**C#**

The **dynamic** type enables the operations in which it occurs to bypass compile-time type checking. Instead, these operations are resolved at run time. The **dynamic** type simplifies access to COM APIs such as the Office Automation APIs, and also to dynamic APIs such as IronPython libraries, and to the HTML Document Object Model (DOM).

Type **dynamic** behaves like type **object** in most circumstances. However, operations that contain expressions of type **dynamic** are not resolved or type checked by the compiler. The compiler packages together information about the operation, and that information is later used to evaluate the operation at run time. As part of the process, variables of type **dynamic** are compiled into variables of type **object**. Therefore, type **dynamic** exists only at compile time, not at run time.

/\*

Dynamic Dispatch

C# 4.0 supports dynamic late-binding.

With the dynamic keyword, instead of calling a method "Square"

on some object using reflection in this manner,

you can now tell the compiler to treat "Runtime" as dynamic and delay all analysis until run time.

\*/

using System;

using System.Reflection;

class Maths

{

public int Square(int Integer)

{

return Integer \* Integer;

}

}

class Program

{

static void Method()

{

object Instance = new Maths();

Type InstanceType = Instance.GetType();

object Result = InstanceType.InvokeMember("Square", BindingFlags.InvokeMethod, null, Instance, new object[] { 5 });

int Number = Convert.ToInt32(Result);

Console.WriteLine(Number);

dynamic Runtime = new Maths();

Number = Runtime.Square(5);

Console.WriteLine(Number);

}

static void MethodFromAssembly()

{

Assembly AssemblyName = Assembly.LoadFrom("Sample.dll");

Type AssemblyTypeName = AssemblyName.GetType("Maths");

object Instance = Activator.CreateInstance(AssemblyTypeName);

object Result = AssemblyTypeName.InvokeMember("Square", BindingFlags.InvokeMethod, null, Instance, new object[] { 5 });

int Number = Convert.ToInt32(Result);

Console.WriteLine(Number);

dynamic Runtime = Activator.CreateInstance(AssemblyTypeName);

Number = Runtime.Square(5);

Console.WriteLine(Number);

}

static void Main()

{

Method();

MethodFromAssembly();

Console.ReadKey();

}

}