

Symbols

_ (pattern) 66, 87
 -- (combinator) 425
 -- operator 450
 -= operator 462
 = operator (C#) 102
 -> (type constructor) 135
 -> symbol 49
 : (operator) 70
 : (pattern) 71
 :: operator 166
 ! operator 217
 ? 303
 ? operator 254
 ?: operator 36
 ?? 303
 ?> operator 254
 () value 96
 [..] syntax 70
 [...] syntax 321
 [] (indexer) 85
 [] (pattern) 71
 [] (value) 70
 [| ... |] syntax 275, 321
 @ operator 483
 * (type constructor) 60, 111
 /// comment 236
 \ 249
 && 303
 &&& operator 462
 %A (format specifier) 91
 %d (format specifier) 91
 %f (format specifier) 91
 %s (format specifier) 91
 %s format specifier 23

+ operator 47
 += operator 462
 += operator (C#) 102
 +> (custom operator) 145
 <- operator 59
 = operator 217
 =. *See* equality operator, F#
 ==. *See* equality operator
 => (lambda function) 133
 > operator 253
 >> operator 160, 435
 implementing 161
 | symbol 49
 |> operator 146
 |> operator. *See* pipelining operator
 || 303
 ||| operator 447

Numerics

2D arrays 278
 32-bit integer 265

A

abstract
 class 47, 241
 keyword 241
 abstract method 114
 adding 114
 abstract syntax
 representation 422
 abstract value description 242
 abstraction 143

Accept method 203
 accessed members 210
 accessibility area 55
 accessing data 363–372
 accumulated state
 event processing 464
 accumulating
 elements 272
 events 465
 more code 281
 accumulator argument
 264–266, 281
 list processing 271
 summing list 265
 action
 representing 213
 Action delegate 135
 Action<T> delegate
 parallelizing for loop 385
 active patterns 201
 actual value 340
 ad rotator 429
 adaptive document layout 196
 Add method 208
 Add Reference 92, 192, 293
 adding
 operations 202
 types vs. functions 118
 adding members
 avoiding changes in the code 238
 to a type declared earlier 238
 addition
 optimization 266
 addressing memory 397
 Aggregate clause 469

- Aggregate method 172
 - summing clicks 469
- aggregated statistics 364
- aggregated value
 - generic type 79
 - so far 465
- AggregateList method 76
- AggregateNumbers method 40
- aggregating
 - finding largest 79
 - list 76
- aggregation
 - of documents 198
 - integers 78
 - of a numeric range 40
 - operation 170
 - state 198
 - Visual Basic queries 469
 - See also* folding
- AI 18
- algorithm
 - available information 389
 - choosing 273
 - choosing dynamically 212
 - decision tree 223
 - expressing 11
 - loan suitability 223
 - See also* complexity
- Alt+Enter 23
 - shortcut 92
- alternating operations 171
- alternative values 114–122
 - higher-order functions 151
 - processing using HOFs in C# 153
- always function 466
- analyzing type signature 156
- and keyword 225
- angle, calculating 95
- animated value 16, 428–434
- animation 445–454
 - Behavior type 445
 - click event 472
 - combinator library 428
 - composing 450
 - concept 445
 - constant 446
 - creating 445
 - declarative C# library 16
 - displaying 446
 - drawing shadows 449
 - frame 439
 - individual observations 433
 - introducing 428
 - primitive combinators 449
 - reacting to events 471
 - rotating planets 425
 - rotation 452
 - solar system 452
 - speed up 451
 - switch function 472
 - using behaviors 448
 - in WPF 15
- animation example
 - using discriminated union 424
 - using LISP 421
- animation library
 - drawings 440
 - independent
 - components 429
- AnimShape type 427
- anonymous methods 77, 133
- anonymous type
 - inside flattening projection 333
 - use in aggregation 172
- anonymous type (C#) 179
- anonymous types (C# 3.0) 110
- anti-pattern
 - repetition 87
- AOP 44
- API
 - design 240
 - of F# library 294
 - key 363
- apostrophe-type 126
- appending members 238
- application
 - behavior-centric 206
 - creating 399
 - design 83
 - designing structure 411
 - partial 139
 - startup code 94
 - state 213
 - structure 178
 - See also* function application
- Application.Run method 312
- ApplicationClass class 379
- applying filters 205
- arbitrary size integer. *See* bigint
- architecture
 - reactive 460
- argument
 - irrelevant value 301
 - missing 109
- arguments
 - introduction of named 95
- arithmetic
 - for behaviors 449
- array
 - 2D 380
 - blurring values 277
 - cloning 276
 - converting to a list 90
 - creating new 398
 - expression 321
 - F# syntax 275
 - generating 321
 - initialization 275
 - not mutating 276
 - slicing syntax 376
 - type annotations 277
 - using functionally 276–277, 398
 - using functionally in C# 278
- array processing
 - parallel 405
- Array.map function 276
 - for color filters 398
- Array2D module 278, 398
- Array2D.init function 380
- Array2D.map
 - function 399
 - parallelizing 405
- Array2D.Parallel module 406
- arrays 275–279
 - vs. lists 277
- arrow symbol 135
- as this construct 447
- ascending sequence 316
- AsParallel method 21, 387, 418
- aspect
 - of declarative programming 39
- aspect of code 349
- aspect-oriented programming. *See* AOP
- AspectJ 44
- assembly
 - dynamic loading 208
- Assert.Equals method 293
- assignment
 - in C# 36
- assignment operator 127
 - avoiding 33, 276
 - using arrays 275
- asterisk (type) 60
- asterisk symbol 111
- Async
 - module 357
 - prefix 488
- async value 356

