle exceptions, support runtime execution, application domains, garbage collection, and so on.



Demo

- Demo on how the various .NET framework versions are installed on a machine.



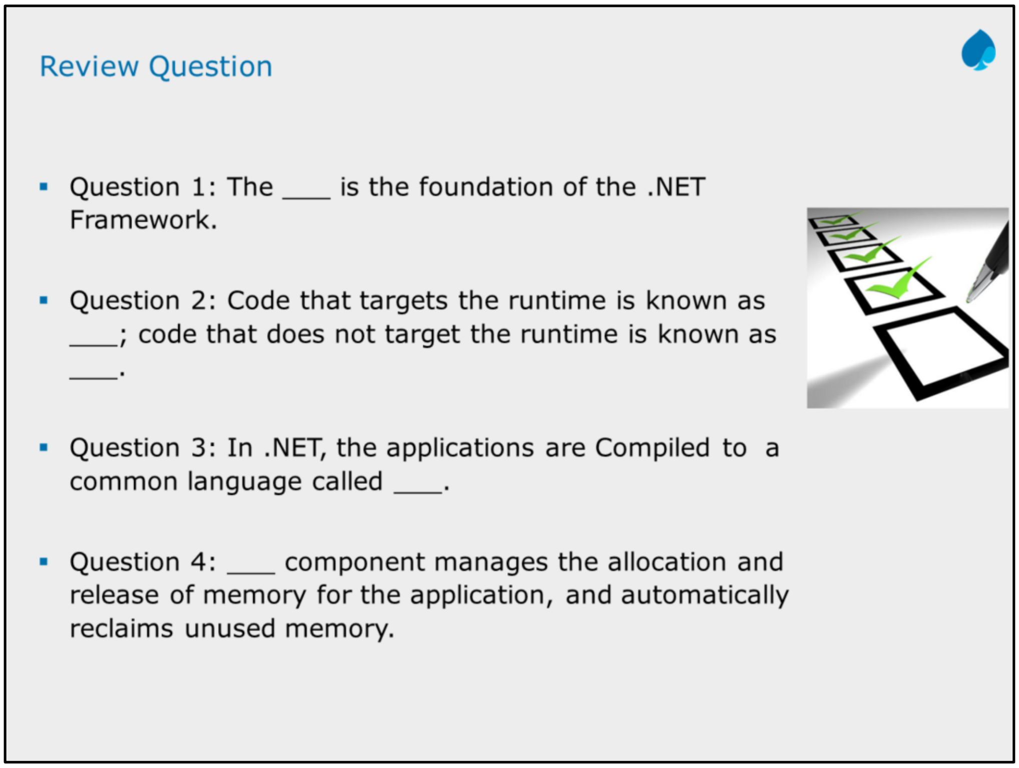
Summary

In this lesson, you have learnt

- Microsoft .NET and its main components
  - Development tools and .NET languages
  - .NET Enterprise servers
  - The Microsoft .NET Framework
  - The .NET Framework class library, Common Language Runtime, and web services
- The Common Language Runtime
  - The components of the Common Language Runtime
  - The concept of managed code, which includes compiler-generated code in Microsoft Intermediate Language, metadata, as well as Just-in-Time compiling into the native, platform-dependent code.

Summary

Add the notes here.



Review Question

- Question 1: The ____ is the foundation of the .NET Framework.
- Question 2: Code that targets the runtime is known as ____; code that does not target the runtime is known as ____.
- Question 3: In .NET, the applications are Compiled to a common language called ____.
- Question 4: ____ component manages the allocation and release of memory for the application, and automatically reclaims unused memory.