C# 10 in a nutshell

Statement => executes sequentially and terminates by a semicolon

Method => performs a function

Class => groups function members and data members to form an object-oriented building block.

Using identifier that clashes with a reserved keyword => use @ => var @using which equals var using

Contextual keyword => can be used as an identifier

C# Types =>

* Value Types
* Reference Types
* Generic Type Parameters
* Pointer Types

Value Types => most built-in types => All numeric types => char and the bool Type

* The content of a value type variable is simply a value => e.g: int => 32 bits
* Define a custom value type => using Struct =>
* E.g :public struct point{ public int X; public int Y;}
* The assignment of a value type always copies the instance
* E.g :

Point p1 = new Point();

p1.X = 7;

Point p2 = p1; // Assignment causes copy

Console.WriteLine (p1.X); // 7

Console.WriteLine (p2.X); // 7

p1.X = 9; // Change p1.X

Console.WriteLine (p1.X); // 9

Console.WriteLine (p2.X); // 7

* P1 and p2 have independent storage

Reference Types => class, array, delegate and interface

* It has an object and a reference to that object
* The content of a reference type var or const => is a reference to an object that contains the value
* Assigning a reference type copies the reference not the object instance
* This allows multiple var to refer to the same object
* E.g :

Point p1 = new Point();

p1.X = 7;

Point p2 = p1; // Copies p1 reference

Console.WriteLine (p1.X); // 7

Console.WriteLine (p2.X); // 7

p1.X = 9; // Change p1.X

Console.WriteLine (p1.X); // 9

Console.WriteLine (p2.X); // 9

* A reference can be assign the literal null => reference points no object
* A value type cannot ordinarily have a null value

Null Operators =>

* Null-Coalescing Operator => ?? => if the operand to the left is non-null give it to me => otherwise give me another value

e.g:

string s1 = null;

string s2 = s1 ?? "nothing"; // s2 evaluates to "nothing"