- Async.FromContinuations
 - method 361
 - Async.Parallel method 358
 - Async.RunSynchronously
 - method 357
 - Async.Start method 357, 413
 - Async.StartAsTask method 358
 - Async<'a> type 356
 - AsyncGetResponse method 356
 - asynchronous
 - callback 357
 - control flow 6
 - state machine 477
 - waiting for events 474
 - asynchronous operations
 - using delegate 488
 - asynchronous
 - programming 353–372
 - in C# 362
 - model 361
 - asynchronous workflow
 - 354, 412
 - building using
 - continuations 362
 - in C# 362
 - computation type 360
 - delayed code 357
 - dispose continuation 360
 - error continuation 360
 - executing 357
 - executing thread 475
 - handling exceptions 365
 - implementing primitives 361
 - parallel execution 358
 - processing messages 482
 - programming UIs 474
 - recursive loop 365
 - running on GUI thread 479
 - sequence of workflows 372
 - step-by-step 359
 - waiting for events 475
 - AsyncPostAndReply
 - method 484
 - AsyncReplyChannel type 481
 - AsyncSleep method 361
 - attribute
 - Fact 295
 - augmenting
 - code meaning 337
 - type 235
 - automatic
 - conversions 88
 - logging 346
 - properties 208
 - automatic generalization
 - 87, 130, 164–165
 - F# algorithm 164
 - restricting 170
 - automatic property
 - immutable type 73
 - average color
 - calculating 407
 - weighted 397
 - averaging
 - array values 277
 - AwaitEvent
 - method 489
 - primitive 474
- B**
-
- background
 - computations 312
 - process 412
 - type checking 47
 - background operation
 - running using asynchronous
 - workflows 413
 - balanced tree 280
 - bar chart 323
 - base class 114
 - batch processing 178
 - BeginGetResponse method 360
 - BeginXyz/EndXyz pattern 361
 - behavior 206
 - adding easily 206
 - collection 206–210
 - composed 219
 - as interface 207
 - loading using Reflection 208
 - multiple functions 219
 - OO representation 207
 - parallelizing 408
 - preserving 289
 - specifying 43
 - behavior (value)
 - absolute value 436
 - addition 436
 - addition using lifting 437
 - applying function 435
 - calculating with 450
 - changing using mutation 473
 - combining with events 471
 - constant 432
 - creating 431–434
 - creating animations 448
 - creating from event 472
 - hiding implementation 431
 - introducing 429–431
 - lifting in C# 438
 - lifting in F# 437
 - multiplying floats 448
 - oscilating 432
 - overloaded operators 450
 - potentially infinite set 440
 - primitive functions 431–434
 - primitives in C# 431
 - primitives in F# 433
 - reading value 434
 - real-world applications 429
 - representing in C# 430
 - representing in F# 429
 - speed up 451
 - squaring 435
 - type signatures 433
 - visualizing 432
 - working with 434
 - behavior difference 288
 - behavior-centric application 408
 - image processing 395
 - improving 240–243
 - behavior-centric program
 - 178, 206
 - Behavior.lift1 function 437
 - Behavior.map function 435
 - Behavior<"T"> type 429
 - Behavior<Drawing> type 445
 - behavioral design patterns 212
 - big list 261
 - big O notation 274
 - bigint type 88
 - binary tree 119
 - generating 391
 - parallel processing 391
 - representing in C# 393
 - BinaryExpression class 119
 - bind function 334
 - Bind member 340
 - Bind method
 - implementing
 - SelectMany 345
 - bind operation 174
 - analysis 156
 - asynchronous
 - computations 359
 - for options 155
 - queries 335
 - step-by-step evaluation 156
 - three steps 347
 - typical structure 344
 - binding
 - recursive 69
 - unit value 348

