What's new in C# 7.2

C# 7.2 is another point release that adds a number of useful features. One theme for this release is working more efficiently with value types by avoiding unnecessary copies or allocations.

The remaining features are small, nice-to-have features.

C# 7.2 uses the language version selection configuration element to select the compiler language version.

The new language features in this release are:

- Reference semantics with value types
  - A combination of syntax improvements that enable working with value types using reference semantics.
- Non-trailing named arguments
  - Named arguments can be followed by positional arguments.
- Leading underscores in numeric literals
  - Numeric literals can now have leading underscores before any printed digits.
- `private protected` access modifier
  - The `private protected` access modifier enables access for derived classes in the same assembly.

Reference semantics with value types

Language features introduced in 7.2 let you work with value types while using reference semantics. They are designed to increase performance by minimizing copying value types without incurring the memory allocations associated with using reference types. The features include:

- The `in` modifier on parameters, to specify that an argument is passed by reference but not modified by the called method.
- The `ref readonly` modifier on method returns, to indicate that a method returns its value by reference but doesn't allow writes to that object.
- The `readonly struct` declaration, to indicate that a struct is immutable and should be passed as an `in` parameter to its member methods.

- The `ref struct` declaration, to indicate that a struct type accesses managed memory directly and must always be stack allocated.

You can read more about all these changes in Using value types with reference semantics.

Non-trailing named arguments

Method calls may now use named arguments that precede positional arguments when those named arguments are in the correct positions. For more information see Named and optional arguments.

Leading underscores in numeric literals

The implementation of support for digit separators in C# 7.0 didn't allow the `_` to be the first character of the literal value. Hex and binary numeric literals may now begin with an `_`.

For example:

```
int binaryValue = 0b_0101_0101;
```

private protected access modifier

Finally, a new compound access modifier: `private protected` indicates that a member may be accessed by containing class or derived classes that are declared in the same assembly. While `protected internal` allows access by derived classes or classes that are in the same assembly, `private protected` limits access to derived types declared in the same assembly.