Day 44/60 - Difference Between ReadOnly, Const, and Static in C#

Understanding const, readonly, and static is important for managing data immutability and memory efficiency in C#. Here's how they differ:

const (Compile-Time Constant)

Value must be assigned at declaration and cannot change.
Stored in assembly metadata (not in memory).
Evaluated at compile-time, meaning changes require recompilation.

```
const double Pi = 3.14159; // Fixed at compile-time
```

Use it for truly constant values like mathematical constants.

readonly (Runtime Constant)

Value can be assigned in the constructor, but not modified afterward.
Evaluated at runtime, making it flexible.
Used for instance fields that should not change after initialization.

```
public class Example
{
 public readonly int MaxLimit;
 public Example(int limit) { MaxLimit = limit; } // Can be assigned in constructor
}
```

Use it when a value needs to be set at runtime and should remain unchanged.

static (Shared Across Instances)

Belongs to the class itself, not instances.
Shared across all objects of the class.
Used for utility methods and global states.

```
public static class Logger
{
 public static void Log(string message) => Console.WriteLine(message);
}
```

Use it for methods and fields that don't depend on instance data.

Summary:
Use const for fixed values like Pi.
Use readonly for values that should stay unchanged after initialization.
Use static for shared data and utility methods.

Choosing the right one improves performance, readability, and maintainability in your .NET applications!

#dotnet #csharp #constants #static #readonly