**C# 6.0 Language Features – Declaration Expressions**

Welcome to C# 6.0 language features article series. As you might have heard about Visual Studio 2015 and .Net framework 4.6 announcements in various Microsoft events earlier, one of the most exciting news was the next version of C# language, which is 6.0 and new features added to it. C# has come a long way since 2002 when C# 1.0 was released along with Visual Studio. Since then with every major release of .Net framework, C# became better and better language, which has tremendously helped application developers to minimize the code complexity and focus on business functionality. With the new version of .Net framework and Visual Studio around the corner, you can start learning new language features today using Visual Studio 2015 preview release. There are many useful features added to the language and I don’t know at the moment, how many articles I will write in this series, but it will probably something like this.

* Static using statements
* Auto property initializer
* Null conditional operator
* String interpolation
* nameof operator
* **Declaration expressions - This Article**

**Declaration Expressions**

In most of the programming languages including C#, we always had declaration statements to declare variable and expressions to produce values. C# 6.0 introduces a new syntax to mix both declaration and expression; that means you can declare new variable and produce a value using an expression at the same time. Let’s understand it with few code examples –

The int.TryParse method requires an out parameter which returns the output of the expressions. Before C# 6.0, we had to declare the out variable first, before we can use it in an expression

int result;

if(int.TryParse(input, out result))

{

    return result;

}

return 0;

With C# 6.0, same result can be achieved using single statement as shown below

if (int.TryParse(input, out var result))

{

    return result;

}

return 0;

You can also use declaration expressions to declare variable inside an if statement

if ((var firstName = user.FirstName) != null)

{

    return user;

}

…and in lambda expression

int result = 0;

foreach (var n in var odd = numbers.Where(n => n % 2 == 1).ToList())

{

    result += n + odd.Count();

}

return result;

Although it’s a small feature, it can be handy in few scenarios.