Creating Reactive Applications in .NET

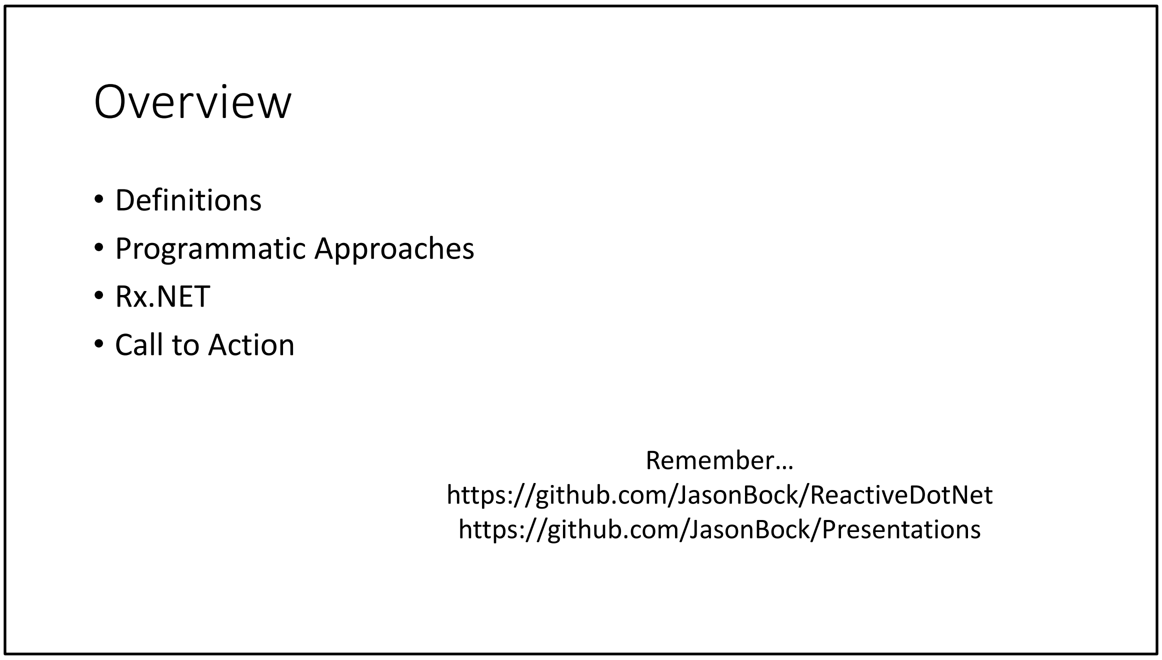
Jason Bock
Staff Software Engineer
Rocket Mortgage

Personal Info

- http://jasonbock.net
- https://mstdn.social/@jasonbock
- https://bsky.app/profile/jasonbock.bsky.social
- https://github.com/jasonbock
- https://youtube.com/JasonBock
- https://www.linkedin.com/in/jasonrbock/
- jason.r.bock@outlook.com

Downloads

https://github.com/JasonBock/ReactiveDotNet
https://github.com/JasonBock/Presentations



Overview

- Definitions
- Programmatic Approaches
- Rx.NET
- Call to Action

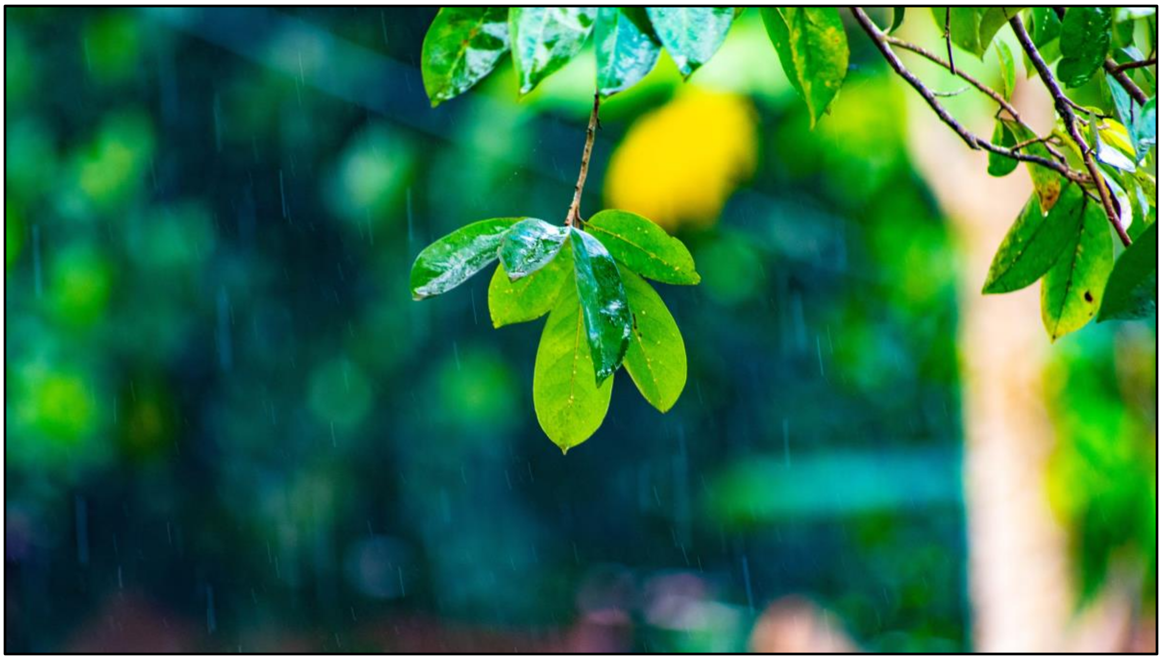
Remember…
https://github.com/JasonBock/ReactiveDotNet
https://github.com/JasonBock/Presentations

Definitions

Creating Reactive Applications in .NET

The universe is reactive. We are constantly reacting to events, whether they're on the universal scale of supernova or merging black holes, or local events on earth.

https://webbtelescope.org/contents/media/images/2023/127/01H1Q1CGJD51CDJTK2NHJWD06M

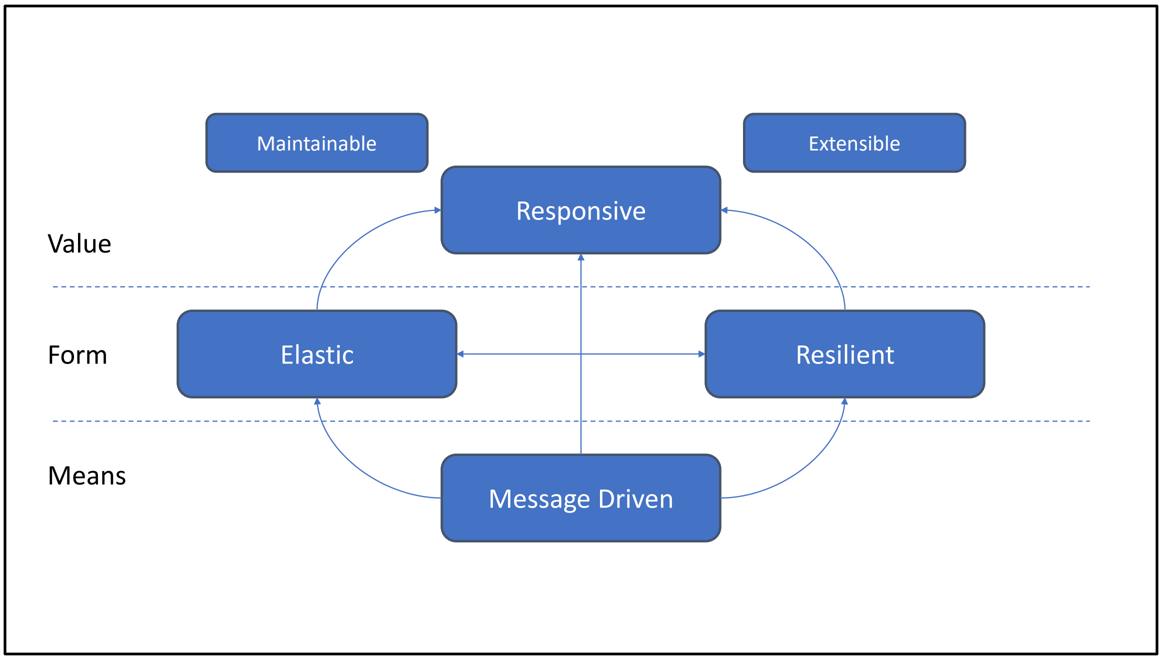


Maybe it starts raining out. How do you react? If you were planning on doing yard work, that may change your plans. Or you decide to sit by a window and relax to the sound.

https://unsplash.com/photos/green-leaves-in-tilt-shift-lens-CeihbxSUNxw

Or if you're driving in bad conditions. You may start to slide, or a car loses control coming the other way. You need to react, fast.

