Shaheen Aziz • 2nd
.NET Core | MVC | Web API | Microservices | EF Core | C# | Git | SQL| Angular| Types...
1mo •

Follow ···

🚀 50 Essential C# Interview Questions & Answers

🔷 What is C#?

A modern, object-oriented language by Microsoft, part of the .NET ecosystem.

🔷 String vs. StringBuilder?

String is immutable; StringBuilder allows efficient modifications.

🔷 const vs. readonly?

const is compile-time constant; readonly can be set at runtime.

🔷 using statement?

Manages namespaces and auto-disposes objects.

🔷 Boxing & Unboxing?

Converting value type ↔ reference type.

🔷 var keyword?

Implicitly types variables based on assigned values.

🔷 async & await?

For asynchronous programming without blocking threads.

🔷 == vs. Equals()?

== compares values, Equals() compares object contents.

🔷 Delegate?

A reference type for method pointers.

🔷 LINQ?

Querying collections like databases.

🔷 Garbage Collection?

Automatic memory management in .NET.

🔷 Singleton Pattern?

Ensures only one instance of a class.

🔷 throw vs. throw ex?

throw keeps stack trace; throw ex resets it.

🔷 try-catch-finally?

Handles exceptions, with finally ensuring execution.

🔷 sealed keyword?

Prevents class inheritance.

🔷 Interface?

Defines methods without implementations.

🔷 Method Overloading?

Same method name, different parameters.

🔷 out keyword?

Passes parameters by reference.

🔷 Property?

Encapsulates private fields with getters & setters.

🔷 IEnumerable?

Allows collection iteration.

🔷 Indexers?

Objects behave like arrays.

🔷 Events?

Enable notifications between objects.

🔷 lock statement?

Prevents race conditions in multithreading.

🔷 params keyword?

Allows a variable number of arguments.

🔷 Nullable types?

Value types that can store null.

🔷 ref keyword?

Passes variables by reference.

🔷 Interface vs. Abstract Class?

Interface = no implementation, abstract class = partial implementation.

🔷 yield keyword?

Simplifies custom iterators.

🔷 Access Modifiers?

public, private, protected, internal, etc.

🔷 base keyword?

Accesses the parent class.

🔷 this keyword?

Refers to the current instance.

🔷 Polymorphism?

Allows different implementations under a common interface.

🔷 String.Split()?

Splits a string into an array.

🔷 Dispose()?

Releases unmanaged resources.

🔷 Generics?

Enable reusable type-safe methods/classes.

🔷 Nullable<T>?

Represents value types that can be null.

🔷 volatile keyword?

Prevents compiler optimizations in multi-threading.

🔷 as keyword?

Safe type casting.

🔷 static keyword?

Defines class-level members.

🔷 Value Type vs. Reference Type?

Value types store data, reference types store memory addresses.

🔷 using & IDisposable?

Ensures proper resource disposal.

🔷 IoC (Inversion of Control)?

Dependency injection for better architecture.

🔷 Extension Methods?

Add methods to existing types without modifying them.

🔷 nameof operator?

Returns the name of a variable or type as a string.

🔷 async/await pattern?

Manages asynchronous operations efficiently.

🔷 Thread class?

Manages multi-threading in C#.

#CSharp #DotNet #Coding #SoftwareDevelopment