Reactive Extensions for .NET

(Rx)

Orion Edwards, December 2009

If we wish to count lines of code, we should not regard them as lines produced but as lines spent. – *Edsger Dijkstra*

C# 3.0 Recap

- Delegates and Lambdas
- IEnumerable<T>
- Extension Methods
- Linq

C# 3 Recap

```
public bool Filter(string name) {
  return name.EndsWith("n");
}

foreach(var name in listOfNames) {
  if(Filter(name))
    resultList.Add(name);
}
```

C# 3 Recap

```
Func<string, bool> filter =
    name => name.EndsWith("n");

foreach(var name in listOfNames) {
  if(filter(name))
    resultList.Add(name);
}
```

C# 3 Recap

```
public IEnumerable<int> GetSequence() {
  return new int[]{ 1, 2, 3, 4 };
}

public IEnumerable<int> GetSequence() {
  return new List<int>{ 1, 2, 3, 4 };
}

public IEnumerable<int> GetSequence() {
  for(int i = 0; i < 1000; ++i)
    yield return i;
}
```

C# 3 Recap

```
public static int TimesTwo(this int i) {
  return i * 2;
}

var x = 27.TimesTwo();
```

C# 3 Recap

```
var result = listOfNames
    .Where(name => name.EndsWith("n"))
    .ToList();
```

```
public static IEnumerable<T> Where<T>(
    this IEnumerable<T> source, Func<T, bool> predicate)
    {
      foreach(var item in source)
        if(predicate(item))
          yield return item;
    }
```

C# 3 Recap

```
var result = (from name in listOfNames
              where name.EndsWith("n")
              select name).ToList();
```

```
public static IEnumerable<T> Where<T>(
   this IEnumerable<T> source, Func<T, bool> predicate)
   {
     foreach(var item in source)
       if(predicate(item))
         yield return item;
   }
```

START WITH BREVITY. Increase the other dimensions of code AS REQUIRED BY TESTING. – *Wil Shipley*

What is Rx?

Rx is a library for composing asynchronous and event-based programs using observable collections.

Rx is a superset of the standard LINQ sequence operators that exposes asynchronous and event-based computations as push-based, observable collections via the new .NET 4.0 interfaces IObservable<T> and IObserver<T>. These are the mathematical dual of the familiar IEnumerable<T> and IEnumerator<T> interfaces for pull-based, enumerable collections in the .NET framework.

What is Rx?

What is Rx?

- .NET Reactive Framework == Reactive Extensions == Rx
- It's a set of libraries (4 on .NET 3.5)
- A unified way for expressing asynchronous operations
- A lot of very powerful extension methods and utilities to help work with asynchronous operations
- Makes asynchronous code more awesome

Erik Meijer

- Architect at Microsoft
- Invented LINQ
- Considerably smarter than me
- Rx is his latest project

What is asynchronous programming?

- We want to do something
- But the data isn't available yet...
- Wait for the data to arrive
- But still do other things while waiting

Asynchronous Examples

Slow operations

```
ThreadPool.QueueUserWorkItem(_ => {
  var files = Directory.GetFiles(
      @"\\live.sysinternals.com\tools")
  Display(files);
});
```

Asynchronous Examples

Background worker

```
worker = new BackgroundWorker();
worker.DoWork += new DoWorkEventHandler(worker_DoWork);
worker.ProgressChanged +=
  new ProgressChangedEventHandler(worker_ProgressChanged);
worker.RunWorkerCompleted +=
  new RunWorkerCompletedEventHandler(worker_RunWorkerCompleted);

worker.WorkerReportsProgress = true;
worker.RunWorkerAsync();
```

Asynchronous Examples

Begin/End functions

```
BeginProcessing(data, (asyncResult, state) => {
  var data = EndProcessing(asyncResult);
  MessageBox.Show(data);
}, null);
```

Asynchronous Examples

Events

```
this.KeyDown += (sender, e) => {
    m_keyIsDown = true; Process(e); }

this.KeyUp += (sender, e) => {
    m_keyIsDown = false; Process(e); }
```

Why bother with asynchronous programming at all?

