匿名类型提供了一种方便的方法，可用来将一组只读属性封装到单个对象中，而无需首先显式定义一个类型。类型名由编译器生成，并且不能在源代码级使用。每个属性的类型由编译器推断。

可通过使用 [new](https://msdn.microsoft.com/zh-CN/library/51y09td4.aspx) 运算符和对象初始值创建匿名类型。有关对象初始值设定项的详细信息，请参阅[对象和集合初始值设定项（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb384062.aspx)。

以下示例显示了用两个名为 Amount 和 Message 的属性进行初始化的匿名类型。

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

// Rest the mouse pointer over v.Amount and v.Message in the following

// statement to verify that their inferred types are int and string.

Console.WriteLine(v.Amount + v.Message);

匿名类型通常用在查询表达式的 [select](https://msdn.microsoft.com/zh-CN/library/bb384087.aspx) 子句中，以便返回源序列中每个对象的属性子集。有关查询的详细信息，请参阅[LINQ 查询表达式（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb397676.aspx)。

匿名类型包含一个或多个公共只读属性。包含其他种类的类成员（如方法或事件）为无效。用来初始化属性的表达式不能为 **null**、匿名函数或指针类型。

最常见的方案是用其他类型的属性初始化匿名类型。在下面的示例中，假定名为 Product 的类存在。类 Product 包括 Color 和 Price 属性，以及你不感兴趣的其他属性。变量 products 是 Product 对象的集合。匿名类型声明以 **new** 关键字开始。声明初始化了一个只使用 Product 的两个属性的新类型。这将导致在查询中返回较少数量的数据。

如果你没有在匿名类型中指定成员名称，编译器会为匿名类型成员指定与用于初始化这些成员的属性相同的名称。必须为使用表达式初始化的属性提供名称，如下面的示例所示。在下面示例中，匿名类型的属性名称都为 Color 和 Price。

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);

}

通常，当你使用匿名类型来初始化变量时，可以通过使用 [var](https://msdn.microsoft.com/zh-CN/library/bb383973.aspx) 将变量作为隐式键入的本地变来变量进行声明。类型名称无法在变量声明中给出，因为只有编译器能访问匿名类型的基础名称。有关 **var**的详细信息，请参阅[隐式类型的局部变量（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb384061.aspx)。

可通过将隐式键入的本地变量与隐式键入的数组相结合创建匿名键入的元素的数组，如下面的示例所示。

var anonArray = new[] { new { name = "apple", diam = 4 }, new { name = "grape", diam = 1 }};

**备注**

匿名类型是直接从[对象](https://msdn.microsoft.com/zh-CN/library/9kkx3h3c.aspx)派生的[类](https://msdn.microsoft.com/zh-CN/library/0b0thckt.aspx)类型，并且其无法强制转换为除[对象](https://msdn.microsoft.com/zh-CN/library/9kkx3h3c.aspx)外的任意类型。虽然你的应用程序不能访问它，编译器还是提供了每一个匿名类型的名称。从公共语言运行时的角度来看，匿名类型与任何其他引用类型没有什么不同。

如果程序集中的两个或多个匿名对象初始值指定了属性序列，这些属性采用相同顺序且具有相同的名称和类型，则编译器将对象视为相同类型的实例。它们共享同一编译器生成的类型信息。

无法将字段、属性、时间或方法的返回类型声明为具有匿名类型。同样，你不能将方法、属性、构造函数或索引器的形参声明为具有匿名类型。要将匿名类型或包含匿名类型的集合作为参数传递给某一方法，可将参数作为类型对象进行声明。但是，这样做会使强类型化作用无效。如果必须存储查询结果或者必须将查询结果传递到方法边界外部，请考虑使用普通的命名结构或类而不是匿名类型。

由于匿名类型上的 [Equals](https://msdn.microsoft.com/zh-CN/library/bsc2ak47.aspx) 和 [GetHashCode](https://msdn.microsoft.com/zh-CN/library/system.object.gethashcode.aspx) 方法是根据方法属性的 **Equals** 和 **GetHashCode** 定义的，因此仅当同一匿名类型的两个实例的所有属性都相等时，这两个实例才相等。

**请参阅**

[C# 编程指南](https://msdn.microsoft.com/zh-CN/library/67ef8sbd.aspx)

[对象和集合初始值设定项（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb384062.aspx)

[Getting Started with LINQ in C#](https://msdn.microsoft.com/zh-CN/library/bb397933.aspx)

[LINQ 查询表达式（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb397676.aspx)