**.NET Versions**

**Description:**  
.NET is a software development platform developed by Microsoft. It has evolved over time through multiple versions, each adding new features and capabilities. Below is a brief overview of the major versions:

**Versions:**

|  |  |  |
| --- | --- | --- |
| **Version** | **Release Year** | **Description** |
| .NET Framework 1.0 | 2002 | The initial release; supported Windows desktop apps. |
| .NET Framework 3.5 | 2007 | Introduced LINQ and better WPF support. |
| .NET Framework 4.0/4.5 | 2010-2012 | Included TPL (Task Parallel Library) and async/await. |
| .NET Core 1.0 | 2016 | Cross-platform, modular, lightweight. A major rewrite. |
| .NET Core 3.0 | 2019 | Added support for desktop apps (WPF/WinForms). |
| .NET 5 | 2020 | Unified platform replacing .NET Framework and .NET Core. |
| .NET 6 (LTS) | 2021 | First LTS in unified platform; improved performance. |
| .NET 7 | 2022 | Short-term support; minor improvements. |
| .NET 8 (LTS) | 2023 | Major performance and cloud-native enhancements. |
| .NET 9 | 2024 | (Planned) Continues .NET evolution with AI support. |

**Notes:**

* Versions starting from .NET 5 use a unified platform.
* LTS versions are recommended for enterprise applications.

**Namespaces in .NET**

**Description:**  
Namespaces in .NET are used to organize code and avoid name conflicts by grouping logically related classes, interfaces, structs, enums, and delegates.

**Example:**

using System;

namespace MyCompany.MyApp.Models

{

public class Employee

{

public string Name {

get;

set;

}

}

}

**Features:**

* Declared using the namespace keyword.
* Provide logical separation.
* Help manage large codebases.
* Can be nested or extended across files.

**Common .NET Namespaces:**

|  |  |
| --- | --- |
| **Namespace** | **Purpose** |
| System | Core classes (strings, arrays, exceptions) |
| System.IO | File input/output operations |
| System.Collections.Generic | Generic collections like List<T> |
| Microsoft.AspNetCore.Mvc | ASP.NET Core MVC functionalities |
| System.Linq | LINQ querying capabilities |

**.NET Core**

**Description:**  
.NET Core is a free, open-source, cross-platform development platform developed by Microsoft. It is a major part of the .NET ecosystem and was the foundation for the unified .NET 5 and later.

**Features:**

* Cross-platform (Windows, macOS, Linux)
* High performance
* Command-line interface (CLI) tools
* Modular via NuGet packages
* Supports microservices and cloud-native development

**Supported Application Types:**

* Console Apps
* Web APIs
* Web Applications (Razor Pages, MVC, Blazor)
* Windows Desktop Apps (from .NET Core 3.0 onward)

**Architecture:**

* Runtime: CoreCLR
* Libraries: CoreFX
* Tools: dotnet CLI

**Example CLI Commands:**

dotnet new console -n MyApp

dotnet build

dotnet run

**End of Life:**  
.NET Core 3.1 reached end-of-life in December 2022. Migration to .NET 6 or later is recommended.

**Solution in .NET**

**Description:**  
A **Solution** in .NET is a container for organizing related projects. It is represented by a .sln file and is used primarily in Visual Studio and compatible IDEs.

**Features:**

* Can contain multiple projects (e.g., Web App, Class Library, Unit Tests)
* Manages project references and dependencies
* Helps in building, debugging, and deploying related components together

**Typical Structure:**

MySolution/

│

├── MySolution.sln

├── ProjectA/

│ └── ProjectA.csproj

├── ProjectB/

│ └── ProjectB.csproj

**Benefits:**

* Centralized management of builds and settings.
* Logical grouping of components.
* Enables shared references and versioning.

**Managing Solutions:**

* Created via Visual Studio or dotnet CLI:

dotnet new sln -n MySolution

dotnet sln add ./ProjectA/ProjectA.csproj