Table of Contents

Preface vii

1. Concurrency: An Overview 1
1.1. Introduction to Concurrency 1
1.2. Introduction to Asynchronous Programming 3
1.3. Introduction to Parallel Programming 7
1.4. Introduction to Reactive Programming (Rx) 10
1.5. Introduction to Dataflows 12
1.6. Introduction to Multithreaded Programming 14
1.7. Collections for Concurrent Applications 15
1.8. Modern Design 15
1.9. Summary of Key Technologies 16

2. Async Basics 19
2.1. Pausing for a Period of Time 20
2.2. Returning Completed Tasks 22
2.3. Reporting Progress 23
2.4. Waiting for a Set of Tasks to Complete 24
2.5. Waiting for Any Task to Complete 27
2.6. Processing Tasks as They Complete 28
2.7. Avoiding Context for Continuations 32
2.8. Handling Exceptions from async Task Methods 33
2.9. Handling Exceptions from async Void Methods 34

3. Parallel Basics 37
3.1. Parallel Processing of Data 37
3.2. Parallel Aggregation 39
3.3. Parallel Invocation 41
3.4. Dynamic Parallelism 42

3.5. Parallel LINQ 44

4. Dataflow Basics **47**
4.1. Linking Blocks 48
4.2. Propagating Errors 49
4.3. Unlinking Blocks 51
4.4. Throttling Blocks 52
4.5. Parallel Processing with Dataflow Blocks 53
4.6. Creating Custom Blocks 54

5. Rx Basics **57**
5.1. Converting .NET Events 58
5.2. Sending Notifications to a Context 60
5.3. Grouping Event Data with Windows and Buffers 62
5.4. Taming Event Streams with Throttling and Sampling 64
5.5. Timeouts 66

6. Testing **69**
6.1. Unit Testing async Methods 70
6.2. Unit Testing async Methods Expected to Fail 71
6.3. Unit Testing async void Methods 73
6.4. Unit Testing Dataflow Meshes 74
6.5. Unit Testing Rx Observables 76
6.6. Unit Testing Rx Observables with Faked Scheduling 78

7. Interop **83**
7.1. Async Wrappers for "Async" Methods with "Completed" Events 83
7.2. Async Wrappers for "Begin/End" methods 85
7.3. Async Wrappers for Anything 86
7.4. Async Wrappers for Parallel Code 88
7.5. Async Wrappers for Rx Observables 89
7.6. Rx Observable Wrappers for async Code 90
7.7. Rx Observables and Dataflow Meshes 92

8. Collections **95**
8.1. Immutable Stacks and Queues 98
8.2. Immutable Lists 100
8.3. Immutable Sets 102
8.4. Immutable Dictionaries 104
8.5. Threadsafe Dictionaries 106
8.6. Blocking Queues 108
8.7. Blocking Stacks and Bags 110
8.8. Asynchronous Queues 112

8.9. Asynchronous Stacks and Bags 115
8.10. Blocking/Asynchronous Queues 117

9. Cancellation **121**
9.1. Issuing Cancellation Requests 122
9.2. Responding to Cancellation Requests by Polling 125
9.3. Canceling Due to Timeouts 126
9.4. Canceling async Code 127
9.5. Canceling Parallel Code 128
9.6. Canceling Reactive Code 130
9.7. Canceling Dataflow Meshes 132
9.8. Injecting Cancellation Requests 133
9.9. Interop with Other Cancellation Systems 134

10. Functional-Friendly OOP **137**
10.1. Async Interfaces and Inheritance 137
10.2. Async Construction: Factories 139
10.3. Async Construction: The Asynchronous Initialization Pattern 141
10.4. Async Properties 144
10.5. Async Events 147
10.6. Async Disposal 150

11. Synchronization **155**
11.1. Blocking Locks 160
11.2. Async Locks 162
11.3. Blocking Signals 164
11.4. Async Signals 165
11.5. Throttling 167

12. Scheduling **169**
12.1. Scheduling Work to the Thread Pool 169
12.2. Executing Code with a Task Scheduler 171
12.3. Scheduling Parallel Code 173
12.4. Dataflow Synchronization Using Schedulers 174

13. Scenarios **175**
13.1. Initializing Shared Resources 175
13.2. Rx Deferred Evaluation 177
13.3. Asynchronous Data Binding 178
13.4. Implicit State 180

Index **183**