Introduction

In this section, you'll get a subtle introduction on delegates and you'll know why they're used.

WE'LL COVER THE FOLLOWING

A **delegate object** encapsulates any *reference method* or *methods* that can then be called upon just by calling the delegate object. This way, the users won't know explicitly which methods are being invoked at compile time.

> **A delegate is a way of telling C# which method to call when an event is triggered.**

For example, if you click a Button on a form, the program would call a specific method. It is this pointer that is a delegate. Delegates are good, as you can notify several methods that an event has occurred if you wish so.

Call a Function without using Delegates

We normally call a function simply by instantiating the class and directly calling the required function through it:

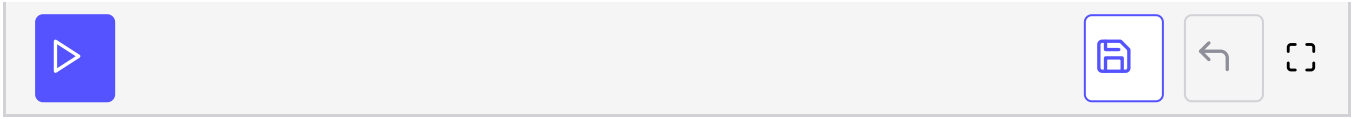
```
using System;

public class DemoClass
{
    public void Method1()
    {
        Console.WriteLine("Method 1");
    }
}

class DemoMain
{
    static void Main()
```

```
    {
        DemoClass myClass = new DemoClass();
        myClass.Method1();

    }
}
```



This method works fine, but sometimes we want some other tool to carry out this job and invoke the necessary functions for a certain task without us knowing.

For example, in the `GUI` applications, upon clicking an option or a button, we want the backend functions to execute without the public knowing. This is called **“event handling”**.

> Delegates are important while performing event handling tasks.

Let us now properly begin “Delegates” next!