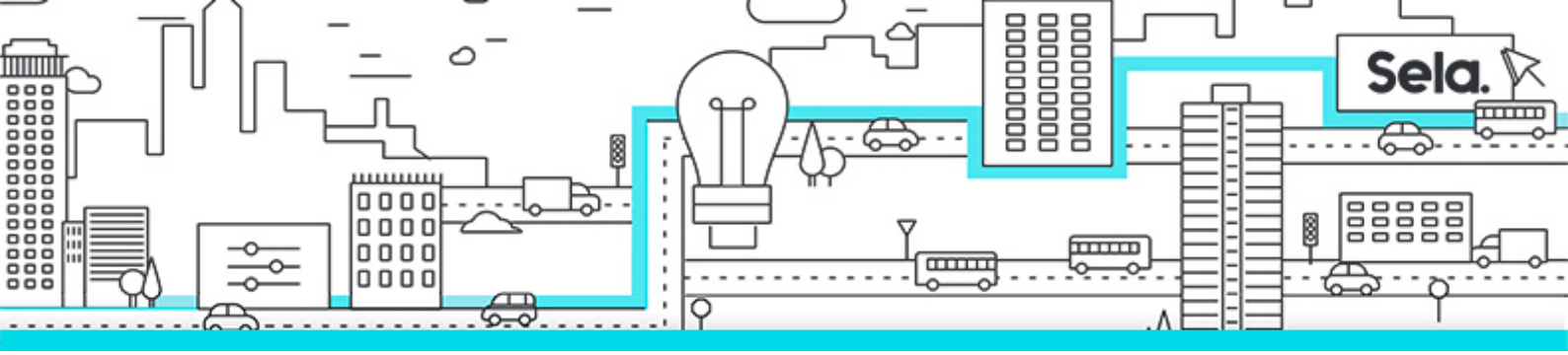
Sela.

RXTDF

Asynchronous Computing and Composing Asynchronous and Event-Based

college@sela.co.il 03-6176666



Asynchronous Computing and Composing Asynchronous and Event-Based

RXTDF - Version: 1

5 days course

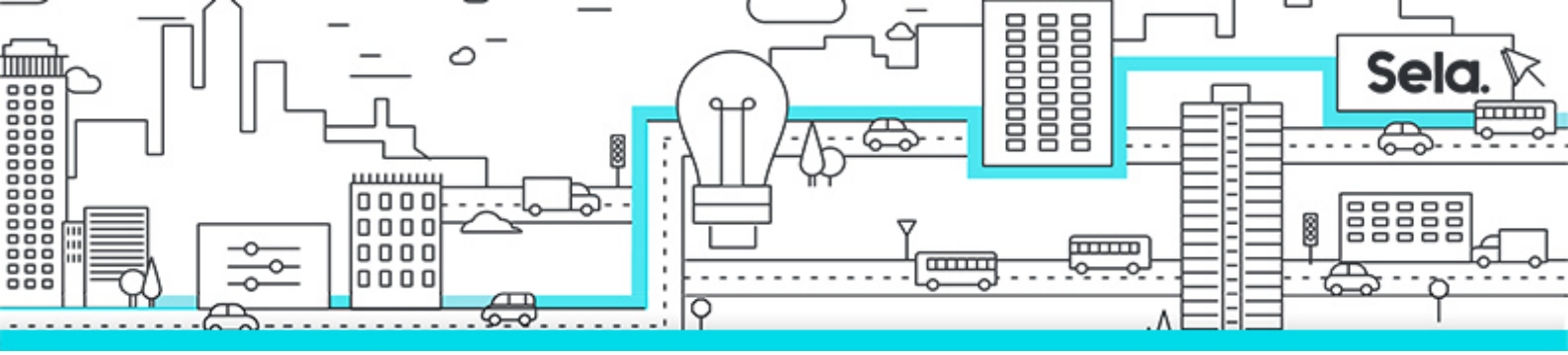
Description:

5 days that target the different approach to parallelism and computation of asynchronous events using
Async / Await, Reactive Extension (RX) and TPL Dataflow.
Rx is a functional programming library designed to handle complex event processing. The course deep-dives into the library's concept and guidelines. It will cover topics like exception handling, testing, remote processing, and scheduling. Students will master both practice and theory and become familiar with numerous RX operators.
TPL Dataflow is an agent-based library designed to achieve high throughput and low latency for both I/O- and CPU-bound operations. The course deep-dives into the library's concept and guidelines. Students will master both practice and theory.

