In .net 3.5 some new generic delegates - Func<T>, Action<T> and Predicate<T> were introduced. Using generic delegates, it is possible to concise delegate type means you don't have to define the delegate statement. These delegates are the Func<T>, Action<T> and Predicate<T> delegates and defined in the System namespace.

Action<T> performs an operation on the generic arguments. Func<T> performs an operation on the argument(s) and returns a value, and Predicate<T> is used to represent a set of criteria and determine if the argument matches the criteria.

Here "in" shows the input parameters and "out" shows the return value by the delegate.

**Generic delegate example**

```csharp
using System;

class demo
{
delegate void MyDelegate(string str);
static void Main(string[] args)
{
MyDelegate d = show;
d("Hello World!");
Console.ReadLine();
}
static void show(string str)
```
Above code can be written as using generic delegate.

```csharp
using System;
class demo {
    static void Main(string[] args) {
        Action<string> d = show;
        d("Hello World!");
        Console.ReadLine();
    }
    static void show(string str) {
        Console.WriteLine(str);
    }
}
```

### Generic delegate using anonymous method

```csharp
using System;
class demo {
    static void Main(string[] args) {
        Action<string> d = s => Console.WriteLine(s);
        d("Hello World!");
    }
}
```