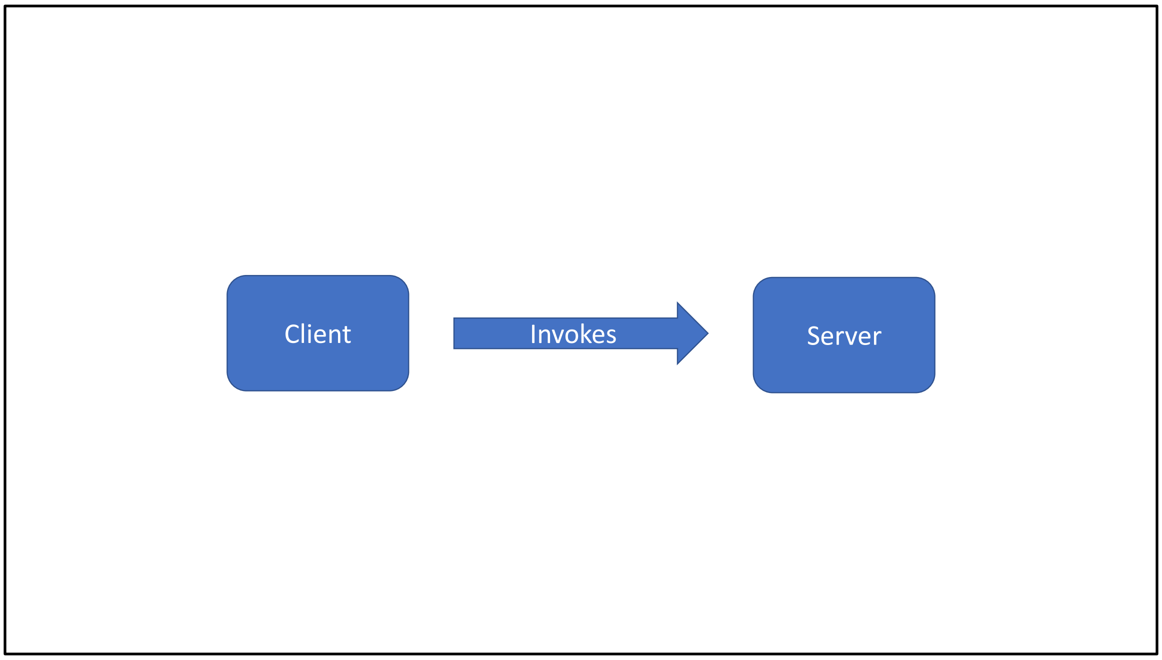
https://www.pexels.com/photo/red-sedan-in-the-middle-of-forest-376361/



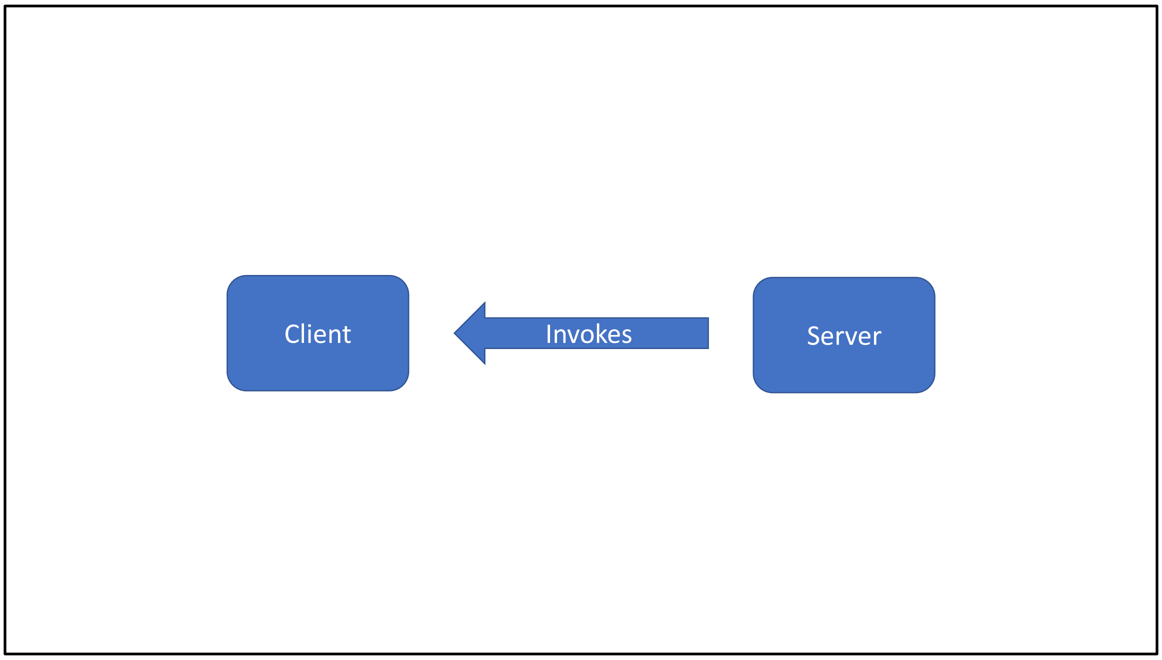
The reactive manifesto defines 4 characteristics of reactive systems:

- Responsive - Responsive systems focus on providing rapid end consistent response times, establishing reliable upper bounds so they deliver a consistent quality of service
- Resilient - The system stays responsive in the face of failure. This applies not only to highly-available, mission critical systems — any system that is not resilient will be unresponsive after a failure. Resilience is achieved by replication, containment, isolation and delegation
- Elastic - Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs
- Message Driven - Reactive Systems rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation and location transparency