Intended Audience:

.NET developers or team leaders with:

-
-
-
-



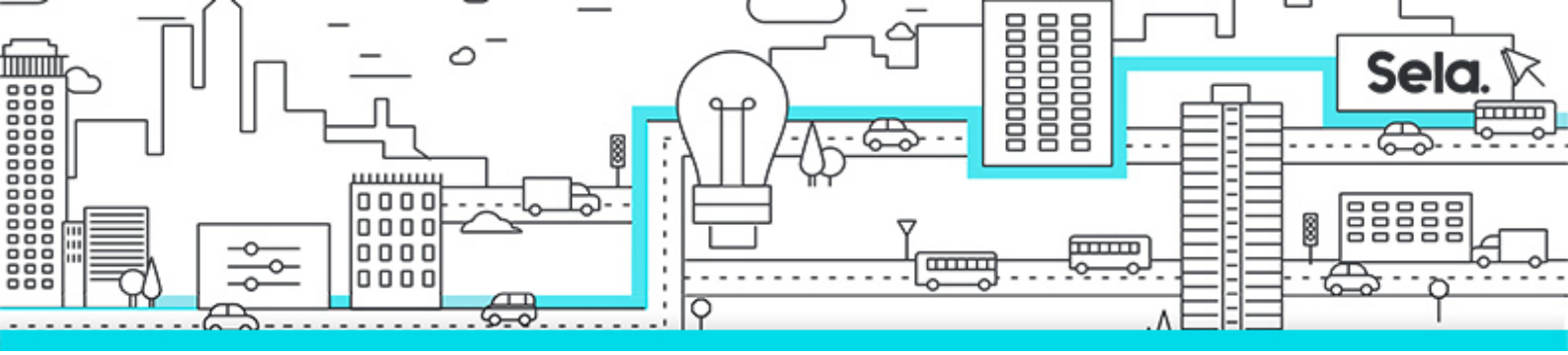
Objectives:

- Appreciate the architectural and design principles of Async, RX and TPL Dataflow programming model
- Practice complex event and messaging processing
- Learn to handle exceptions
- Test complex event pipeline
- Know how to design RX flow and TPL Dataflow
- Learn to combine different approach together
- Master design principles and guidelines of the agent base model
- Practice the complex parallel flow
- Understand TPL Dataflow blocks and performance tuning

Topics:

- **Introduction**
  - Moore's Law
  - Amdahl's Law
  - Thread safety
  - .NET Parallel History
  - Task and Task
  - Custom Task
  - Continuation
  - Async and Await
  - Cancellation
  - Exception Handling
  - Concurrent Collection

- **Rx Introduction**
  - What is Rx?
  - Push standard
  - LINQ-able
  - Like Events but better



