20 .NET Core Interview Questions and Answers

fullstack.cafe/blog/dot-net-core-interview-questions-and-answers

Kill Your Tech Interview

- .NET Core 68
- ADO.NET 33
- ASP.NET 54
- ASP.NET MVC 36
- ASP.NET Web API 33
- Agile & Scrum 47
- Angular 120
- Azure 35
- C# 127
- Dependency Injection 17
- Design Patterns 68
- DevOps 44
- Entity Framework 57
- Git 36
- HTML5 55
- JavaScript 179
- Kotlin 68
- LINQ 38
- MongoDB 77
- MySQL 60
- Node.js 126
- PHP 82
- PWA 22

.NET Core 68
ASP.NET 54

Q1:
What is .NET Core?

Answer

Entry
.NET Core 68

The .NET Core platform is a new .NET stack that is optimized for open source development and agile delivery on NuGet.

.NET Core has two major components. It includes a small runtime that is built from the same codebase as the .NET Framework CLR. The .NET Core runtime includes the same GC and JIT (RyuJIT), but doesn't include features like Application Domains or Code Access Security. The runtime is delivered via NuGet, as part of the ASP.NET Core package.

.NET Core also includes the base class libraries. These libraries are largely the same code as the .NET Framework class libraries, but have been factored (removal of dependencies) to enable to ship a smaller set of libraries. These libraries are shipped as `System.*` NuGet packages on NuGet.org.

Q2:
Can ASP.NET Core work with the .NET framework?

Answer

Junior ASP.NET 54

Yes. This might surprise many, but ASP.NET Core works with .NET framework and this is officially supported by Microsoft.

ASP.NET Core works with:

- .NET Core framework
- .NET framework

Q3:
What are some characteristics of .NET Core?

Answer

Junior .NET Core 68

- **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 OSes. The supported Operating Systems (OS), CPUs and application scenarios will grow over time, provided by Microsoft, other companies, and individuals.

- **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 Library.

- **Open source**: The .NET Core platform is open source, using MIT and Apache 2 licenses. Documentation is licensed under CC-BY. .NET Core is a .NET Foundation project.

- **Supported by Microsoft**: .NET Core is supported by Microsoft, per .NET Core Support

---

Q4:
What is CTS?

Answer

Junior
.NET Core 68

The **Common Type System (CTS)** standardizes the data types of all programming languages using .NET under the umbrella of .NET to a common data type for easy and smooth communication among these .NET languages.

CTS is designed as a singly rooted object hierarchy with `System.Object` as the base type from which all other types are derived. CTS supports two different kinds of types:

1. **Value Types**: Contain the values that need to be stored directly on the stack or allocated inline in a structure. They can be built-in (standard primitive types), user-defined (defined in source code) or enumerations (sets of enumerated values that are represented by labels but stored as a numeric type).
2. **Reference Types**: Store a reference to the value's memory address and are allocated on the heap. Reference types can be any of the pointer types, interface types or self-describing types (arrays and class types such as user-defined classes, boxed value types and delegates).

Q5:
What is the difference between .NET Core and Mono?

Answer

Junior
.NET Core 68

To be simple:

- Mono is third party implementation of .Net Framework for Linux/Android/iOs
- .Net Core is Microsoft's own implementation for same.

Q6:
What's the difference between *SDK* and *Runtime* in .NET Core?

Answer

Junior
.NET Core 68

- The SDK is all of the stuff that is needed/makes developing a .NET Core application easier, such as the CLI and a compiler.

- The runtime is the "virtual machine" that hosts/runs the application and abstracts all the interaction with the base operating system.

Q7: Explain the difference between `Task` and `Thread` in .NET

Answer

Mid

.NET Core 68

- **Thread** represents an actual OS-level thread, with its own stack and kernel resources. Thread allows the highest degree of control; you can `Abort()` or `Suspend()` or `Resume()` a thread, you can observe its state, and you can set thread-level properties like the stack size, apartment state, or culture. `ThreadPool` is a wrapper around a pool of threads maintained by the CLR.

- The **Task class** from the Task Parallel Library offers the best of both worlds. Like the `ThreadPool`, a task does not create its own OS thread. Instead, tasks are executed by a `TaskScheduler`; the default scheduler simply runs on the ThreadPool. Unlike the ThreadPool, Task also allows you to find out when it finishes, and (via the generic Task) to return a result.

