**var Keyword in C#**

The var keyword in C# is used for **implicitly typed local variables**. This means the compiler determines the type of the variable at **compile-time** based on the value assigned to it.

**Syntax**

csharp

CopyEdit

var variableName = value;

* The variable **must be initialized** at the time of declaration.
* The **data type cannot change** after initialization.

**Example**

csharp

CopyEdit

var number = 10; // Compiler infers int

var name = "John"; // Compiler infers string

var isActive = true; // Compiler infers bool

**Key Features of var**

✅ **Type Inference** → The compiler automatically detects the type.  
✅ **Improves Code Readability** (when used correctly).  
✅ **Reduces Repetitive Type Definitions** in complex object initializations.

🚫 **Cannot be Used for Class Fields** → Only for **local variables**.  
🚫 **Cannot be Assigned null Without Explicit Type Casting**.  
🚫 **Cannot be Used Without Initialization**.

**Example: Implicit vs Explicit Typing**

csharp

CopyEdit

var age = 25; // Implicitly typed (int)

int age2 = 25; // Explicitly typed (int)

Both variables are of type int, but var allows cleaner code when the type is obvious.

**Using var with Collections**

csharp

CopyEdit

var numbers = new List<int> { 1, 2, 3, 4 };

Here, numbers is automatically inferred as List<int>.

**Best Practices**

✔ Use var when the type is **clear** and improves readability.  
✔ Avoid var when it reduces code clarity (e.g., primitive types).  
✔ Use var with **anonymous types and LINQ queries**.

**Example: var in LINQ**

csharp

CopyEdit

var students = new List<string> { "Alice", "Bob", "Charlie" };

var result = students.Where(s => s.StartsWith("A"));

foreach (var student in result)

{

Console.WriteLine(student);

}

Here, result is automatically inferred as an IEnumerable<string>.

**Conclusion**

* var helps in **clean, readable, and maintainable** code.
* It is **type-safe** because the compiler assigns the type at compile-time.
* Use it **where it makes sense**, but don't overuse it.