**Anonymous Types (C# Programming Guide)**

Updated: July 2008

Anonymous types provide a convenient way to encapsulate a set of read-only properties into a single object without having to first explicitly define a type. The type name is generated by the compiler and is not available at the source code level. The type of the properties is inferred by the compiler. The following example shows an anonymous type being initialized with two properties called Amount and Message.

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl01other');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl01other');)

var v = new { Amount = 108, Message = "Hello" };

Anonymous types are typically used in the [select](http://msdn.microsoft.com/en-us/library/bb384087.aspx) clause of a query expression to return a subset of the properties from each object in the source sequence. For more information about queries, see [LINQ Query Expressions (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/bb397676.aspx).

Anonymous types are created by using the [new](http://msdn.microsoft.com/en-us/library/51y09td4.aspx) operator with an object initializer. For more information about object initializers, see [Object and Collection Initializers (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/bb384062.aspx).

Anonymous types are [class](http://msdn.microsoft.com/en-us/library/0b0thckt.aspx) types that consist of one or more public read-only properties. No other kinds of class members such as methods or events are allowed. An anonymous type cannot be cast to any interface or type except for [object](http://msdn.microsoft.com/en-us/library/9kkx3h3c.aspx).

The most common scenario is to initialize an anonymous type with some properties from another type. In the following example, assume a class that is named Product that includes Color and Price properties together with several other properties that you are not interested in. Products is a collection of Product objects. The anonymous type declaration starts with the **new** keyword. It initializes a new type that uses only two properties from Product. This causes a smaller amount of data to be returned in the query.

If you do not specify member names in the anonymous type, the compiler gives the anonymous type members the same name as the property being used to initialize them. You must provide a name to a property that is being initialized with an expression.

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl08CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl08CSharp');)

var productQuery =

from prod in products

select new { prod.Color, prod.Price };

foreach (var v in productQuery)

{

Console.WriteLine("Color={0}, Price={1}", v.Color, v.Price);

}

When an anonymous type is assigned to a variable, that variable must be initialized with the [var](http://msdn.microsoft.com/en-us/library/bb383973.aspx) construct. This is because only the compiler has access to the underlying name of the anonymous type. For more information about **var**, see [Implicitly Typed Local Variables (C# Programming Guide)](http://msdn.microsoft.com/en-us/library/bb384061.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Remarks

Anonymous types are reference types that derive directly from [object](http://msdn.microsoft.com/en-us/library/9kkx3h3c.aspx). The compiler gives them a name although your application cannot access it. From the perspective of the common language runtime, an anonymous type is no different from any other reference type, except that it cannot be cast to any type except for [object](http://msdn.microsoft.com/en-us/library/9kkx3h3c.aspx).

If two or more anonymous types have the same number and type of properties in the same order, the compiler treats them as the same type and they share the same compiler-generated type information.

An anonymous type has method scope. To pass an anonymous type, or a collection that contains anonymous types, outside a method boundary, you must first cast the type to [object](http://msdn.microsoft.com/en-us/library/9kkx3h3c.aspx). However, this defeats the strong typing of the anonymous type. If you must store your query results or pass them outside the method boundary, consider using an ordinary named struct or class instead of an anonymous type.

Anonymous types cannot contain [unsafe](http://msdn.microsoft.com/en-us/library/chfa2zb8.aspx) types as properties.

Because the [Equals](http://msdn.microsoft.com/en-us/library/system.object.equals.aspx) and [GetHashCode](http://msdn.microsoft.com/en-us/library/system.object.gethashcode.aspx) methods on anonymous types are defined in terms of the **Equals** and **GetHashcode** of the properties, two instances of the same anonymous type are equal only if all their properties are equal.