1. .NET Versions

.NET is a software development framework created by Microsoft. Over time, it evolved through several versions:

■ .NET Framework

- First released in 2002.
- Used mainly for Windows applications.
- Versions: 1.0 to 4.8 (latest and last major release).
- Includes support for WinForms, ASP.NET Web Forms, WPF, etc.

■ .NET Core

- Released in 2016 to support cross-platform development.
- Works on Windows, macOS, and Linux.
- Lightweight, modular, and faster.
- Removed many old legacy APIs.
- Versions: 1.0, 2.0, 3.1 (Long-Term Support - LTS).

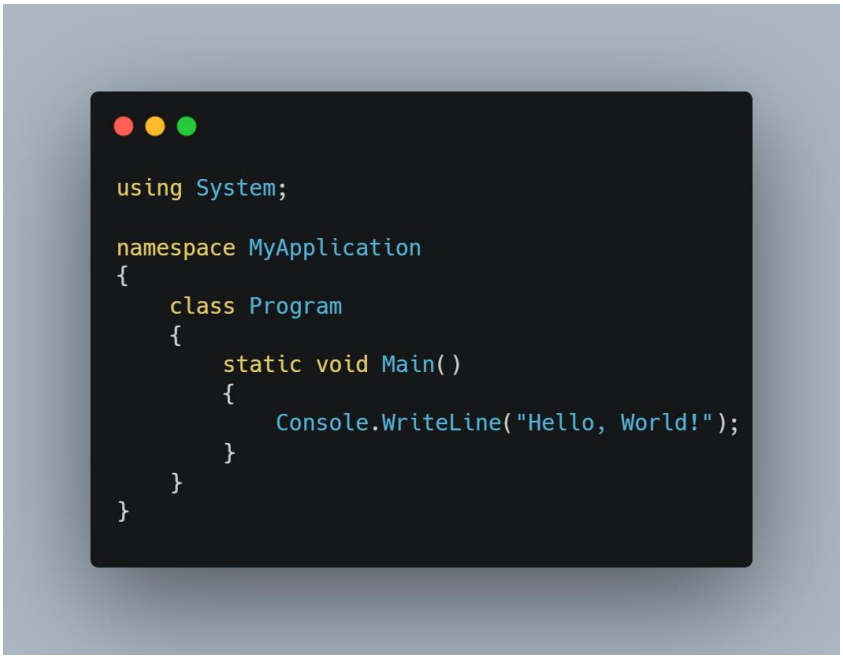
■ .NET 5 and Beyond

- Starting from .NET 5 (2020), Microsoft unified the platform.
- Replaces .NET Framework and .NET Core.
- Single platform for web, mobile, desktop, cloud, gaming, and IoT.
- Versions: .NET 5, 6 (LTS), 7, 8 (LTS).

2. Namespace in .NET

A **namespace** in .NET is a way to organize and group related classes, interfaces, enums, and other types to avoid naming conflicts.

📜 **Example:**



```
using System;

namespace MyApplication
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

✅ **Common Namespaces:**

- System: Base classes (like Console, Math, String)
- System.IO: For file handling
- System.Net: For networking
- System.Linq: For LINQ queries
- Microsoft.AspNetCore: For ASP.NET Core development

3. .NET Core

✅ What is .NET Core?

.NET Core is a **cross-platform**, **open-source**, and **modular** runtime developed by Microsoft.

✅ Key Features:

- Cross-platform (Windows, Linux, macOS)
- Command-line tools
- Side-by-side versioning
- High performance and scalable
- Used for ASP.NET Core, Blazor, Console apps, etc.

✅ Components:

- **CoreCLR**: Runtime
- **CoreFX**: Base class libraries
- **CLI**: Command-line tools
- **Roslyn**: Compiler

✅ Typical .NET Core Project:

```
dotnet new console -n HelloWorld

cd HelloWorld

dotnet run
```

4. Solution in .NET

A **Solution** (.sln file) in .NET is a container that groups multiple related projects.

✅ **Why use a Solution?**

- Manage large applications with many parts (UI, API, Class Library).
- Helps organize builds, references, and dependencies.

✅ **Structure:**

pgsql

CopyEdit

MySolution/

├── MyApp.Web/ → ASP.NET Core project

├── MyApp.Business/ → Business logic library

├── MyApp.Data/ → Data access library

├── MySolution.sln → Solution file

📜 **Example:**

- You open a .sln file in Visual Studio or VS Code to work on all the projects together.