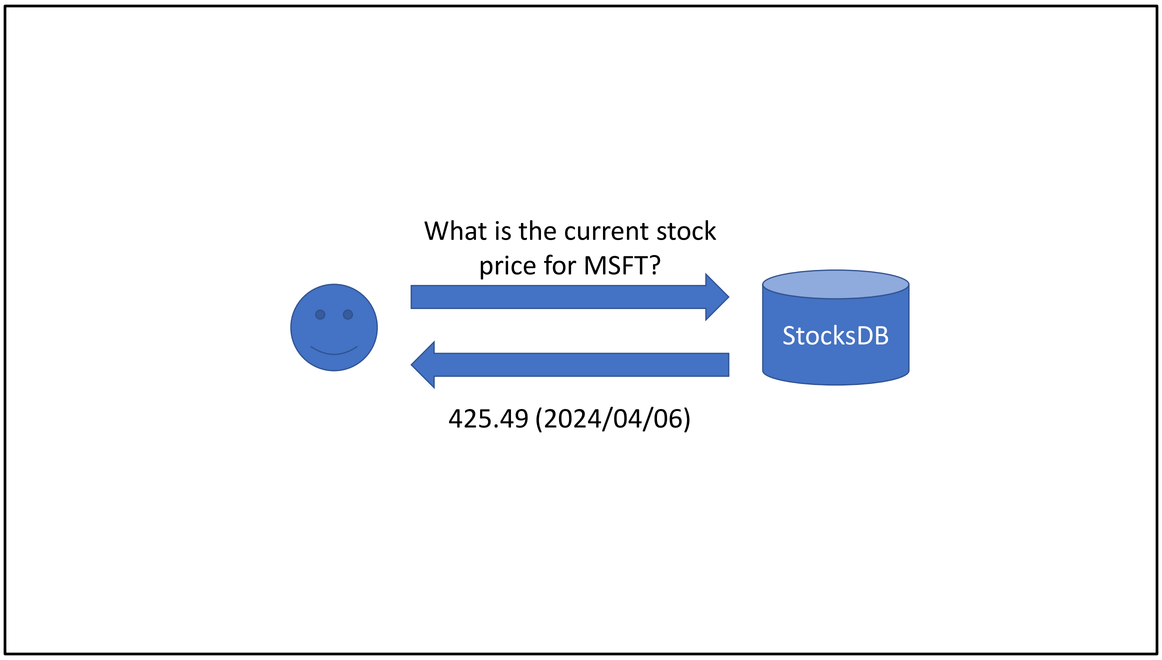
http://www.reactivemanifesto.org/



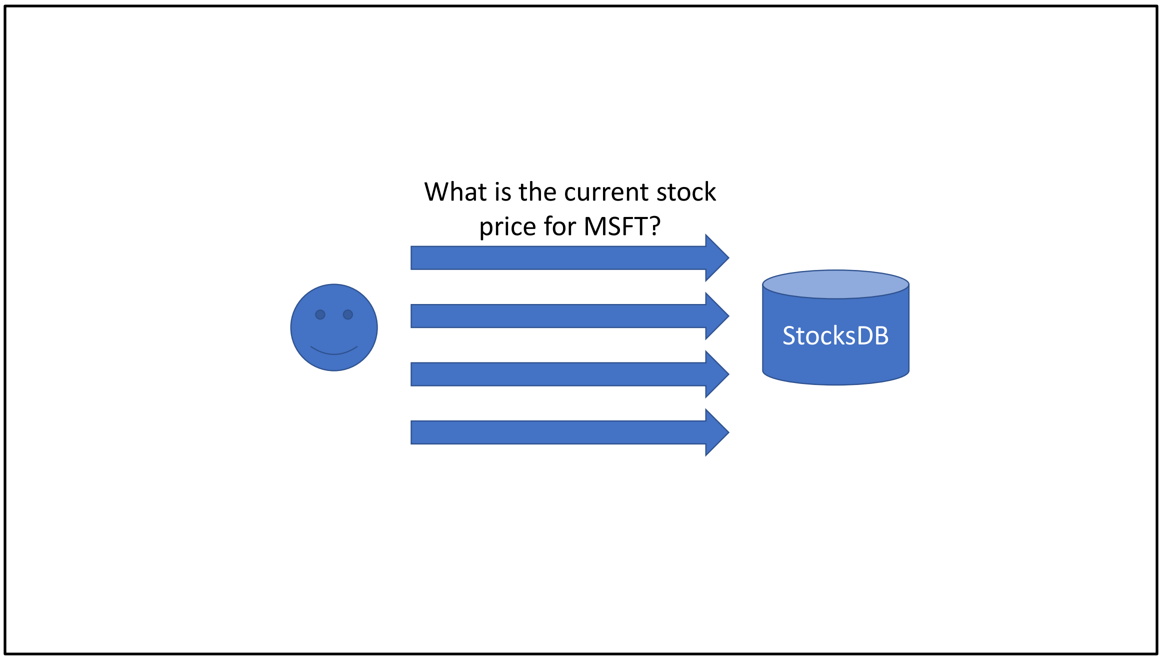
How do we typically do communications between clients and servers? Easy, the client makes a call to the server. This is usually done with REST these days, but it can happen in other ways. The point is, the client initializes the call, and will either “wait” for a response (typically asynchronous), or fire and forget.



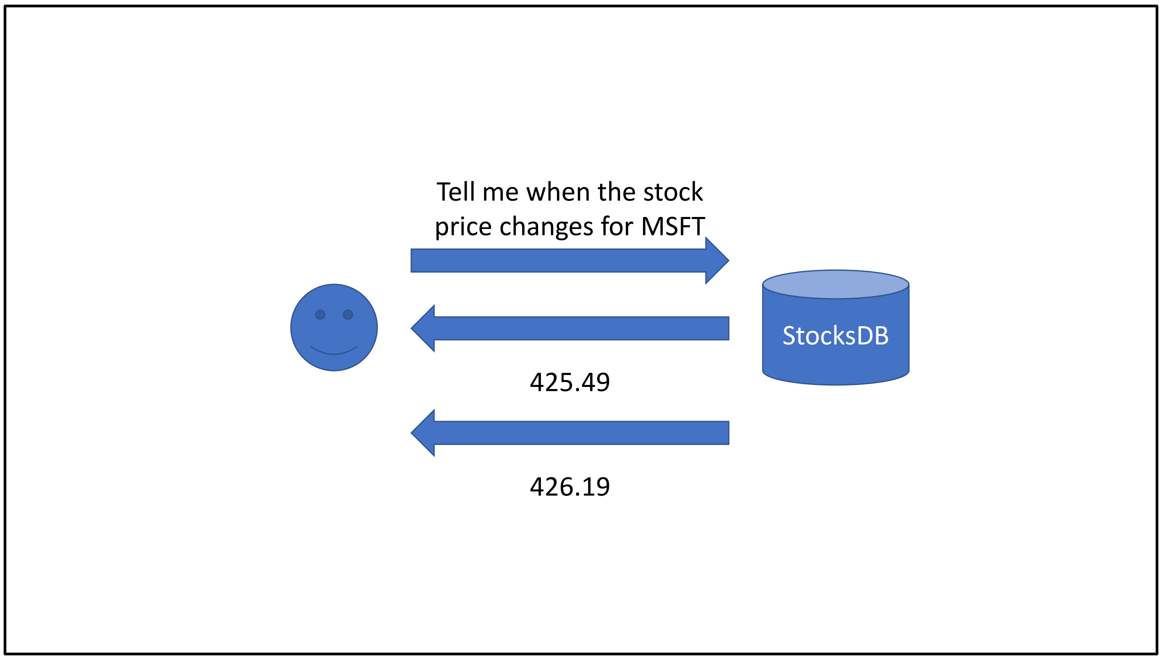
But what if we reverse the arrow? Maybe we want our service to notify the client when specific actions have occurred even though the client didn't initialize the request.



Let’s talk about stock tickers. We can ask for the price of a stock at a particular point in time.



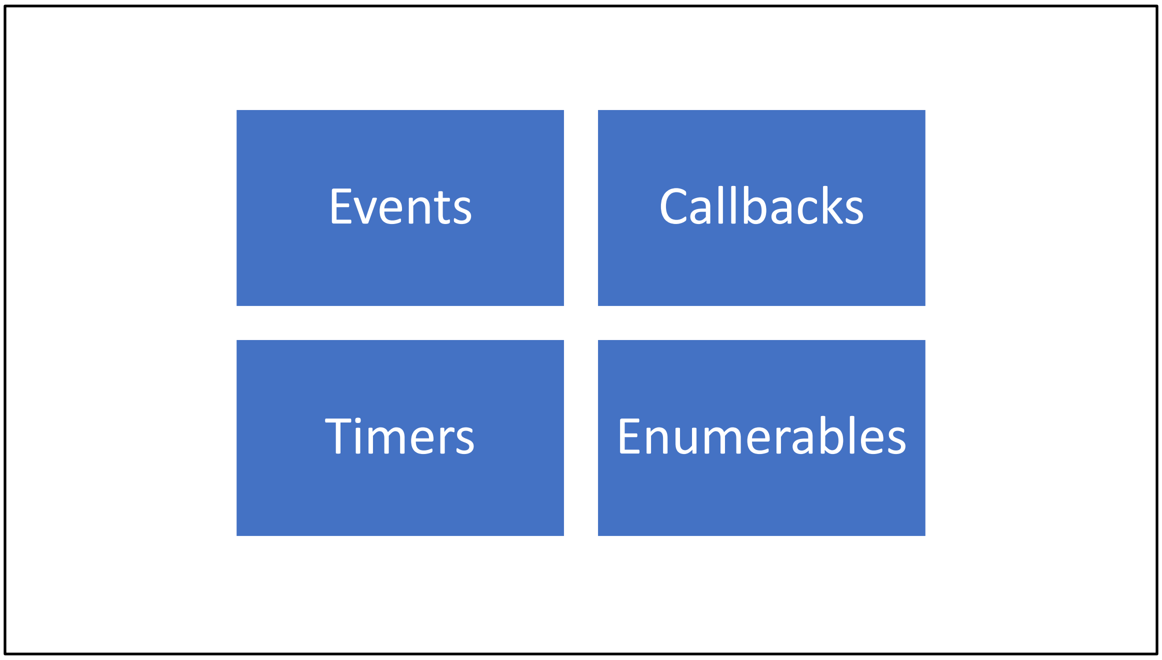
Doing this over and over can work, but we can miss changes, and this may unnecessarily cause network resource usage. You lose either way no matter what your interval strategy is.