Q8: Explain what is included in .NET Core?

Answer

Mid

.NET Core 68

- A .NET runtime, which provides a type system, assembly loading, a garbage collector, native interop and other basic services.

- A set of framework libraries, which provide primitive data types, app composition types and fundamental utilities.

- A set of SDK tools and language compilers that enable the base developer experience, available in the .NET Core SDK.

- The 'dotnet' app host, which is used to launch .NET Core apps. It selects the runtime and hosts the runtime, provides an assembly loading policy and launches the app. The same host is also used to launch SDK tools in much the same way.

Kotlin 68
ASP.NET Web API 33
iOS 36
UX Design 78
ASP.NET MVC 36
HTML5 55
Golang 49
ASP.NET 54
Flutter 67
WCF 37

Q9:
What are the benefits of *Explicit Compilation (AOT)*?

Answer

Mid
.NET Core 68

Ahead of time (AOT) delivers **faster start-up time**, especially in large applications where much code executes on startup. But it requires more disk space and more memory/virtual address space to keep both the IL and precompiled images. In this case the JIT Compiler has to do a lot of disk I/O actions, which are quite expensive.

Q10:
What is CoreCLR?

Answer

Mid
.NET Core 68

CoreCLR is the .NET execution engine in .NET Core,
performing functions such as garbage collection and compilation to machine code.

Consider:

Q11:
What is JIT compiler?

Answer

Mid
.NET Core 68

Before a computer can execute the source code, special programs called compilers must rewrite it into machine instructions, also known as object code. This process (commonly referred to simply as "compilation") can be done explicitly or implicitly.

Implicit compilation is a two-step process:

- The first step is converting the source code to intermediate language (IL) by a language-specific compiler.
- The second step is converting the IL to machine instructions. The main difference with the explicit compilers is that only executed fragments of IL code are compiled into machine instructions, at runtime. The .NET framework calls this compiler the **JIT (Just-In-Time) compiler**.

Q12:
What is Kestrel?

Answer

Mid
.NET Core 68

- Kestrel is a cross-platform web server built for ASP.NET Core based on libuv – a cross-platform asynchronous I/O library.
- It is a default web server pick since it is used in all ASP.NET Core templates.
- It is really fast.
- It is secure and good enough to use it without a reverse proxy server. However, it is still recommended that you use IIS, Nginx or Apache or something else.

Q13:
What is difference between .NET Core and .NET Framework?

Answer

Mid
.NET Core 68

.NET as whole now has 2 flavors:

- .NET Framework
- .NET Core

.NET Core and the *.NET Framework* have (for the most part) a subset-superset relationship. .NET Core is named “Core” since it contains the core features from the .NET Framework, for both the runtime and framework libraries. For example, .NET Core and the .NET Framework share the GC, the JIT and types such as `String` and `List`.

.NET Core was created so that .NET could be open source, cross platform and be used in more resource-constrained environments.

Q14:
What's the difference between .NET Core, .NET Framework, and Xamarin?

Answer

Mid
.NET Core 68

- **.NET Framework** is the "full" or "traditional" flavor of .NET that's distributed with Windows. Use this when you are building a desktop Windows or UWP app, or working with older ASP.NET 4.6+.
- **.NET Core** is cross-platform .NET that runs on Windows, Mac, and Linux. Use this when you want to build console or web apps that can run on any platform, including inside Docker containers. This does not include UWP/desktop apps currently.
- **Xamarin** is used for building mobile apps that can run on iOS, Android, or Windows Phone devices.

Xamarin usually runs on top of Mono, which is a version of .NET that was built for cross-platform support before Microsoft decided to officially go cross-platform with .NET Core. Like Xamarin, the Unity platform also runs on top of Mono.

Q15:
When should we use .NET Core and .NET Standard Class Library project types?

Answer

Mid

.NET Core 68

- Use a **.NET Standard** library when you want to increase the number of apps that will be compatible with your library, and you are okay with a decrease in the .NET API surface area your library can access.
- Use a **.NET Core** library when you want to increase the .NET API surface area your library can access, and you are okay with allowing only .NET Core apps to be compatible with your library.