- Course Goals
- Why Rx

• **Push vs. pull**

• **Test Case: Cloud search**

• **Get started**
- NuGet

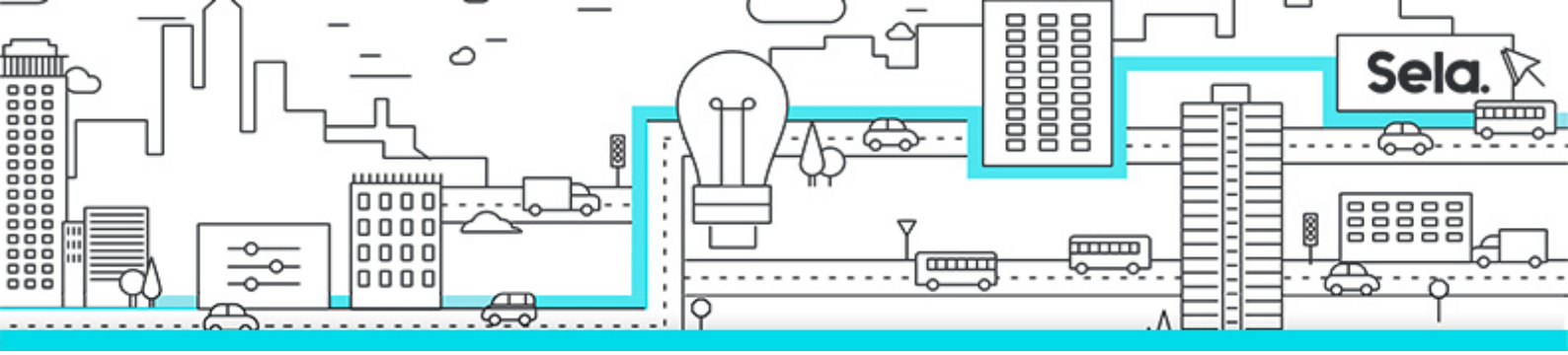
• **Concept**
- Producer / Consumer

• **Library structure**
- Different offering
- Enlighten concept

• **Marble Diagram**
- Concept
- Select
- Where

• **Built-in factories**
- Interval
- Timer
- Range
- Return
- Create
- Generate

• **Monitoring**
- Do operator



- Visual RX

• **LAB 01**

• **Concurrency model**
- Scheduler
- Built-in Scheduler
- ObserveOn
- SubscribeOn

• **Backwards compatibility**
- Composite events

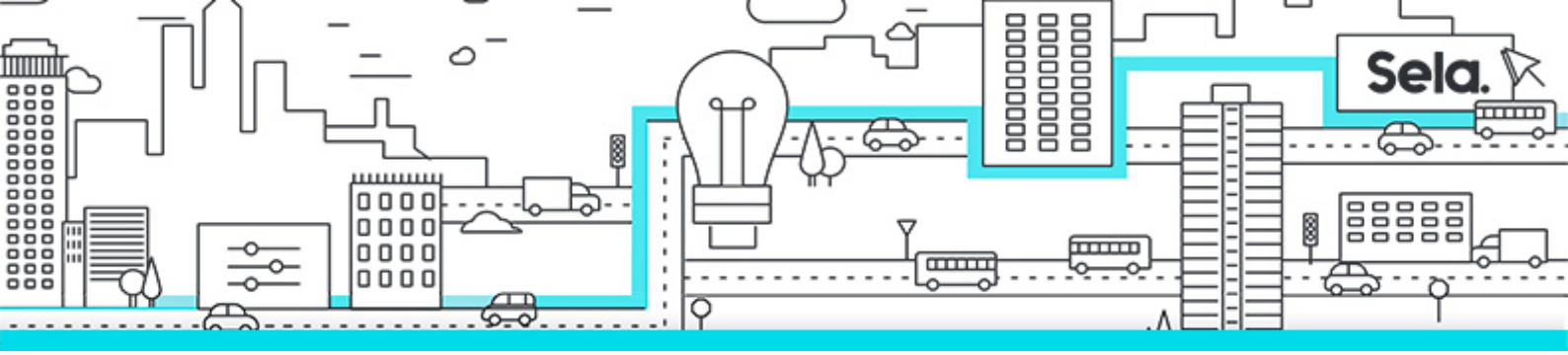
• **Exception Handling**
- Retry
- Catch
- Finally
- SubscribeSafe