Instead, let’s react to the changes in price.

Programmatic Approaches

Creating Reactive Applications in .NET



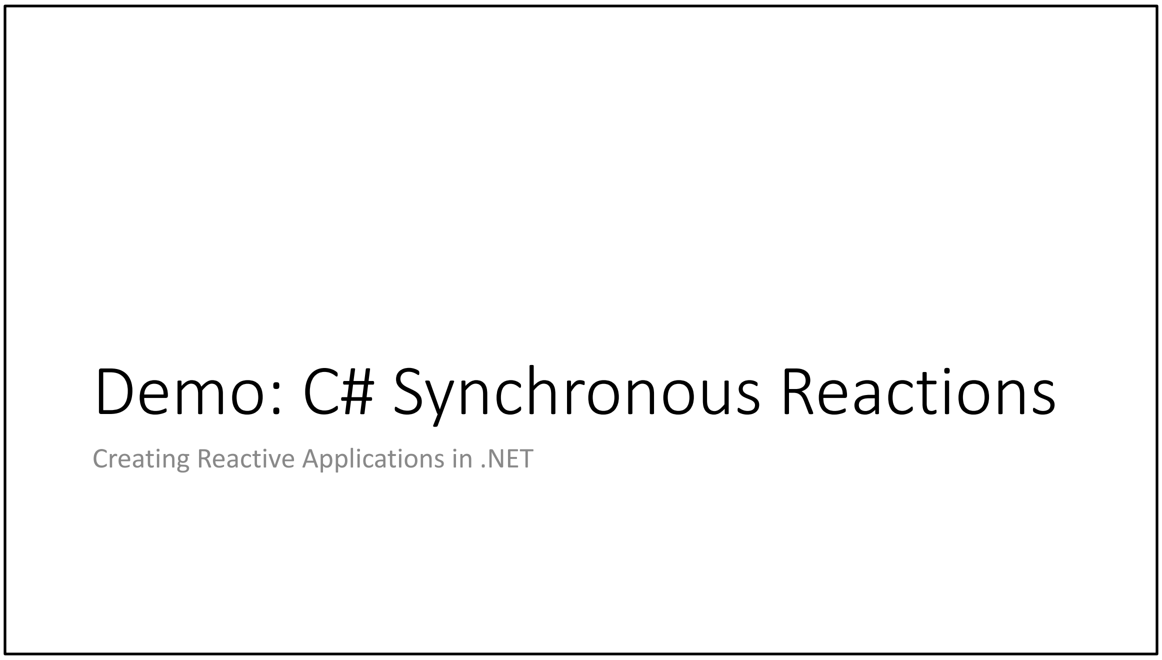
Events

Callbacks

Timers

Enumerables

There are a number of synchronous ways we can react to changes in C#



Demo: C# Synchronous Reactions

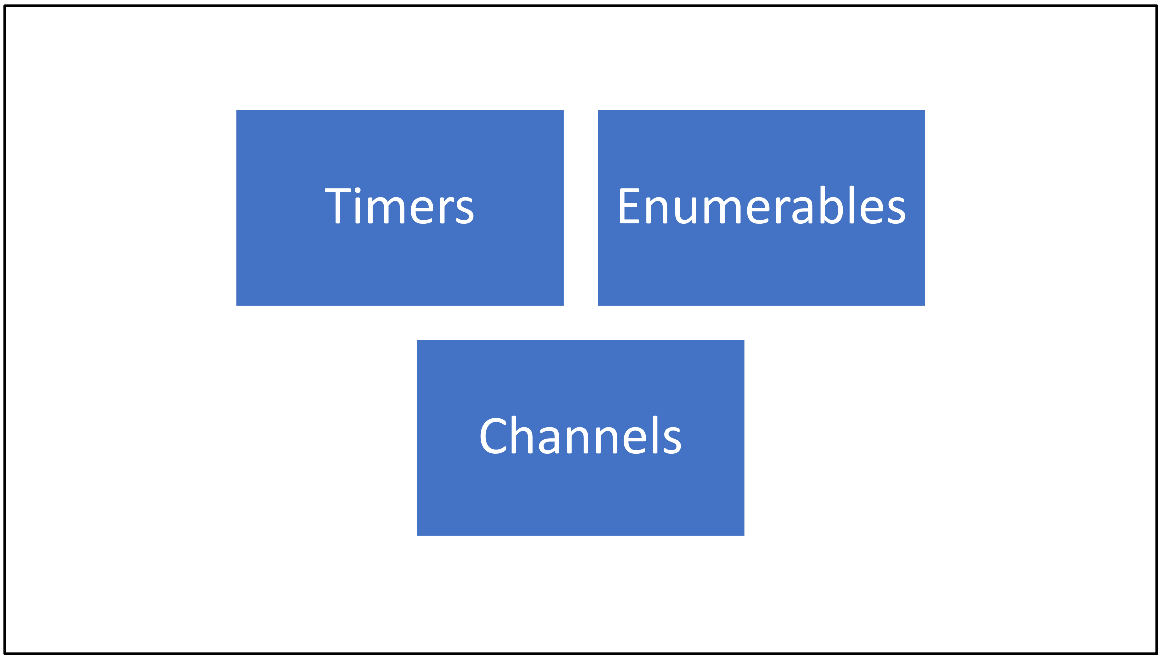
Creating Reactive Applications in .NET

Handling events in a synchronous fashion isn’t desirable. In the real world, do you wait by the washing machine until it’s done, and then wait until the dryer is done?

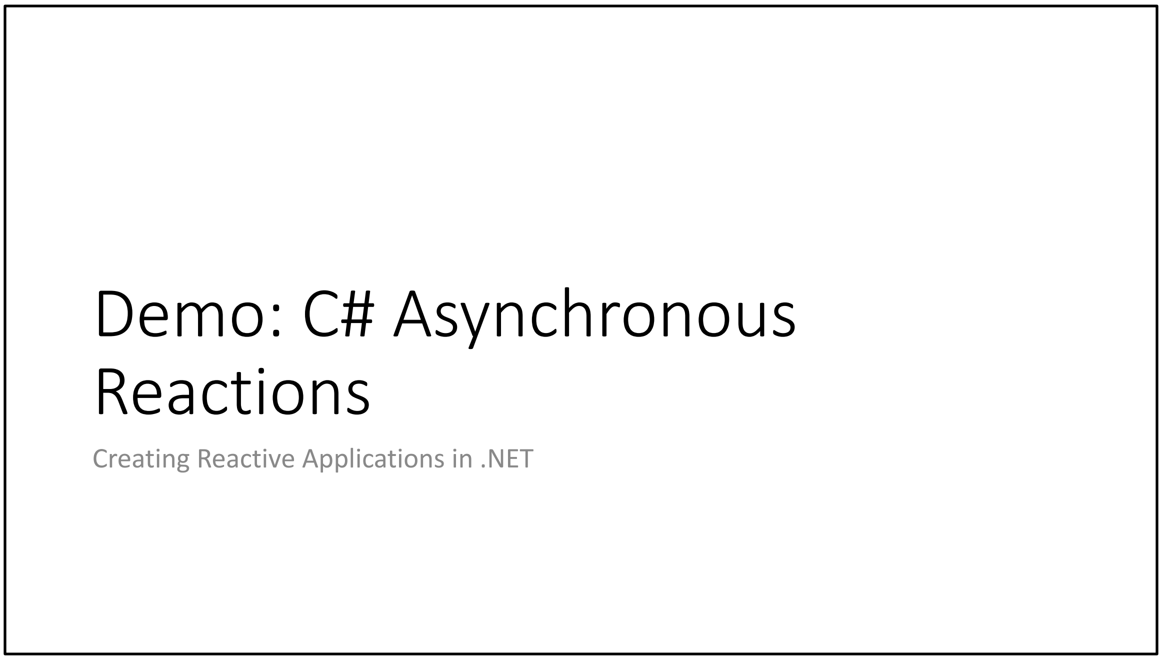
https://unsplash.com/photos/white-front-load-washing-machine-nUL_PP69IPA