- Bitmap
 - converting to array 397
 - type 311
 - bitmap
 - drawing 100
 - file 82
 - in-memory 100
 - BitmapUtils module 397
 - blur effect 396
 - implementing 406
 - blur filter 278
 - blurring
 - parallelization 385
 - values 277
 - body
 - interpretation 318
 - boilerplate code 130
 - Boolean operator 35
 - bounding box 182, 190
 - branch
 - pattern matching 67
 - recursion 69
 - break statement 36
 - bringing a structure 354
 - brush 95
 - buffered download 355
 - bug
 - dereferencing null value 120
 - eliminating 292
 - fixing early 92
 - building blocks
 - putting together 421
 - built-in
 - concepts 39
 - language construct 386
 - business rules 423
 - busy loop 411
- C**
-
- C 8, 94
 - C#
 - accessing F# members 237
 - anonymous types 313
 - calling F# code 255
 - events are special language constructs 102
 - functional design 159, 178
 - higher-order functions 159
 - integration 233
 - interoperability 294
 - lambda expression 131
 - only expressions 56
 - params keyword 84
 - specify types 45
 - C# - F# parity 249
 - C# 2.0 355
 - iterators 316
 - lifting 436
 - C# 3.0 13, 93
 - anonymous types 110
 - closures 218
 - collection initializers 207, 402
 - extension methods 145, 239
 - object initializers 94
 - query syntax 328
 - similarity to F# 208
 - type inference 63, 127
 - type inference for local variables 46
 - C# 4.0 45, 95
 - dynamic 255
 - C# class library
 - using from F# 400
 - C# compiler
 - automatic lifting 437
 - implicit conversion 252
 - iterator transformation 317
 - query translation 469
 - statements 56
 - C# in Depth 163
 - C# Language Specification 163
 - C++ 8
 - caching
 - lazy values 307
 - list cells 308
 - results 266
 - using lazy values 310
 - calculation
 - using tuples 64
 - callback function 461
 - caller
 - returning to 263
 - caller thread
 - waiting 392
 - calling function twice 266
 - captured parameter 215
 - capturing 215
 - mutable state 217
 - reference cells 217
 - state in C# 214
 - state in F# 215
 - capturing mutable state 267
 - casting
 - class hierarchies 118
 - casting. *See* type cast
 - catching an exception 365
 - categories
 - classification 223
 - CCR 362
 - chain
 - event processing 468
 - chart
 - drawing 322
 - drawing bars 323
 - save as a bitmap file 82
 - See also* pie chart
 - ChartWizard method 381
 - child element 193
 - Choice type 489
 - Choice1Of2 constructor 489
 - Choice2Of2 constructor 489
 - Church, Alonzo 6, 31
 - circle
 - colorful 444
 - creating drawing 442
 - moving center 443
 - city
 - incrementing inhabitants 64
 - representing using tuple 61
 - city information
 - processing 160
 - class
 - constructing 93–94
 - functional and imperative 249
 - inheritance in F# 447
 - named 242
 - purely functional 249
 - class declaration 242
 - in F# 248
 - class hierarchy 47, 114
 - decision tree 227
 - representing document 203
 - revealing structure 114
 - classifying input 223
 - clean-up
 - encapsulating 184
 - cleaning resources 245
 - client
 - suitability 206
 - using record 209
 - client-server 480
 - CLIEvent attribute 471
 - cloning
 - objects 181
 - cloning a list 274
 - closure 215–219
 - in C# 218, 267
 - capturing color filter in C# 401
 - capturing reference cells 217
 - compilation 217
 - for loop 218