• **LAB 02**

• **Producer nature**
- Hot Vs. Cold Observables
- Publish
- RefCount

• **Subjects**
- Subject
- Replay and ReplaySubject

• **Operators**
- District



- DistinctUntilChanged
- Sample
- Aggregate
- Scan

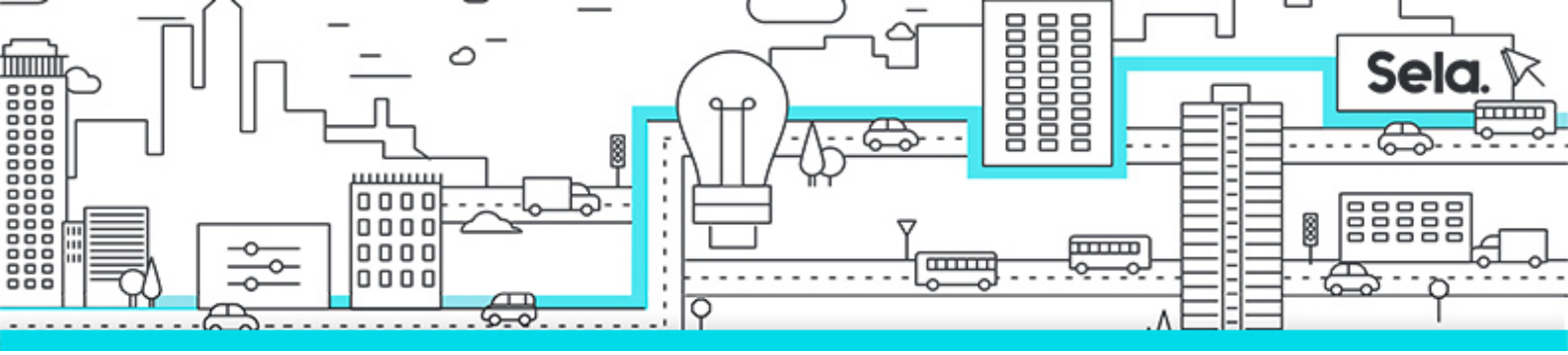
• **Common Combinators**
- Merge
- Zip
- CombineLatest
- Amb

• **Splitting**
- Buffer
- Window
- Select Many
- Group By

• **LAB 03**

• **Time-oriented**
- Interval
- Timeout
- Timestamp
- TimeInterval
- Throttle
- Sample
- Take and Skip
- Generate Delay
- Delay Subscription

• **Time-based Combinators**
- Join



- Group Join

• **Disposables**
  - Create
  - CompositeDisposable
  - RefCountDisposable
  - CancellationDisposable
  - BooleanDisposable
  - ContextDisposable
  - ScheduledDisposable
  - SingleAssignmentDisposable
  - MultipleAssignmentDisposable
  - SerialDisposable

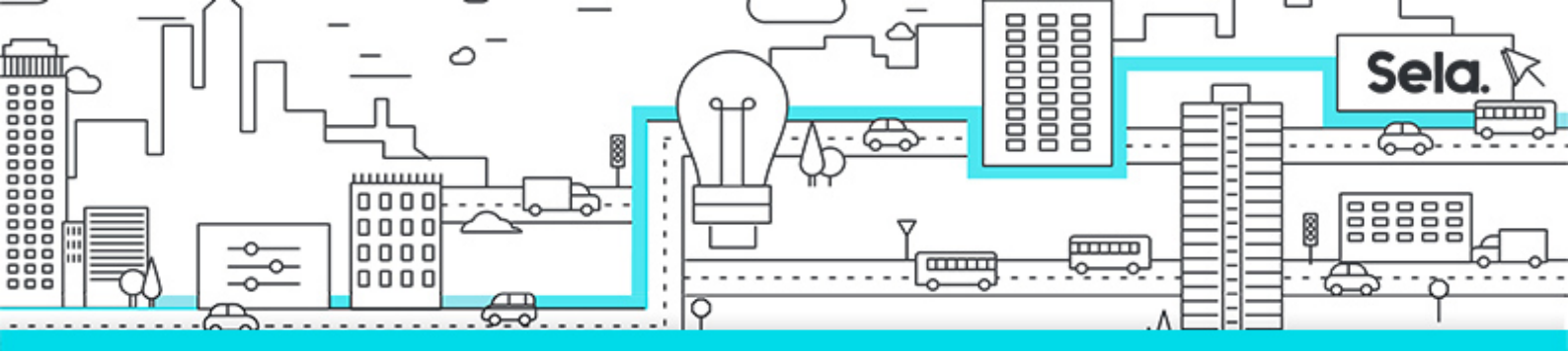
• **LAB 04**

• **Scheduling**
  - Custom Scheduler
  - Virtual time
  - Historical Scheduler

• **Testing**
  - Notification
  - TestScheduler
  - CreateHotObservable / CreateColdObservable

