**Snippet 02 — “var” keyword (implicit typing)**

* **Code Recap**

var myString = "string value";

var myNumber = 123;

1. **Working Theory (keywords & concepts)**

**var**

* **Meaning:** compiler infers the type from the assigned value.
* var myString = "string value"; → compiler sees a string literal, so type is **string**.
* var myNumber = 123; → compiler sees an integer literal, so type is **int**.
* **Not dynamic:** once the type is inferred, it is fixed.

var x = 10;   // x is int

// x = "hello"; ❌ error: cannot assign string to int

**When to use var**

* ✅ when the type is **obvious** from the right-hand side:

var name = "Anas";  // clearly a string

var count = 5;      // clearly an int

* ✅ when the type is **long or complex** (like LINQ queries or generics):

var result = customers.Where(c => c.Age > 18);

* ❌ avoid when it makes code **less clear**:

var x = DoSomething(); // unclear what type x is!

**Literal values**

* "string value" → a **string literal** (sequence of characters).
* 123 → an **integer literal** (default type is int).

**Explicit typing alternative**

You could also write:

string myString = "string value";

int myNumber = 123;

Both versions compile the same way. The choice is about **readability and style**.

**Practical (from scratch)**

1. **Create project**

* dotnet new console -n Snippet02Demo
* cd Snippet02Demo

1. **Program.cs**

using System;

class Program

{

    static void Main()

    {

        var myString = "String Value";

        var myInteger = 123;

        Console.WriteLine($"Value of String = {myString}");

        Console.WriteLine($"Value of Integer = {myInteger}");

        // Explicit typing for Comparison

        string explicitSting = "another String";

        int explicitInteger = 456;

        Console.WriteLine($"Value of Explicit String = {explicitSting}");

        Console.WriteLine($"Value of Explicit Integer = {explicitInteger}");

        // Dynamic type for Comparison

        dynamic justcheckingString = "Dynamic Value!";

        Console.WriteLine($"Value of Dynamic String = {justcheckingString}");

    }

}

1. **Run**

* dotnet run

**🔧 Extras**

* **Difference with dynamic**:
  + var → type decided **at compile time**.
  + dynamic → type decided **at runtime** (can change, but risk of runtime errors).
* **Difference with object**:
  + object is the base type for all classes, but requires casting to use.
  + var avoids extra casting since compiler knows the real type.
* **Best practice:**  
  Use var when the type is clear or long; use explicit types when clarity matters. Most C# dev teams follow this guideline.

✅ That’s snippet 02 fully explained: you now understand how var works, why it’s not the same as JavaScript’s var, and when to use or avoid it.