It *is* easier just to block and wait...

Because we have to

- Event-loop based applications
- Interaction
- Parallelism
- Basically if we ever want more than one thing at once, we MUST do things asynchronously

Linq is awesome, and Rx gives us asynchronous Linq

Programmers can write roughly the same number of lines of code per year regardless of language.
– Barry Boehm - Software Engineering Economics

```
interface IEnumerable<T> {
  IEnumerator<T> GetEnumerator();
}
```

```
interface IEnumerator<T> {
  T Current;        Can also throw an exception
  bool MoveNext();  Signals when complete
  void Dispose();   Used to cancel
}
```

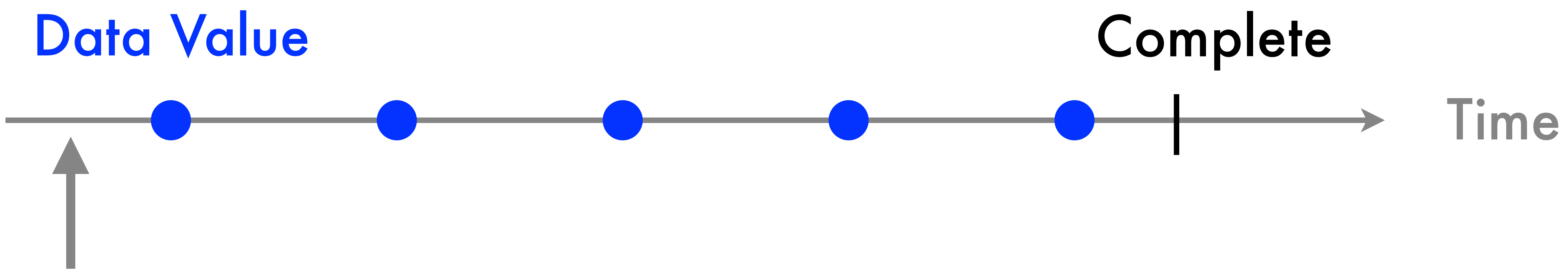
Unified model for “Pull” sequences

```
interface IObservable<T> {
  IDisposable Subscribe(IObserver<T>);
}
```

```
interface IObserver<T> {
  OnNext(T);
  OnError(Exception);
  OnComplete();
}
```

Unified model for “Push” sequences

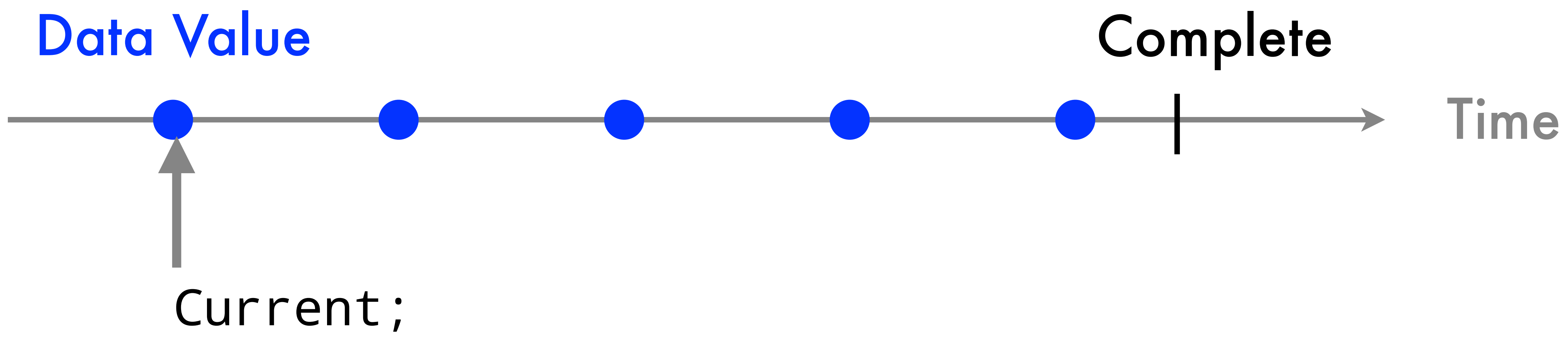
Pull with IEnumerable



```
MoveNext();
```

