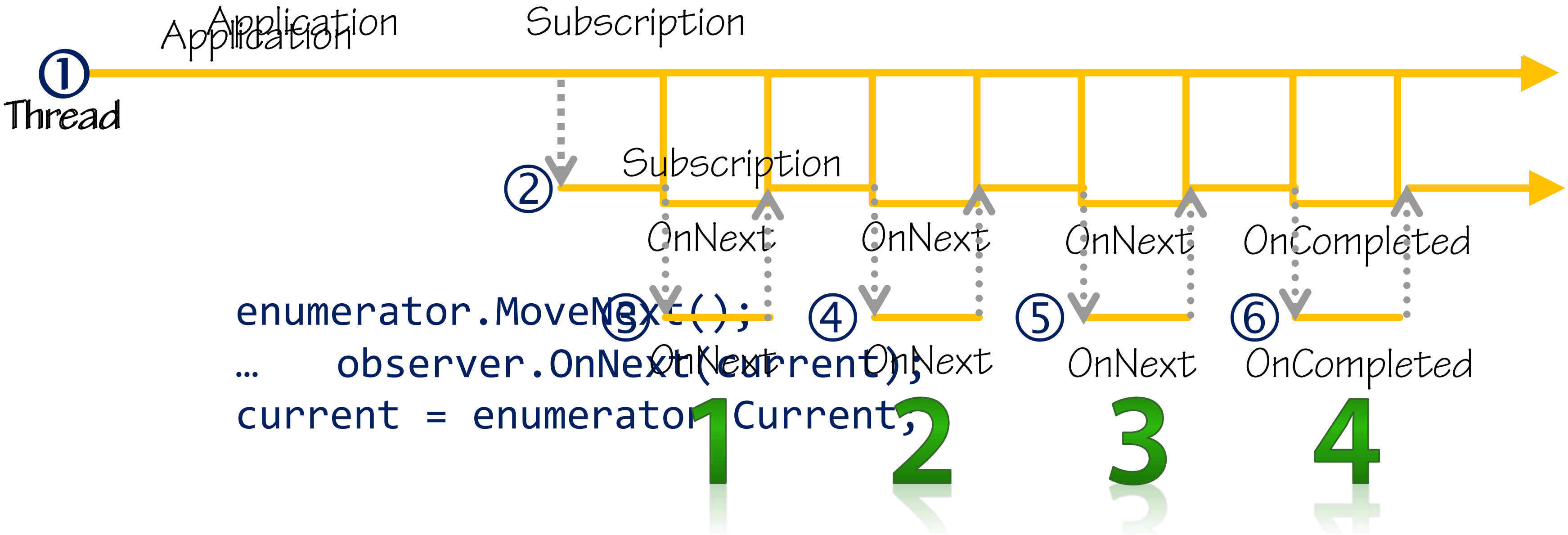
Managing Concurrency

The left hand ***does*** *know what the right is doing.*

Overview

- **Scheduler**
- **ToObservable, SubscribeOn ObserveOn**



```
enumerator.MoveNext();
…    observer.OnNext(current);
current = enumerator.Current;
```

Scheduling Delegates

- ToObservable
- SubscribeOn ObserveOn

subscription

```
enumerator.MoveNext()
…
current = enumerator.Current;
```

```
ToObservable(Scheduler.NewThread)
```

```
SubscribeOn(Scheduler.NewThread).ObserveOn(Scheduler.Dispatcher)
```

```
Scheduler.NewThread.Schedule(MyDelegate)
```

```
NewThread
ThreadPool
TaskPool
CurrentThread
Immediate
Dispatcher
```

part of subscription

```
from number in Enumerable.Range(1,3) select number;
```

```
from number in Enumerable.Range(1,3) select Process(number)
```

part of observation

```
observer.OnNext(current);
```

```
observer.OnNext(Process(current))
```

pluralsight
see what you can learn

Specific Scheduling

- ObserveOn
- SubscribeOn

Observable Lifetime

- Subscriptions cleanup automatically
- Subscriptions are disposable

```
var subscription= observableSequence.Subscribe(Console.WriteLine);
//…
subscription.Dispose();
```

Cleanup

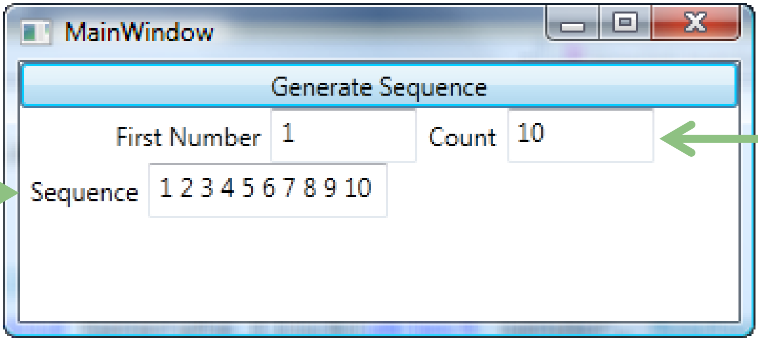
- Using
- Finally

Scheduler.Dispatcher

- **GUI applications**
  - WPF, Forms

observation

application thread



MainWindow

First Number 1

Count 10

Sequence 1 2 3 4 5 6 7 8 9 10

subscription

background thread

Summary

- **Declarative**
- **Subscription / Observation**
  - SubcribeOn / ObserveOn
- **Observation Grammer / Concurrency**
  - (OnNext*)?, (OnError | OnCompleted)?
- **Subscription termination**
- **Disposable objects**
- **Cleanup**
- **WPF and Windows Forms applications**