- closure (*continued*)
 - sharing value 217
 - using for memoization 267
- CLR 4
- clue
 - type of function 167
- code
 - assumptions about 289
 - augmenting with logging 346
 - block 286
 - changing 5
 - combining 14
 - compositionality 427
 - dependencies 19, 289, 390
 - duplication 221, 287
 - execution 384
 - fluency 159
 - lock-free 11
 - manipulating with 315
 - organizing using
 - modules 399
 - path 287
 - readability 11
 - recurring 185
 - scalability 383
 - separation 324
 - sharing 95
 - verbosity 63
 - See also* readability
 - See also* reasoning about code
- code block
 - independent 390
- code structure
 - similarity 221
- coding conventions 242
- coding style
 - sequence expressions 323
- collection 271–279
 - of commands 213
 - composing primitives 444
 - filtering 130
 - of filters 205
 - fully evaluated 410
 - of functions 207–208
 - initializers 207
 - parallel processing 384
 - of places 288
 - processing 130, 290
 - unified processing 314
 - using queries 110
- collection initializers
 - composing values 424
- collection of behaviors 206–210
 - in C# 207
 - in F# 209
- collection processing 13, 169
 - function composition 162
 - schedule example 154
 - simulating animals 415
- color
 - adjusting 396
 - calculating with 395
 - invalid 398
- color filter 396
 - applying 398
 - creating in F# 398
 - implementing in C# 397
 - turning to effects 401
- color type
 - customized 395
- combinator
 - for financial contracts 456
 - using together 425
- combinator library 425
 - for animations 428
 - declarative description 436
 - implementing in C# 426
- combining
 - concepts 237
 - elements 334
 - values 109
- comma-separated value. *See* CSV
- command
 - in imperative languages 8
 - organizing 8
 - parallel composition 11
- Command component 214
- command component
 - created implicitly 214
- command pattern 213
- command-line arguments 92
- comment
 - documentation 236
- common language 173
- common language runtime. *See* CLR
- Common Lisp 44
- common operations 194
- commonality
 - identifying 221
- communicating using
 - message 483
- communication
 - lightweight processes 480
- comparison
 - C# and F# classes 252
- compilation order 225
- compile-time
 - checking 51
 - error 292
- compiler
 - discriminated unions 118
 - enforcing null handling 120
 - type inference 127
- complete pattern 67
- complex product 222
- complex structure
 - building 420
- complexity 274
- component
 - projection 148
- component-wise operations 397
- composability 420
 - monads 342
 - option processing 345
 - sequence expressions 319
- composable design 421–428
- composable library 420–459
 - comparing approaches 423
- composed values 424
- composing functions and
 - objects 425
 - executable computation 422
 - implementing 422
 - in C# and F# 423
 - single expression 427
 - tree-like data structure 422
- Compose method 162
- composed behavior 219
 - building 221
 - in C# 223
- composed entity 423
- composed value 424–425
 - creating in C# 424
 - generic 122
 - structure 148
 - See also* alternative values
 - See also* function values
 - See also* multiple values
- composed workflow 358
- composing functions 425
- composing sequences 319
- composite
 - data type 34
 - design pattern 50
 - format string 90
 - pattern 200
- composite value 200
 - behavior 428
- composition
 - animations 16
 - of basic commands 42
 - financial modeling
 - primitives 457
 - graphical drawings 444