In a similar vein, we don’t force people when they order products to wait until the product shows up at their residence.

https://unsplash.com/photos/person-using-laptop-computer-holding-card-Q59HmzK38eQ



There are a number of asynchronous ways we can react to changes in C#



Demo: C# Asynchronous Reactions

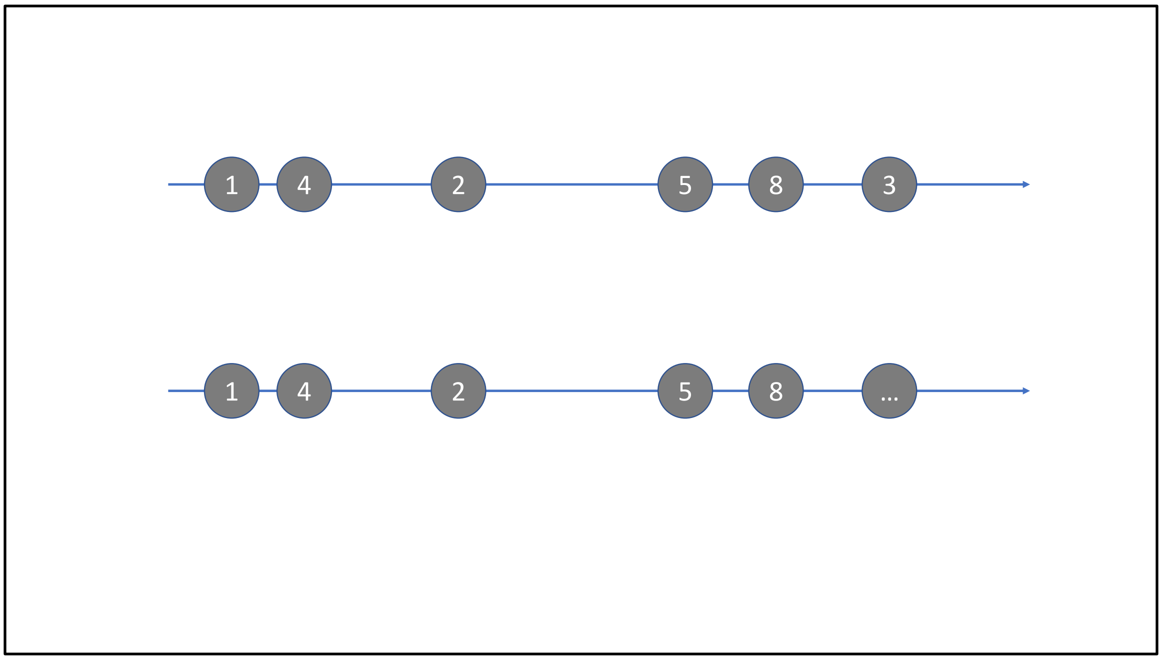
Creating Reactive Applications in .NET

Rx.NET

Creating Reactive Applications in .NET

There are lots of asynchronous ways to handle events:

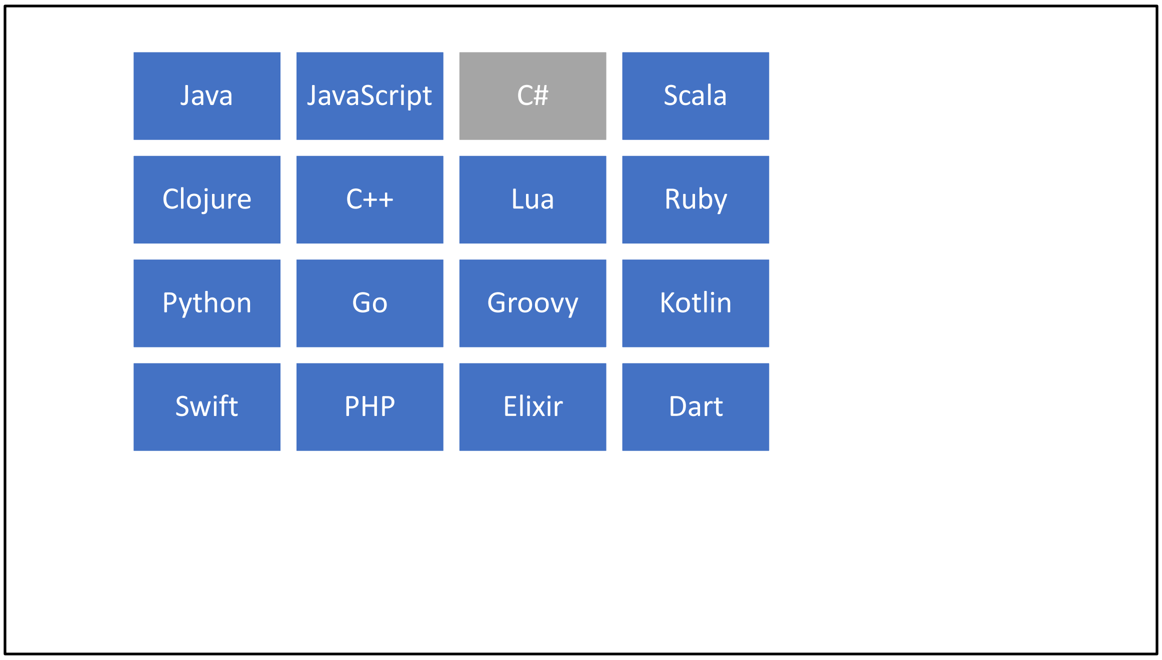
- Amazon Simple Queuing Service (SQS)
- Actors – Orleans
- Web Sockets – SignalR
- Messaging – Wolverine
- And others, like Kafka



So how does a .NET dev handle these kinds of streams? There are two that you can think of: one that has a finite set of items, and one that is “infinite” – you don’t really know when it’s going to end. (Note: these are marble diagrams)

This is where Reactive Extensions in .NET, or Rx.NET, comes in very handy.