• **Custom Operations**
  - Defer
  - Extending the framework

• **LAB 05**



- **Remote**
  - Rx-Remoting
  - IQbservable

- **Introduction to Async / Await (C#5)**
  - Using
  - Loops

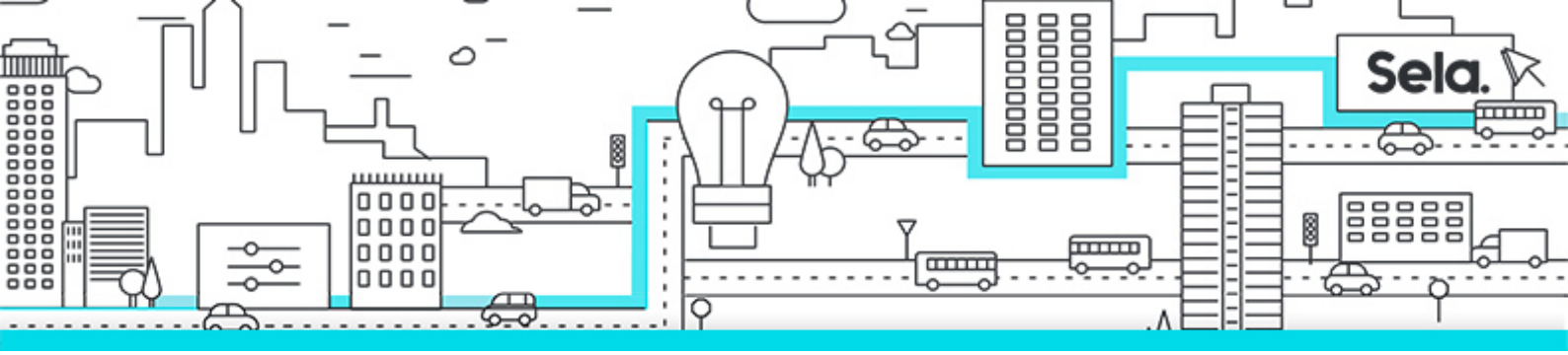
- **TPL Advance**
  - Scheduling
  - Async and UI

- **Parallel I/O**
  - What makes I/O operations different?
  - I/O completion port

