Observables

"Call me if you find out anything."

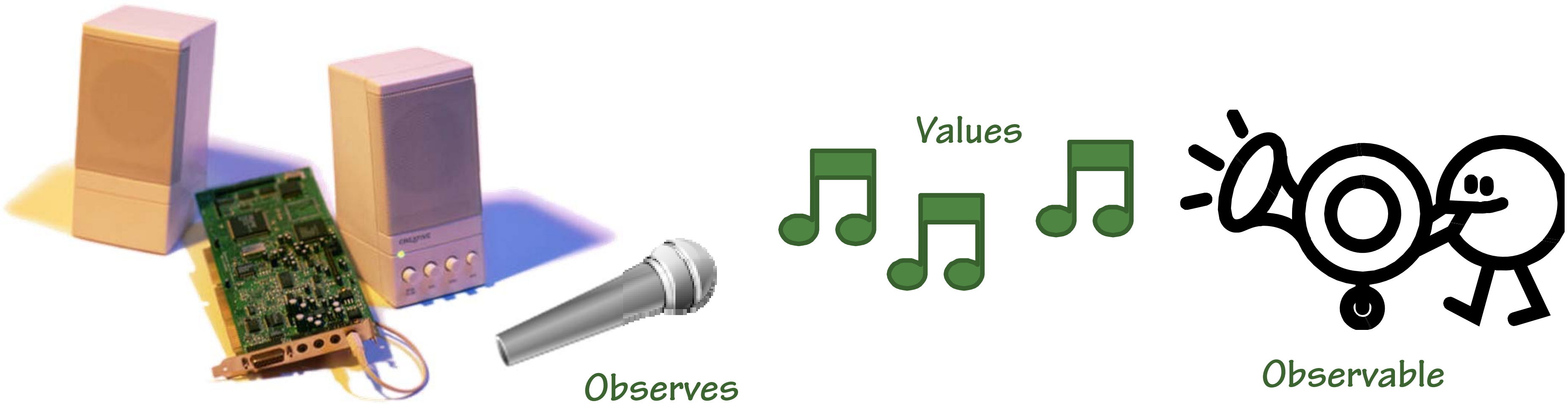
Overview

- Observable
- Observations ≡ Callbacks
- Generating Callbacks

```
var ChunckOfValues = "12, 4, 8, 23, 15";
foreach(var IndividualValue in ChunckOfValues.Split(
 new char[] {',', ' '},
 StringSplitOptions.RemoveEmptyEntries))
{
 Console.WriteLine(int.Parse(IndividualValue));
}
```

Overview

- Observable
- Observations ≡ Callbacks
- Generating Callbacks



Values

Processes

Observes

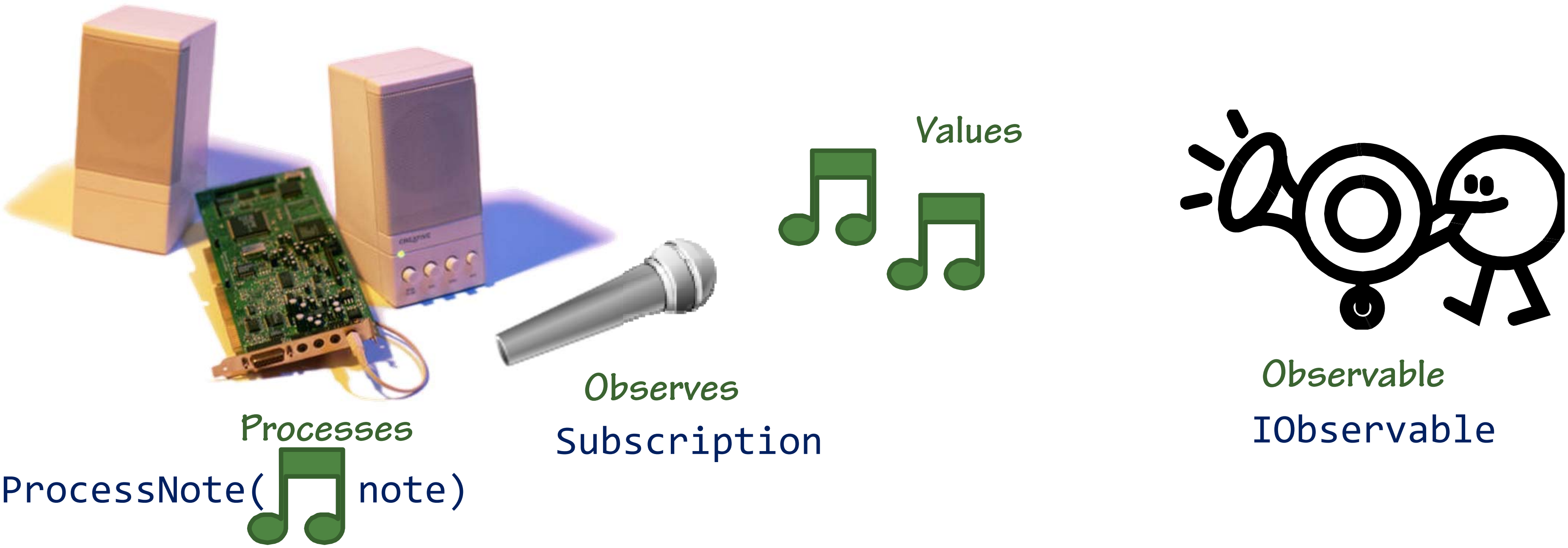
Subscription

Observable

IObservable

Overview

- Observable
- Observations ≡ Callbacks
- Generating Callbacks



Values

Observable

IObservable

Observes

Subscription

Processes

ProcessNote(♫ note)

