

The C# programming language provides support for extension methods from C# 3.0. An extension method is one that is used to extend the functionality of existing types by adding methods sans the need of creating new derived types. You don't need to create subclasses of existing classes or recompile or modify your existing classes to work with extension methods. Extension methods improve the readability of your code while at the same time allowing you to extend functionality of existing types.

The common extension methods in .Net include the LINQ standard query operators that adds additional query capabilities to the System.Collections.IEnumerable and System.Collections.Generic.IEnumerable<T> types. Note that you can take advantage of extension methods to extend a class or an interface but you cannot override their methods. [The MSDN states](https://msdn.microsoft.com/en-IN/library/bb383977.aspx): "Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type. Extension methods are a special kind of static method, but they are called as if they were instance methods on the extended type."

Essentially, an extension method is a special type of a static method and enable you to add functionality to an existing type even if you don't have access to the source code of the type. An extension method is just like another static method but has the “this” reference as its first parameter. You can add as many extension methods as you want to any type. Most importantly, you can also add extension methods to even a value type.

When working with extension methods, keep these points in mind:

* An extension method must be a static method
* An extension method must be inside a static class -- the class can have any name
* The parameter in an extension method should always have the "this" keyword preceding the type on which the method needs to be called

Note that it you define an extension method on a type that has the same signature as any other method of the type that you are extending, the extension method will never be called.

**Programming extension methods in C#**

In this section we will explore how to program extension methods using C#. The following code listing illustrates how an extension method looks like.

public static class StringExtensions

**[ Discover how to be a great scrum master in your agile development projects:**[**The scrum master role, explained**](https://www.infoworld.com/article/3236445/agile-development/what-is-a-scrum-master-the-agile-development-leader-defined.html)**• Take your agile career to the next level:**[**How to improve your scrum master skills**](https://www.infoworld.com/article/3236547/agile-development/how-to-improve-your-scrum-master-skills.html)**. • Not sure what “agile” really means? InfoWorld**[**explains the agile methodology**](https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html)**. | Keep up with hot topics in programming with InfoWorld’s**[**App Dev Report newsletter**](https://www.infoworld.com/newsletters/signup.html)**. ]**

    {

        public static bool IsNumeric(this string str)

        {

            double output;

            return double.TryParse(str, out output);

        }

    }

Note the first parameter to the extension method. As already discussed, any extension method should be static and should have "this" keyword preceding the parameter you want the method to be called on. When you specify the "this" keyword in the parameter list the way it has been shown in the above example, you inform the compiler that an extension method has been defined for the string class.

Here's how you can use the extension method IsNumeric on a string instance.

static void Main(string[] args)

        {

            string str = "100";

            if (str.IsNumeric())

                Console.WriteLine("The string object named str contains numeric value.");

            Console.Read();

        }

When you execute the above program, the message ("The string object named str contains numeric value." Is displayed in the console window.

You can use extension methods to inject new functionality via methods to a type sans the need of modifying, deriving or recompiling the original type. As I mentioned earlier, extension methods can also be applied to value types. Let's see how this can be achieved with an example.

The following class named IntegerExtensions contains an extension method named IsEven that returns true if the integer on which it is called is even, false otherwise.

public static class IntegerExtensions

    {

        public static bool IsEven(this int i)

        {

            return ((i % 2) == 0);

        }

    }

And, here's how you can use the extension method IsEven on an integer.

int n = 2;

if(n.IsEven())

 Console.WriteLine("The value of the integer is even.");