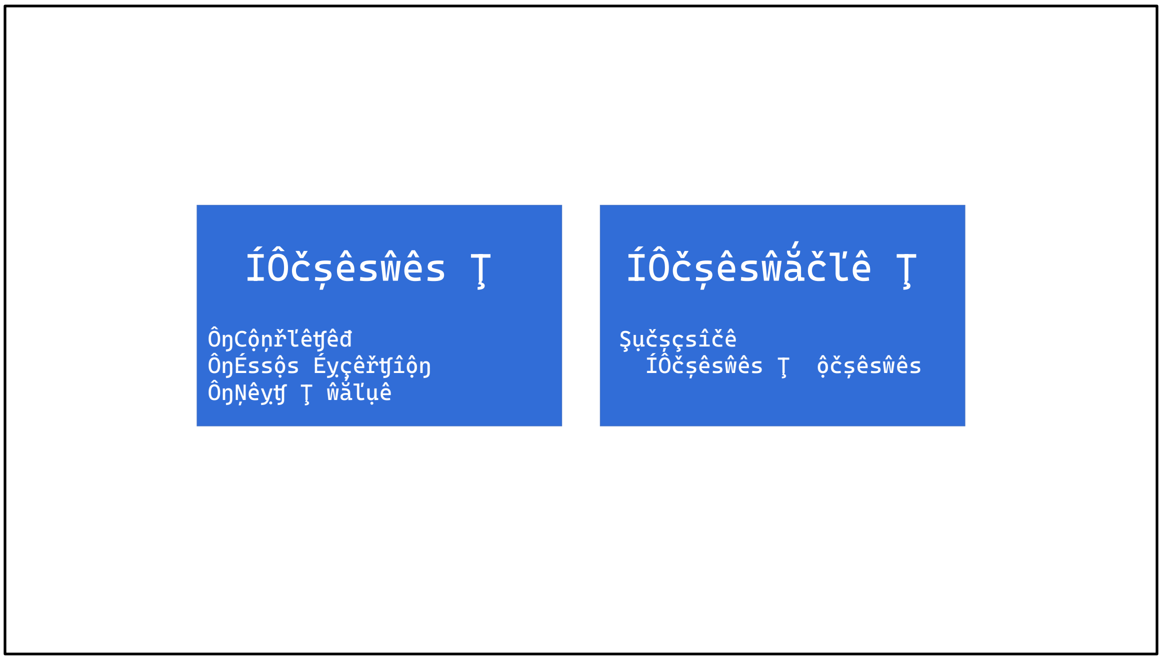
https://github.com/dotnet/reactive



Java	JavaScript	C#	Scala
Clojure	C++	Lua	Ruby
Python	Go	Groovy	Kotlin
Swift	PHP	Elixir	Dart

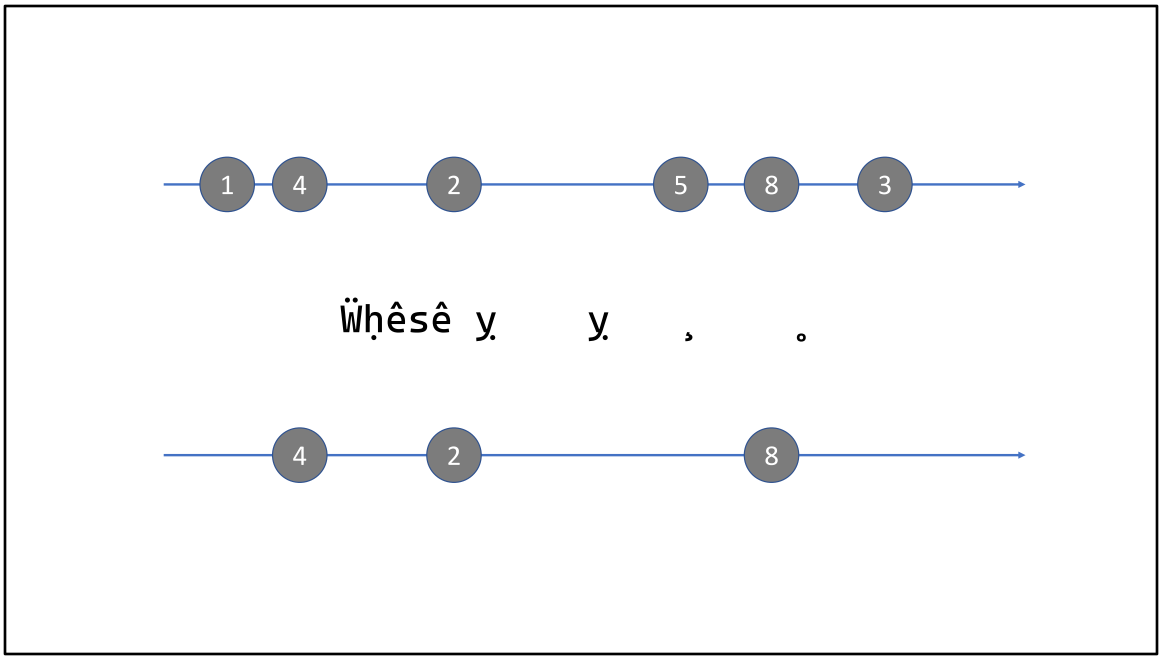
Here are all the known languages that have an implementation of Rx to some degree.

