Chapter 7

https://csharp.christiannagel.com

Delegates, Lambdas, and Events

Delegates

```
// declare a delegate
delegate void IntMethodInvoker(int x);
```

- Type-safe pointer to one or more methods
- A delegate is a class / instantiate an object to reference a method

```
delegate string GetAString();
int x = 40;
GetAString firstStringMethod = new GetAString(x.ToString);
Console.WriteLine($"String is {firstStringMethod()}");
```

Passing Delegates to Methods

- Contracts
- Compare with Interface Contracts
- Array.Sort with a delegate

```
public static class MathOperations
{
  public static double MultiplyByTwo(double value) => value * 2;
  public static double Square(double value) => value * value;
}
```

```
delegate double DoubleOp(double x);
```

Action<T> and Func<T>

Action

- Delegate to methods with void return
- and T1 .. Tn parameters

Func

- Delegate with T return
- and T1 .. Tn parameters

Multicast-Delegate

- A delegate can reference multiple methods
- += and -= Operators

Lambda Expression

- Use lambda operator =>
- In-place implementation of a method
- Closures

```
Func<double, double> square = x => x * x;
```

```
Func<double, double> square = x =>
{
  return x * x;
};
```

```
int someVal = 5;
Func<int, int> f = x => x + someVal;
```

Events

- Base on delegates
- Publish/subscribe mechanism

```
private EventHandler<CarInfoEventArgs>? _newCarCreated;
public event EventHandler<CarInfoEventArgs>? NewCarCreated
{
  add => _newCarCreated += value;
  remove => _newCarCreated -= value;
}
```

```
private EventHandler<CarInfoEventArgs>? NewCarCreated;
```

Summary

- Delegates
- Lambda Expressions
- Events