- **What is TPL Dataflow?**
  - The agent base concept
  - Evolution
  - Goals

- **Getting started**
  - Namespace
  - NuGet

- **Contract**
  - Source API
  - Target API
  - Block API
  - Push vs. Pool



- **Blocks**
  - Blocks categories

- **Action block**
  - Structure
  - Functionality
  - Throttling

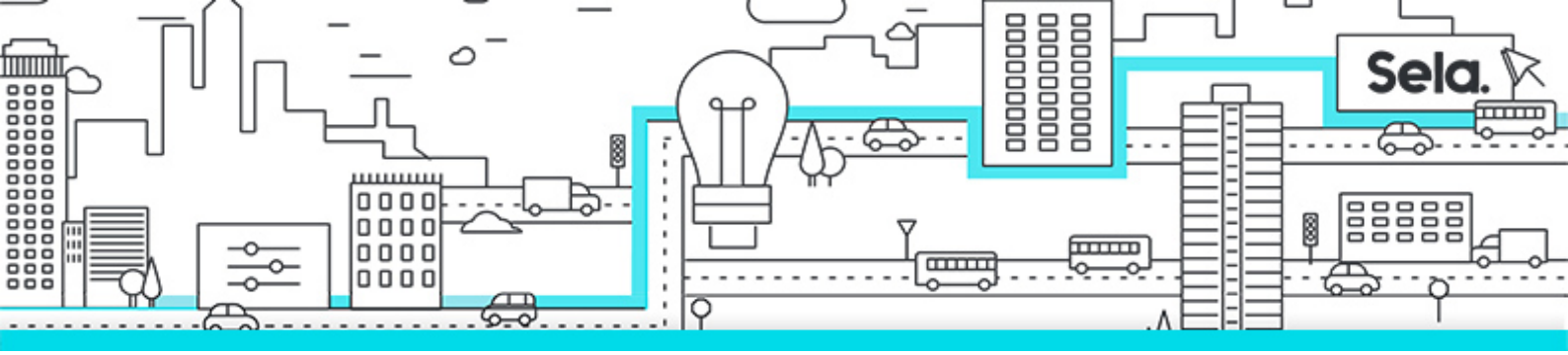
- **Buffer Block**
  - Structure
  - Functionality
  - Push and Pool
  - Bounded capacity

- **Broadcast Block**
  - Structure
  - functionality
  - What makes it different than Buffer Block?

- **Transform Block**
  - Structure
  - Functionality

- **Transform Many Block**
  - Structure
  - Functionality

- **TPL Dataflow and Async**
  - Using async / await with TDF
  - Processing I/O operations



- **LAB 06**

- **Performance tuning**
  - MaxMessagesPerTask

- **Case Study: Web Crawler**

- **Batch Block**
  - Structure
  - Functionality

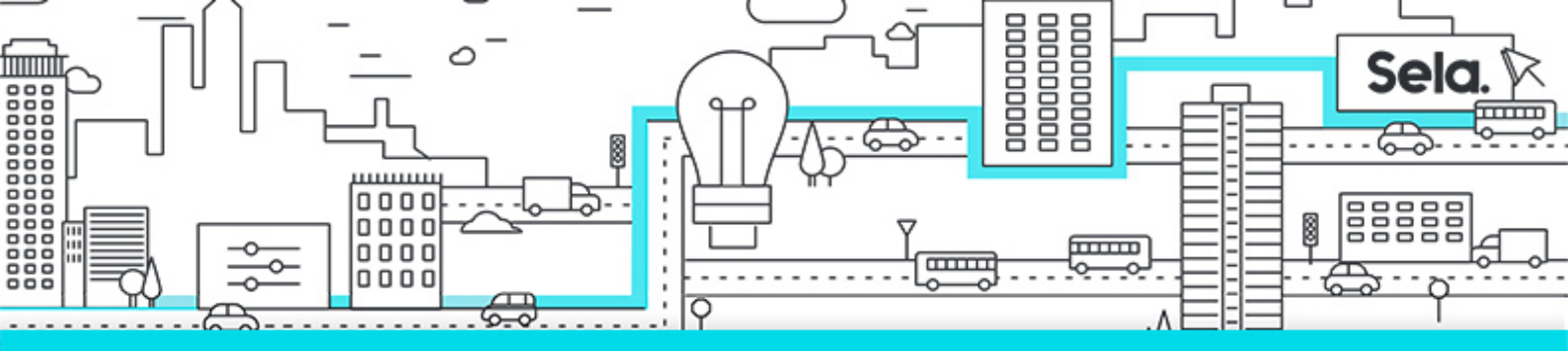
- **Join Block**
  - Structure
  - Functionality

- **LAB 07**

- **Greediness**
  - Targeting complex scenarios and correlation
  - ConsumeMessage API
  - Two-phase commit
  - ReserveMessage API
  - ReleaseReservation API

- **Greediness and built-in blocks**
  - BatchBlock
  - JoinBlock

- **BatchedJoinBlock**
  - Structure
  - Functionality



- **WriteOnceBlock**
  - Structure
  - Functionality

- **LAB 08**

- **Rx vs. TDF**
  - Design differences
  - Better together

- **Summary**