https://reactivex.io/languages.html

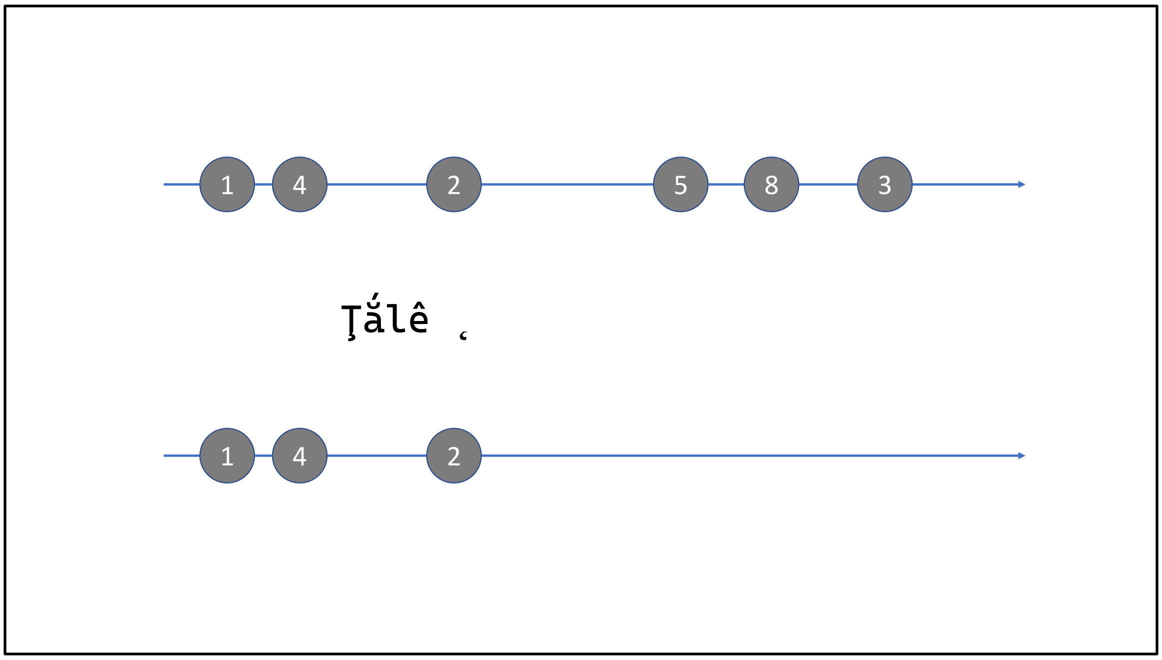


Rx is based on two key interfaces, IObserver<T> and IObservable<T>

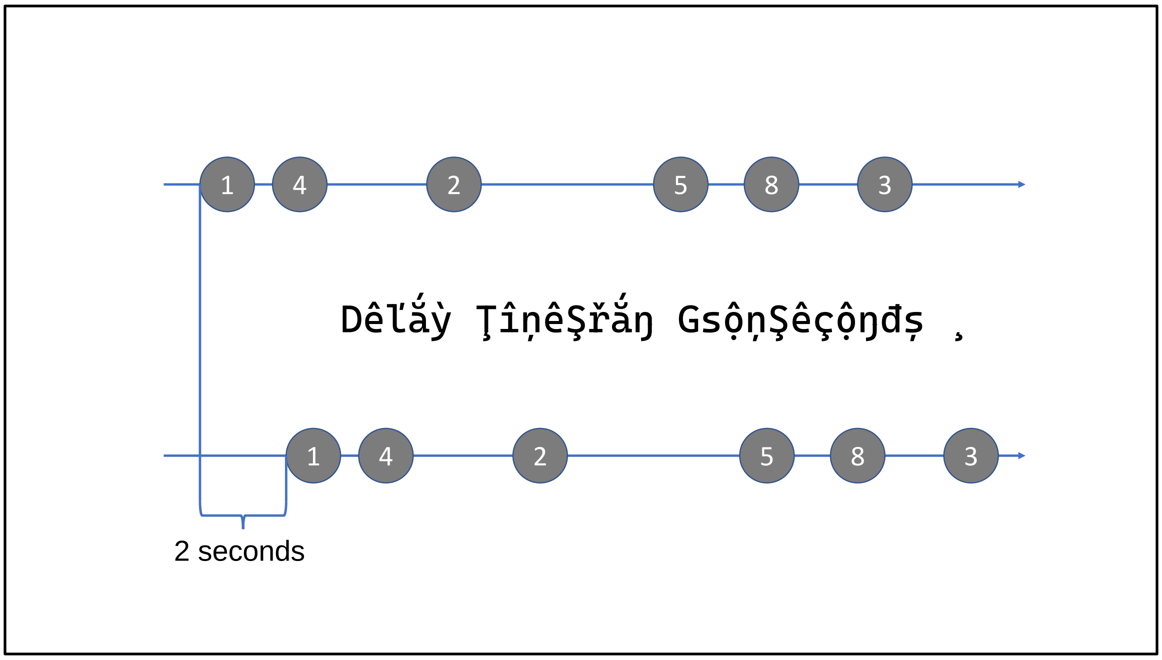
https://learn.microsoft.com/en-us/dotnet/api/system.iobserver-1
https://learn.microsoft.com/en-us/dotnet/api/system.iobservable-1



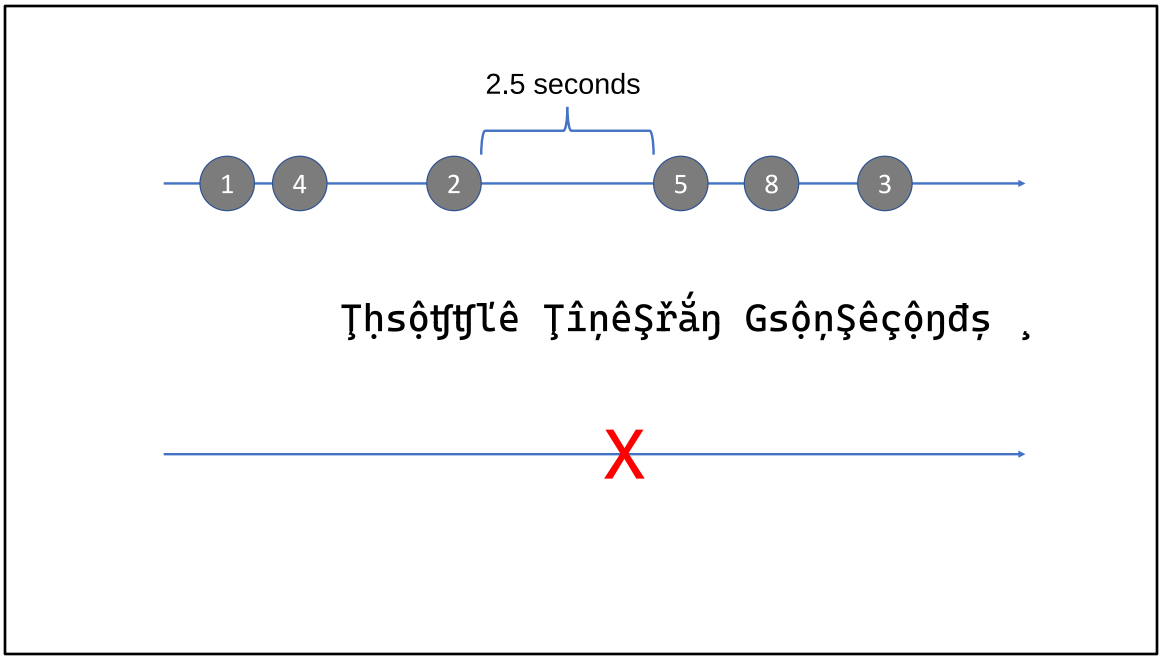
What’s so awesome about Rx is that it makes streams composable. That is, you can write code to handle these streams just like you would with collections. Let’s say I only cared about even numbers. I could a Where clause….



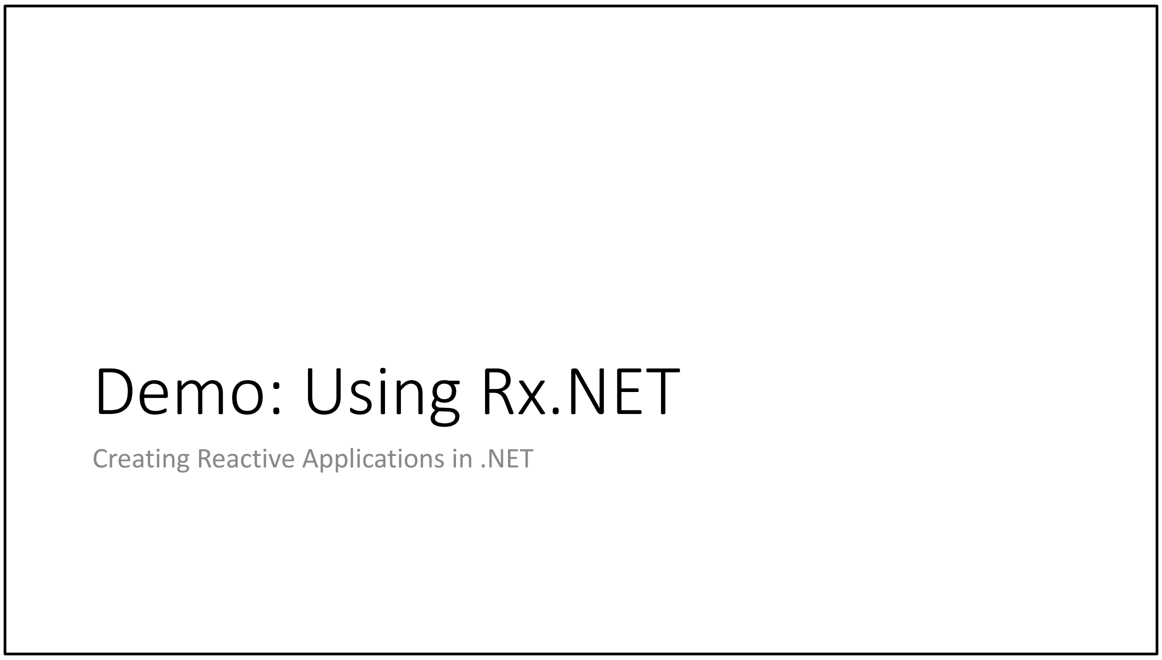
Or I only wanted the first three values. Take()



Or I wanted to delay the production of these values. Delay()

