Anonymous Functions (C# Programming Guide)

Visual Studio 2013 7 out of 10 rated this helpful

An anonymous function is an "inline" statement or expression that can be used wherever a delegate type is expected. You can use it to initialize a named delegate or pass it instead of a named delegate type as a method parameter.

There are two kinds of anonymous functions, which are discussed individually in the following topics:

- Lambda Expressions (C# Programming Guide) .
- Anonymous Methods (C# Programming Guide)

Note

Lambda expressions can be bound to expression trees and also to delegates.

The Evolution of Delegates in C#

In C# 1.0, you created an instance of a delegate by explicitly initializing it with a method that was defined elsewhere in the code. C# 2.0 introduced the concept of anonymous methods as a way to write unnamed inline statement blocks that can be executed in a delegate invocation. C# 3.0 introduced lambda expressions, which are similar in concept to anonymous methods but more expressive and concise. These two features are known collectively as *anonymous functions*. In general, applications that target version 3.5 and later of the .NET Framework should use lambda expressions.

The following example demonstrates the evolution of delegate creation from C# 1.0 to C# 3.0:

C#

```
class Test
{
    delegate void TestDelegate(string s);
    static void M(string s)
    {
        Console.WriteLine(s);
    }

    static void Main(string[] args)
    {
        // Original delegate syntax required
        // initialization with a named method.
        TestDelegate testDelA = new TestDelegate(M);

        // C# 2.0: A delegate can be initialized with
        // inline code, called an "anonymous method." This
        // method takes a string as an input parameter.
        TestDelegate testDelB = delegate(string s) { Console.WriteLine(s); };

        // C# 3.0. A delegate can be initialized with
        // a lambda expression. The lambda also takes a string
        // as an input parameter (x). The type of x is inferred by the compiler.
        TestDelegate testDelC = (x) => { Console.WriteLine(x); };

        // Invoke the delegates.
        testDelA("Hello. My name is M and I write lines.");
        testDelB("That's nothing. I'm anonymous and ");
        testDelC("I'm a famous author.");

        // Keep console window open in debug mode.
```

```
            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }
    }
    /* Output:
        Hello. My name is M and I write lines.
        That's nothing. I'm anonymous and
        I'm a famous author.
        Press any key to exit.
     */
```

C# Language Specification

For more information, see the C# Language Specification. The language specification is the definitive source for C# syntax and usage.

See Also

Reference
Statements, Expressions, and Operators (C# Programming Guide)
Lambda Expressions (C# Programming Guide)
Delegates (C# Programming Guide)
Concepts
Expression Trees (C# and Visual Basic)