**Self Studying**

**What is the Overhead in JIT?**

When you run a .NET program, the code isn’t immediately converted to machine instructions. Instead, it's first compiled into IL (Intermediate Language), and then at runtime, it's converted into native code using the JIT compiler. This process introduces delay the first time a method is executed — and this is the runtime overhead.

**What is Done to Reduce This Overhead?**

1. Pre-JIT Compilation using ReadyToRun (R2R)

* You can compile code ahead of time to native during deployment using tools like:
  + crossgen
  + dotnet publish -r win-x64 --self-contained
* This reduces or eliminates the need for JIT by delivering pre-compiled native code.

1. Tiered Compilation

* Introduced in .NET Core, Tiered JIT Compilation means:
  + Methods are first compiled quickly (less optimized).
  + If a method is called frequently, it’s recompiled in a more optimized way.
* This balances startup speed with long-term performance.

1. AOT Compilation (Ahead-Of-Time)

* In .NET 6/7/8, Native AOT allows:
  + Full compilation to native code before runtime.
  + Eliminates JIT completely.
  + Results in faster startup and smaller footprint — ideal for containers or mobile.

1. Code Warm-Up

* You can perform manual warm-up by calling frequently used methods at startup.
* This triggers JIT early so later calls are faster.

1. Avoiding Dynamic Features

* Features like dynamic, Reflection.Emit, or Expression.Compile() increase JIT overhead.
* Reducing or avoiding them improves performance.

| **الحل** | **بيعمل إيه؟** | **مفيد إمتى؟** |
| --- | --- | --- |
| ReadyToRun | يحضّر الكود قبل التشغيل | لتقليل وقت تشغيل أولي |
| Tiered JIT | يترجم بسرعة ثم يحسّن | برامج كبيرة أو طويلة المدى |
| Native AOT | يجهز الكود بالكامل من الأول | برامج حساسة وسريعة جدًا |
| Code Warm-up | يشغل الكود في الخلفية | ألعاب/تطبيقات تحتاج أداء سريع |
| قلل dynamic | تقلل جهد المعالج | دائمًا مفيد |

**report about Dot net version , namespace ,.net core and solution ?**

**1. .NET Versions**

**.NET is a software development platform created by Microsoft. It allows developers to build applications for Windows, Linux, macOS, web, mobile, cloud, gaming, and IoT.**

| **Version** | **Description** |
| --- | --- |
| **.NET Framework** | **The original version (since 2002) – Windows-only** |
| **.NET Core** | **Cross-platform, open-source rewrite introduced in 2016** |
| **.NET 5** | **First unified version combining .NET Core and .NET Framework features** |
| **.NET 6 / 7 / 8** | **Latest LTS (Long-Term Support) releases with performance and API improvements** |

**2. Namespaces in .NET**

**A namespace is a logical grouping of related classes, interfaces, structs, enums, and delegates.**

**🧾 Purpose of Namespaces:**

* **Avoid name conflicts between classes.**
* **Organize code in a readable and maintainable structure.**

| **Namespace** | **Purpose** |
| --- | --- |
| **System** | **Core functions (data types, exceptions, console)** |
| **System.Collections** | **Data structures like List, Dictionary** |
| **System.IO** | **File and stream I/O operations** |
| **System.Net** | **Networking functionality** |
| **Microsoft.AspNetCore** | **Web development in ASP.NET Core** |

**3. .NET Core**

**.NET Core is the modern, cross-platform version of .NET.**

**🔧 Key Features:**

* **Cross-platform (Windows, Linux, macOS)**
* **Open-source**
* **Fast and lightweight**
* **Suitable for web, microservices, and cloud-native applications**

**📦 Technologies in .NET Core:**

* **ASP.NET Core: For building web applications**
* **Entity Framework Core: For data access**
* **Blazor: Web UI framework using C#**

**4. Solution and Project Structure in .NET**

**A Solution (.sln) is a container that holds one or more projects in Visual Studio or .NET CLI.**

**🧱 Structure:**

* **Solution (.sln): The top-level file that groups projects.**
* **Projects (.csproj): Contain actual code, dependencies, and settings.**

**🔄 Types of Projects:**

* **Console App (.NET Console Application)**
* **Web App (ASP.NET Core Web App)**
* **Class Library (.NET Class Library)**

**⚙ Tools:**

* **Visual Studio (GUI)**
* **dotnet CLI (Command Line Interface)**

| **Topic** | **Summary** |
| --- | --- |
| **.NET Versions** | **Transitioned from .NET Framework to modern unified .NET (Core → 5/6/7/8)** |
| **Namespaces** | **Organize code logically and prevent naming conflicts** |
| **.NET Core** | **Fast, cross-platform, modular framework used for modern development** |
| **Solution** | **A container for one or more projects, used to organize large applications** |