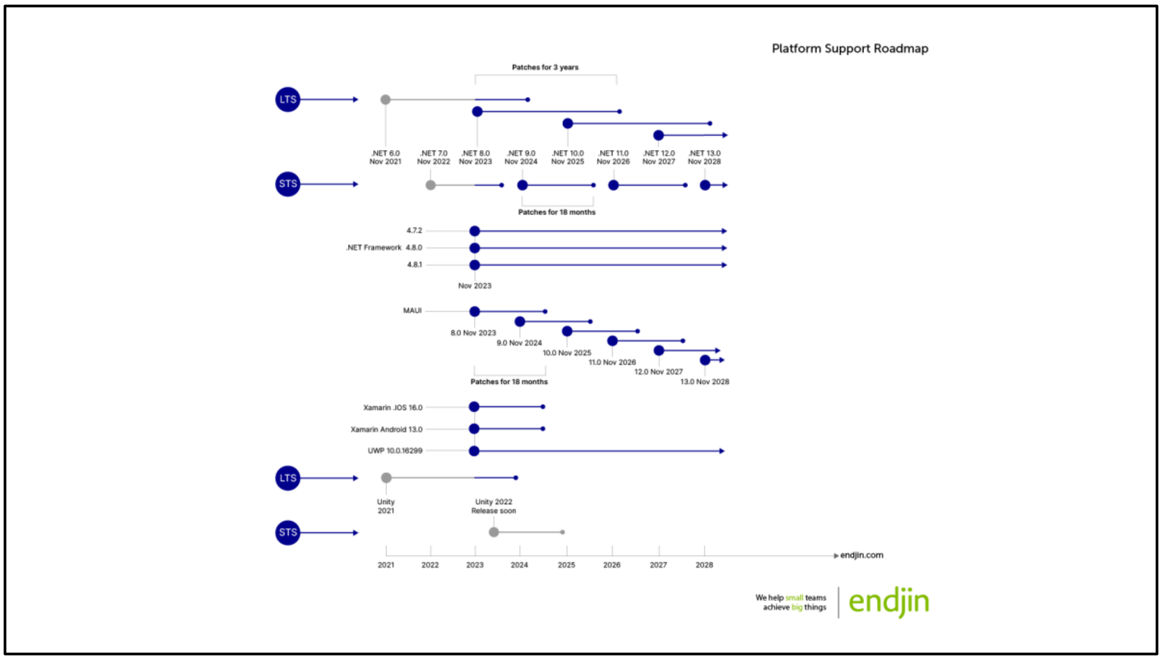


Or I want to know when something hasn’t happened in a while. Throttle()



Demo: Using Rx.NET

Creating Reactive Applications in .NET

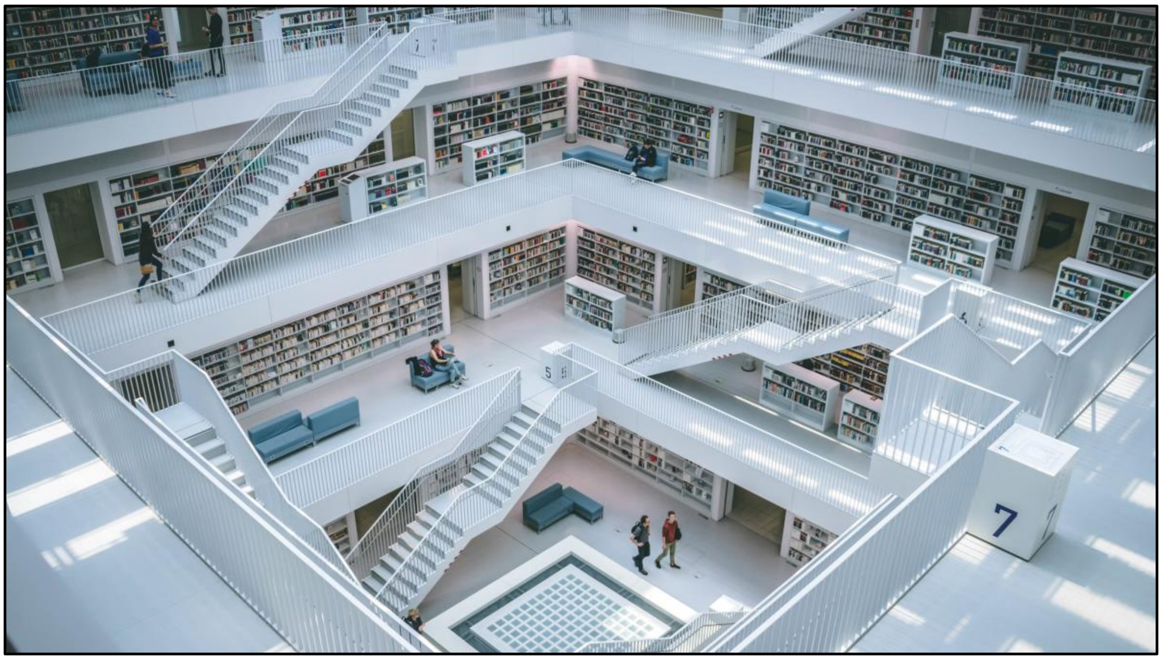


The history of Rx.NET is interesting, and somewhat sad. Thankfully Endjin has taken ownership of the library and has created a roadmap for the future. There's a lot of work here and some difficult decisions to be made, but my personal hope is that they get the support needed to bring Rx.NET back to the forefront.

https://github.com/dotnet/reactive/blob/main/Rx.NET/Documentation/RX-Platform-Support-Roadmap.png
https://github.com/dotnet/reactive/blob/main/Rx.NET/Documentation/Rx-Roadmap-2023.md

Call to Action

Creating Reactive Applications in .NET

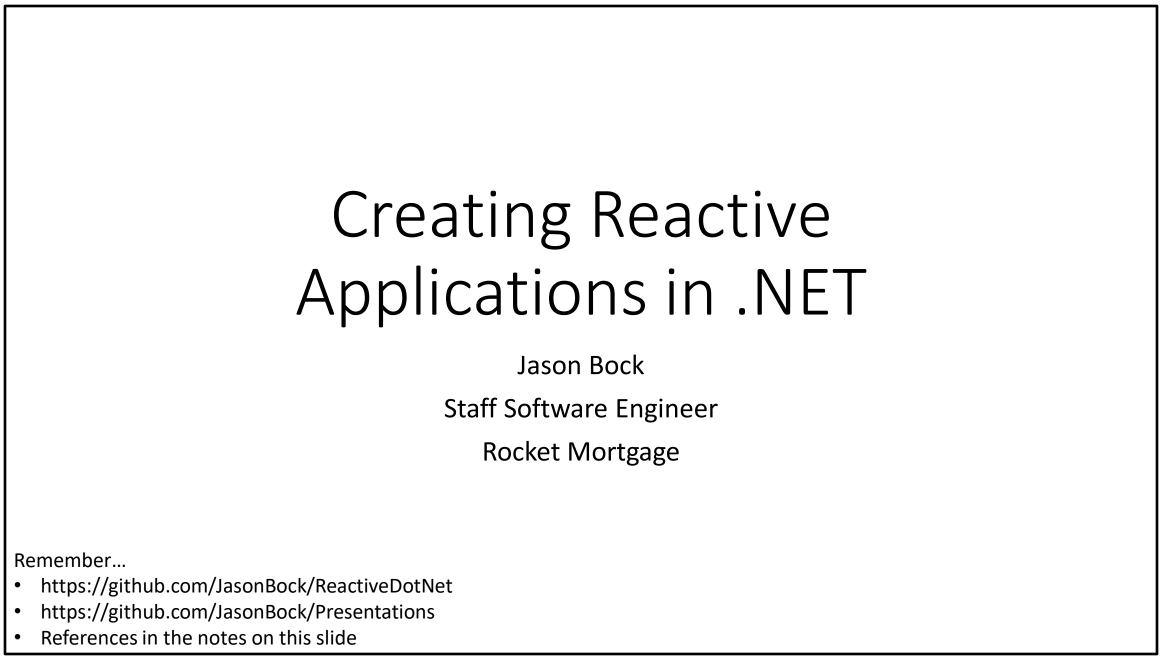


Hopefully you learned something new in this session. Keep diving in! Think about places you can apply reactive programming to your applications.

https://unsplash.com/photos/people-walking-inside-library-Y7d265_7i08

Consider volunteering to help out with Rx.NET. It’s a wonderful library that needs community support to not only stay alive, but thrive.

https://www.pexels.com/photo/volunteers-collecting-trash-on-green-grass-field-5029859/



Creating Reactive Applications in .NET

Jason Bock

Staff Software Engineer

Rocket Mortgage

Remember…
- https://github.com/JasonBock/ReactiveDotNet
- https://github.com/JasonBock/Presentations
- References in the notes on this slide