**.Net Versions report**

**.NET** is a software framework developed by Microsoft, providing a platform for building and running applications across multiple environments. Below is an overview of its evolution:

**1. .NET Framework (2002 - Present)**

* **Initial Release**: .NET Framework 1.0 (2002)
* **Features**:
  + Windows-only support.
  + Uses CLR (Common Language Runtime).
  + ASP.NET for web development.
  + WinForms and WPF for desktop apps.
* **Key Versions**:
  + **1.0-3.5**: Introduced ASP.NET, LINQ, and WPF.
  + **4.0-4.8**: Added Entity Framework, async programming, and enhanced WPF.

**2. .NET Core (2016-2020)**

* **Initial Release**: .NET Core 1.0 (2016)
* **Features**:
  + Cross-platform (Windows, macOS, Linux).
  + Modular and lightweight.
  + CLI (Command Line Interface).
  + High-performance APIs (Kestrel).
* **Key Versions**:
  + **1.x**: Minimal features.
  + **2.x**: Added SignalR, Razor Pages.
  + **3.x**: Support for Windows desktop apps (WPF, WinForms).

**3. .NET 5 (2020)**

* Unified .NET Core and .NET Framework.
* Cross-platform, single runtime for all app types (desktop, web, mobile, cloud).
* Features: Improved performance, C# 9.0, and gRPC support.

**4. .NET 6 (2021)**

* **LTS (Long-Term Support)** version.
* Features:
  + C# 10.0.
  + MAUI (Multi-platform App UI) for cross-platform UIs.
  + Performance optimizations.

**5. .NET 7 (2022)**

* Non-LTS release.
* Enhanced .NET MAUI and improved cloud-native development.

**6. .NET 8 (2023)**

* LTS release with a focus on AI integration, better performance, and seamless cloud development.

**Namespace**

A **namespace** in C# is a way to organize and group related classes, interfaces, structs, enums, and methods logically. It helps avoid naming conflicts in large projects.

**1. Purpose of Namespaces**

* **Organization**: Keeps code modular and manageable.
* **Avoid Naming Conflicts**: Prevents clashes between identifiers in different parts of a program.
* **Reusability**: Allows the same class name in different namespaces.

**Advantages:**

* Simplifies code readability.
* Makes large projects scalable.
* Prevents duplication and confusion in naming.

### ****2. Declaring a Namespace****

A namespace is declared using the namespace keyword:

**namespace MyNamespace**

**{**

**class MyClass**

**{**

**public void MyMethod()**

**{**

**Console.WriteLine("Hello from MyNamespace!");**

**}**

**}**

**}**

### ****3. Using a Namespace****

Namespaces are accessed using the using keyword:

**using MyNamespace;**

**class Program**

**{**

**static void Main()**

**{**

**MyClass obj = new MyClass();**

**obj.MyMethod();**

**}**

**}**

### ****4. Nested Namespaces****

Namespaces can be nested for further organization:

**namespace OuterNamespace**

**{**

**namespace InnerNamespace**

**{**

**class InnerClass { }**

**}**

**}**

**Access:**

**using OuterNamespace.InnerNamespace;**

**.Net Core**

**1. Overview**

* **Initial Release**: June 2016 (.NET Core 1.0).
* **Purpose**: A lightweight, modular, and cross-platform alternative to the .NET Framework.
* **Target Applications**: Web, cloud, microservices, console, IoT, and more.

**2. Key Features**

1. **Cross-Platform**:
   * Runs on **Windows**, **macOS**, and **Linux**.
   * Enables building applications for multiple platforms using a single codebase.
2. **Modular Architecture**:
   * Uses **NuGet packages** for adding specific features, reducing overhead.
3. **High Performance**:
   * Optimized runtime for fast execution.
   * Ideal for **microservices** and **cloud-native** applications.
4. **Open Source**:
   * Hosted on GitHub with community contributions.
5. **Compatibility**:
   * Supports multiple languages: C#, F#, and VB.NET.
   * Works seamlessly with tools like **Visual Studio** and **Visual Studio Code**.
6. **Command-Line Interface (CLI)**:
   * Provides a CLI for development, build, and deployment tasks (dotnet commands).

### ****3. Versions and Evolution****

| **Version** | **Release Year** | **Key Features** |
| --- | --- | --- |
| **.NET Core 1.x** | 2016 | Basic features, limited APIs. |
| **.NET Core 2.x** | 2017 | Added SignalR, Razor Pages, more APIs. |
| **.NET Core 3.x** | 2019 | WPF, WinForms support, Blazor introduced. |

After .NET Core 3.1 (LTS), it was unified into **.NET 5**, merging with .NET Framework.

**Solution**

A **solution** in .NET is a container that organizes and manages one or more related projects in a structured manner. It serves as the starting point for large-scale development in **Visual Studio** or **.NET CLI**.

**Key Features:**

1. **Multiple Project Management**:
   * Combines multiple projects (e.g., web app, API, library) into a single workspace.
   * Files have a .sln extension.
2. **Dependency Handling**:
   * Manages project references and dependencies between projects.
3. **Centralized Build**:
   * Builds all projects in the solution together for consistency.
4. **Configurations**:
   * Supports multiple build configurations (e.g., Debug, Release).

**Structure:**

1. **Solution File (.sln)**:
   * Defines solution-wide settings and project paths.
2. **Projects**:
   * Each project has its own files (e.g., .csproj) and settings.

**Advantages:**

* Simplifies managing large applications.
* Supports modular development (e.g., separating layers into projects).
* Easier team collaboration.

**self>>overhead at runtime (jitting)>> what is done to reduce this?**

**Strategies to Reduce JIT Overhead:**

1. **Ahead-of-Time (AOT) Compilation**:
   * Compiles IL to native code **before runtime**.
   * Example: **ReadyToRun (R2R)** in .NET Core.
2. **Tiered Compilation**:
   * Initially uses a quick, less-optimized JIT.
   * Optimizes hot code paths during execution.
3. **Cold Code Elimination**:
   * Reduces JIT work by skipping infrequently executed code.
4. **Profile-Guided Optimization (PGO)**:
   * Optimizes based on runtime performance data to prioritize frequently used methods.
5. **Precompilation with ngen** (for .NET Framework):
   * Generates native images at install time to avoid JIT.
6. **Caching**:
   * Keeps compiled machine code in memory for reuse.