Day 51/60 - Understanding the Difference Between var, dynamic, and object in C#

C# provides different ways to declare variables with flexible typing: var, dynamic, and object. Understanding when to use each can improve performance, maintainability, and readability.

var (Implicitly Typed, Compile-Time Type Checking)

Type is determined at compile-time.
Offers strong typing, meaning errors are caught before execution.
Cannot change type after assignment.

Example:

```
var number = 10; // Treated as int
var text = "Hello"; // Treated as string
```

Best for: When the type is clear from the right-hand side and does not need to change.

object (Base Type for All Types, Runtime Boxing/Unboxing)

Can hold any type, but requires casting for type-specific operations.
Used for generic programming.
Boxing/unboxing occurs for value types, impacting performance.

Example:

```
object obj = 10;
int num = (int)obj; // Requires explicit casting
```

Best for: Generic collections (List<object>) and when working with unknown types.

dynamic (Runtime Type Checking, No Compile-Time Safety)

Type is resolved at runtime, meaning no compile-time type checking.
Allows changing types dynamically.
Can lead to runtime errors if used incorrectly.

Example:

```
dynamic value = 10;
value = "Hello"; // No error at compile-time
Console.WriteLine(value.Length); // Works only if value is a string
```

Risk: No IntelliSense or compile-time validation.

When to Use What?

Use var when the type is clear and you want type safety.
Use object for generic programming, but be mindful of casting.
Use dynamic for working with dynamic data (JSON, COM objects, etc.), but avoid excessive use to prevent runtime errors.

Choosing the right variable type makes your code efficient, readable, and maintainable!

#dotnet #csharp #codingtips #dynamic #object #var