Observable Sequence

- Subscribe
- Grammar

IObservable

creates observable

```
var observable = (new int[] {1,2,3}).ToObservable
```

Observable Sequence

- Subscribe
- Grammar

starts callbacks

```
observable.Subscribe(Console.WriteLine)
```

```
IDisposable Subscribe(IObserver observer)
{
   foreach(var i in new int[] {1,2,3})
   {
      observer.OnNext(i);
   }
   return this;
}
```

Observable Sequence

- Subscribe
- Grammar

```
observable.Subscribe(Console.WriteLine)

               IObservable
    IDisposable Subscribe(IObserver observer)
    {
       foreach(var i in new int[] {1,2,3})
       {
          observer.OnNext(i);
       }
       return this;
    }
```

Observable Sequence

- Subscribe
- Grammar

```
observable.Subscribe(Console.WriteLine)

    IDisposable Subscribe(IObserver observer)
    {
       foreach(var i in new int[] {1,2,3})
       {
          observer.OnNext(i);
       }
       return this;
    }
```

Observable Sequence

- Subscribe
- Grammar

```
observable.Subscribe(Console.WriteLine)
```

```
IDisposable Subscribe(IObserver observer)
{
   foreach(var i in new int[] {1,2,3})
   {
      observer.OnNext(i);
   }
   return this;
}
```

Observable Sequence

- Subscribe
- Grammar
  - OnNext OnError OnComplete

```
observable.Subscribe(Console.WriteLine, HandleError, Done)

    IDisposable Subscribe(IObserver observer)
    {
       foreach(var i in new int[] {1,2,3})
       {
          observer.OnNext(i);
       }
       return this;
    }
```

Observable Sequence

- **Subscribe**
- **Grammar**
  - OnNext OnError OnComplete

```
observable.Subscribe(Console.WriteLine, HandleError, Done)
```

```
OnNext 1;
OnNext 2;
OnNext 3;
```

Observable Sequence

- **Subscribe**
- **Grammar**
  - OnNext OnError OnComplete

```
observable.Subscribe(Console.WriteLine, HandleError, Done)
```

OnNext* (OnError | OnComplete)?

Observable Sequence

- Subscribe
- Grammar
  - OnNext OnError OnComplete

```
observable.Subscribe(Console.WriteLine, HandleError, Done)
```

OnNext* (OnError | OnComplete)?

Create

- **Implement IObserver.Subscribe**
  - Ad hoc observable sequence
  - No dependence on LINQ query or IEnumerable
- **Observable.Create Observable.CreateWithDisposable**

```
IEnumerable<int> MyAdHocSequence()
{
        yield return 1;
        yield return 2;
}

var observable = MyAdHocSequence().ToObservable();
```

Run

- **Runs subscription**
  - blocks until complete

Start

- **Subscribes to observable**
  - returns list
  - no OnNext, OnError, or OnComplete
- **Invokes a delegate**
  - IObservable<Type>
  - IObservable<System.Unit>

```
Process.Start
```

Start

- **Subscribes to observable**
  - returns list
  - no OnNext, OnError, or OnComplete
- **Invokes a delegate**
  - IObservable<Type>
  - IObservable<System.Unit>

```
Thread.Start
```

Start

- Subscribes to observable
  - returns list
  - no OnNext, OnError, or OnComplete
- Invokes a delegate
  - IObservable<Type>
  - IObservable<System.Unit>

digit 0 1 2 3 4 5 6 7 8 9

Start

- **Subscribes to observable**
  - returns list
  - no OnNext, OnError, or OnComplete
- **Invokes a delegate**
  - IObservable<Type>
  - IObservable<System.Unit>

digit 🖊🖍✂☎☟☝🗑🖱☽◇

Start

- Subscribes to observable
  - returns list
  - no OnNext, OnError, or OnComplete
- Invokes a delegate
  - IObservable<Type>
  - IObservable<System.Unit>

bit ✂🗑

Start

- **Subscribes to observable**
  - returns list
  - no OnNext, OnError, or OnComplete
- **Invokes a delegate**
  - IObservable<Type>
  - IObservable<System.Unit>

unit ✂

Summary

- ObservableExtensions.Subscribe creates IObservable.Subscribe
- OnNext* (OnError | On Complete)
- Create for ad hoc Subscribe
- Run to wait for completion
- Start
  - ListObservable
  - delegate

```
IObservable.Subscribe(IObserver observer)
{
   observer.OnNext(<value>);
   observer.OnNext(<value>);
   observer.OnCompleted();
}
```

References

- **Redgate Reflector**
  - http://www.reflector.net/