Anonymous

- Anonymous type is a type or class without any name that can contain public read-only properties. It cannot contain other members, such as fields, methods, events, etc.
- ‘new’ operator is used to create an anonymous type.
- The properties of anonymous types cannot be initialized with a null, anonymous function, or a pointer type.
- The properties can be accessed using dot (.) notation.
- Example: **var student = new{ Id = 1, Name = “Richy”, Salary = 88000};**
- An anonymous type property can include another anonymous type.
- Anonymous type can also use to create an Array.
- Anonymous types are directly derived from the System.Object class.

Delegate

- Func is a built-in generic delegate type.
- Func can contains 0 to 16 input parameters and must have one return type.
- Func delegate used to create a delegate only in a single line without using the procedure to create the delegate
- C# provides a built-in delegate that is Func. Using Func delegate you need not follow the following procedure to create a delegate.
- Syntax:

  Public delegate TResult Func<in P1,in P2,out PResult>(P1 arg1,P2 arg2);
- Here, P1, P2 are the type of input parameters, PResult is the type of output parameter.

Action

- Action is a delegate type defined in the System namespace.
- It Encapsulates a method and does not return a value.
- An Action delegate can be initialized using the new keyword or by directly assigning a method
- An Action type delegate is the same as Func delegate except that the Action delegate doesn't return a value.
- Syntax : **public delegate void Action( );**
- Advantages of using Action delegates :

1. Easy and quick to define delegates.
2. Makes code short.
3. Compatible type throughout the application.

Predicate

- Predicate is the delegate like Func and Action delegates. A predicate delegate methods must take one input parameter and return a boolean (true or false).
- Anonymous method and Lambda expression can be assigned to the predicate delegate.
- Syntax :

  public delegate bool Predicate<in T>(T object);

Delegates

- Delegate is a reference to the method. It is objected-oriented, secured and type-safe.
- Delegate declaration defines a class which is the derived class of System.Delegate.
- Syntax :

  Public delegate void DelegateName( );

- Delegates can be used to define call-back methods.
- Delegates can be chained together, multiple methods can be called on a single event.
- You can declare a delegate that can appear on its own or even nested inside a class. There are three steps in using delegates.
    1. Declaring a delegate
    2. Instantiating a delegate
    3. Calling a delegate.