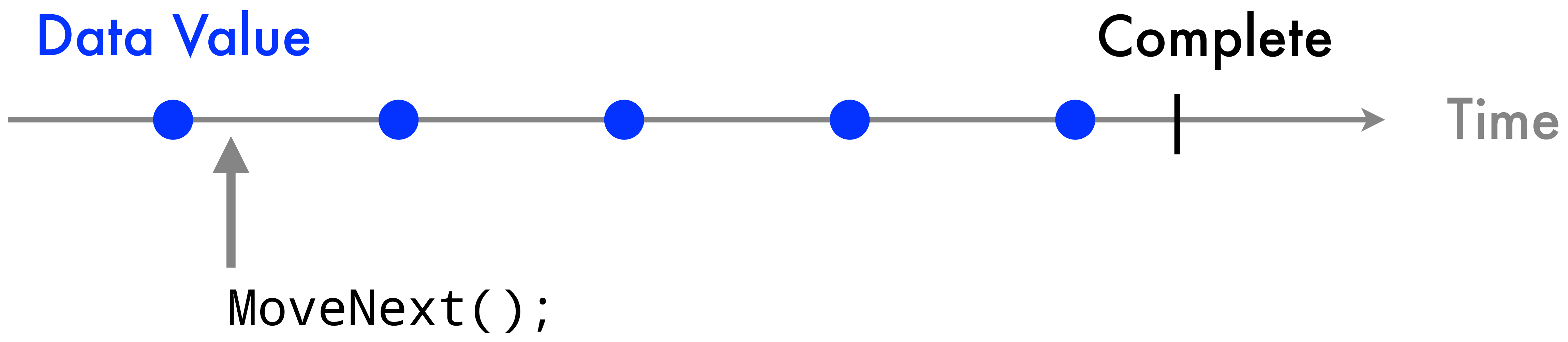
No data yet; will block and wait until some arrives

Pull with IEnumerable



Data has arrived, retrieve it by calling Current

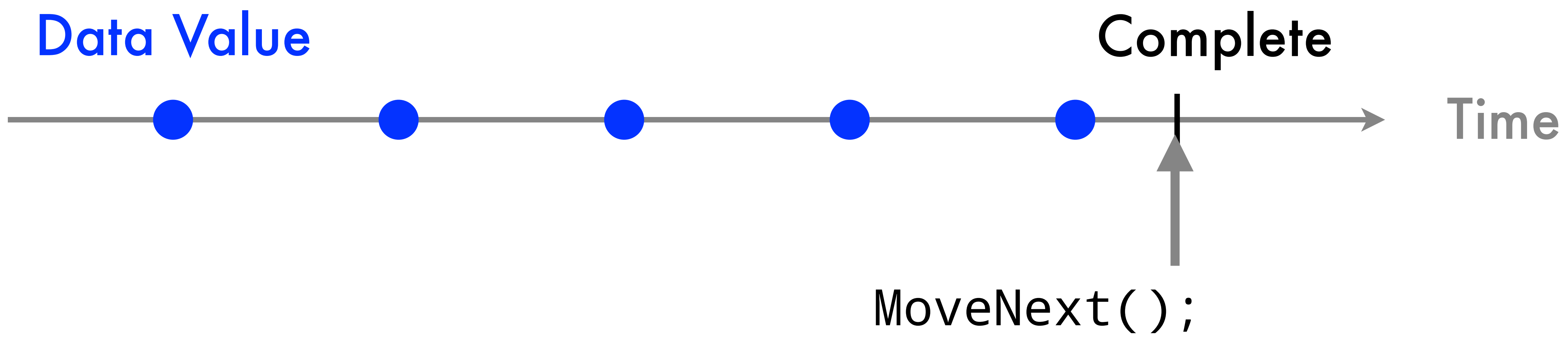
Pull with IEnumerable



```
MoveNext();
```

