# What is new in Visual Studio 2010 C#.

## Type dynamic

The type is a static type, but an object of type dynamic bypasses static type checking. In most cases, it functions like it has type object. At compile time, an element that is typed as dynamic is assumed to support any operation. Therefore, you do not have to be concerned about whether the object gets its value from, either COM, DOM.... However, if the code is not valid, errors are caught at run time.

Any object can be converted to dynamic type implicitly;Conversely, an implicit conversion can be dynamically applied to any expression of type dynamic.

The dynamic language runtime (DLR) is a new API in .NET Framework 4. It provides the infrastructure that supports the dynamic type in C#, and also the implementation of dynamic programming languages such as IronPython and IronRuby.

If you compile by using the [/link (C# Compiler Options)](http://msdn.microsoft.com/en-us/library/dd264728%28VS.100%29.aspx) option, the introduction of the dynamic type enables you to treat the occurrences of object in COM signatures as if they were of type dynamic, and thereby to avoid much of the casting.

### Office Programmability

Access to COM interfaces, including the Office Automation APIs, is greatly enhanced by the addition of named and optional arguments, the dynamic type, indexed properties and optional ref modifiers.

## Named Argument

Named arguments free you from the need to remember or to look up the order of parameters in the parameter lists of called methods.

CalculateBMI(weight: 123, height: 64); yet not CalculateBMI(weight: 123, 64);

## Optional Argument

Any call must provide arguments for all required parameters, but can omit arguments for optional parameters.

### Type Equivalence Support

Embedded type information instead of type information that is imported from a Primary Interop Assembly (PIA). With embedded type information, your application can use types in a runtime without requiring a reference to the runtime assembly. If various versions of the runtime assembly are published, the application that contains the embedded type information can work with the various versions without having to be recompiled. Declare the ID of the interface and the interface can be loaded at runtime without recompiling.

#### Embeded Types from Managed Assemblies

If you embed type information from a strong-named managed assembly, you can loosely couple types in an application to achieve version independence. That is, your program can be written to use types from multiple versions of a managed library without having to be recompiled for each version.