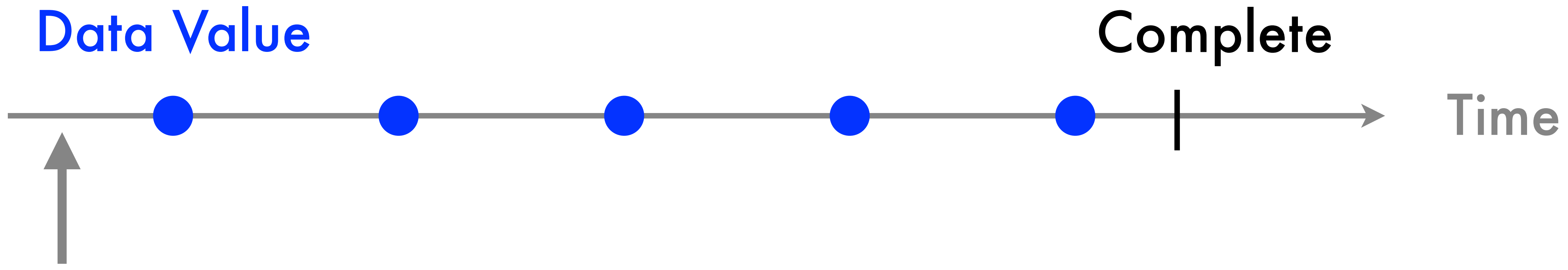
No data yet; will block and wait until the next item

Pull with IEnumerable



No more data, returns false

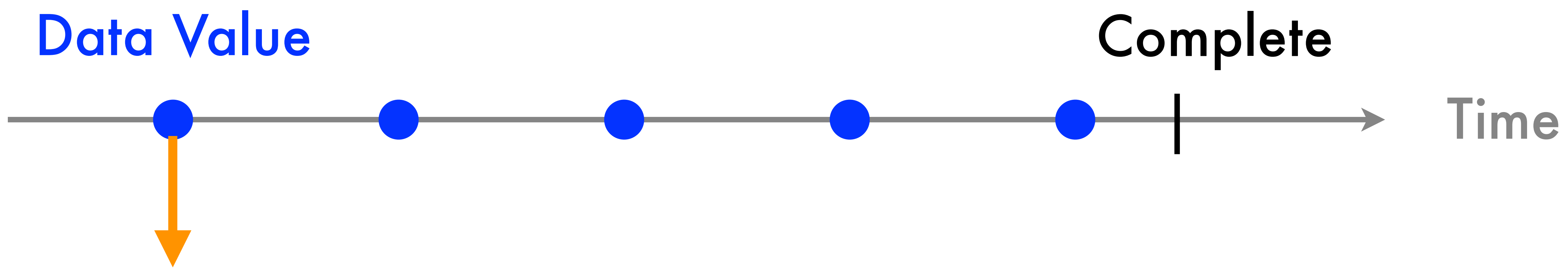
Push with IObservable



```
Subscribe(IObserver);
```

Starts the asynchronous sequence. Thread is now free to do other stuff

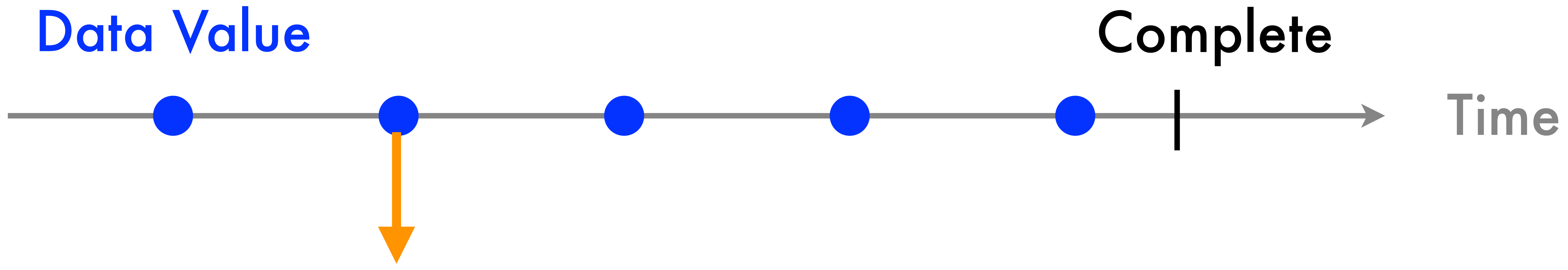
Push with IObservable



```
OnNext(value);
```

Value is posted to the observer (callback function)

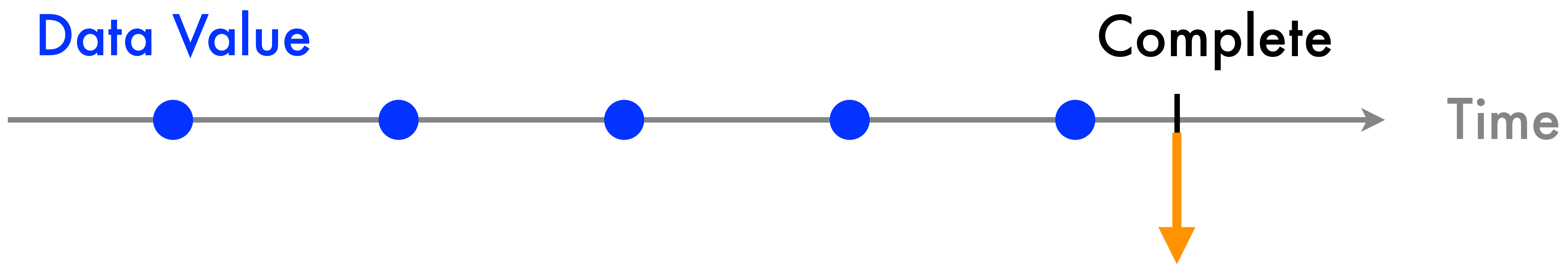
Push with IObservable



```
OnNext(value);
```

Value is posted to the observer (callback function)

Push with IObservable



```
OnComplete();
```

Completion signal is posted to the observer

Example

```
IObservable<string> SlowStrings(); // pretend this exists
```

```
SlowStrings().Subscribe(
  value => Console.WriteLine(value),
  error => MessageBox.Show(error.Message),
  () => MessageBox.Show("Done!"));
```

Demo

3 kinds

- Hot - always running
- Cold - only start when you subscribe
- Single - start when you subscribe and end as soon as a value arrives

Help!

- Rx provides observable.ToEnumerable()
  This runs any observable synchronously!
- Also has .First(), .Single(), etc for one-shot observables
- Has enumerable.ToObservable() too!
  Dependency injection and mock objects are just for people who don’t know maths - Erik Meijer

...and...

- Provides observable versions of every Linq extension method
  Use reflector to poke around!
- *Lots* of utilities to help creating your own observables / wrap existing code
- Many other things that *should* have been in Linq
  So they back-ported to work on IEnumerable as well!

...one more thing

- Rx uses the .NET 4 Parallel tasks library
- So they back-ported it to .NET 3.5!

Summary: Why use Rx?

- Unified way of handling asynchronous operations
- Methods for many common tasks
- Composable
- Concise

The relationship between lines of code and bugs is completely linear. Fewer code means fewer bugs.
– *Jeff Atwood*

Thanks!

Official Site

Google for “reactive framework”

Blog - has lots of helpful videos

blogs.msdn.com/RxTeam

Matt Podwysocki’s Introduction to Rx

Google for “codebetter rx part 1”