- composition (*continued*)
 - in XAML 14
 - operator 160
 - public 201
 - testing 299
 - types 45
 - unit testing 299
- compositionality 17, 420, 427
 - tuples 112
 - types 420
- computation
 - augmenting with logging 346
 - customized meaning 337
 - expression block 346
 - recursion 263
 - recursive 68
 - termination 69
 - using values 107
- computation builder 340
 - async 356
 - Bind member 340
 - Combine member 390
 - Delay member 359
 - for member 389
 - for options 343
 - Return member 341
 - return! keyword 356
 - TryFinally member 366
 - TryWith member 366
 - Using member 359
 - working with behaviors 440
 - Zero member 346, 390
- computation by calculation
 - 37, 302
 - bind operation 156
 - processing documents 190
- computation expression
 - 334–350
 - applications 315
 - asynchronous 358
 - complicated logic 339
 - composing 342
 - delayed 359
 - do! keyword 348
 - for keyword 336, 389
 - for options 343
 - implementing builder 340
 - language-oriented programming 423
 - let! keyword 336
 - limitations 315
 - operation types 341
 - parallel sequences 389
 - primitives 390
 - refactoring 349
 - return keyword 337
 - translation 340, 349
- computation type 338
 - Async 360
 - logging 346
 - primitives 339
 - underlying structure 339
 - value wrapper 338
- computer memory 275
- computing
 - on demand 307
- concepts
 - built-in 39
 - combining 237
 - object-oriented 47
 - using together 441
- conceptual relation 328
- concrete
 - tests 207
 - value 242
- concrete object type 248–251
 - with events 470
- concurrency 10
 - message passing 490
 - shared memory 490
- Concurrency and Coordination Runtime. *See* CCR
- concurrency models
 - See also* declarative parallelism
 - See also* task-based parallelism
- concurrent programming 460
- condition
 - testing clients 206
- conditional compilation 293
- configurable income test 217
- configuration
 - switching 300
- configuring
 - loan tests 214
 - objects 427
- cons cell 69
 - decomposing 71
- Cons discriminator 166
- consistency
 - of styles 243
- console
 - application 89
 - window 89
- console output
 - changing color 246
- Console.WriteLine method 90
- constant complexity 274
- constant function. *See* always function
- ConstantExpression class 119
- constructor
 - in F# 248
 - overloading 249
 - referencing this 447
 - using from F# 93
- context 300
 - of data processing 187
 - scope using use 247
- continuation 279–283
 - argument 281
 - asynchronous execution 360
 - in C# 281
 - disposal handling 360
 - error handling 360
 - let! keyword 360
 - syntactic sugar 360
 - writing code 281
- contract data type 455
- contract. *See* financial contract
- control
 - accessing from threads 475
 - program execution 460
- control flow 261
 - asynchronous 6
 - rectangle drawing 477
 - sequences 320
- control flow construct
 - reactive applications 474
- control flow logic. *See* logic
- control structure
 - adding 9
 - implementing 43
- conversion
 - from XML 366
- converting representations 188
- coordinates
 - representing 112
 - specifying by moving 443
- copy and paste 403
- copying code
 - avoiding 76
- core meaning 349
- core part
 - extracting 184
- corner case
 - null value 120
- correctness 261
- corresponding
 - representations 193
 - type 78
- cosine function 452
- Count method 209, 387
- counting clicks 475
 - limiting rate 476

- counting words 194
- cross join operation 335
- CS0815 127
- CSV 82, 84
 - parsing 83
- CSV file 83
 - as an argument 100
- culture-specific code 247
- current
 - instance 236
 - result 264
 - time 429
- Current property 315
- currently executing
 - statement 36–37
- currying 140
 - See also* partial function application
- custom operator 305
 - infix notation 145
 - mimicking in C# 145
 - See also* overloaded operator
- customizing the meaning 315

D

- data
 - application design 177
 - binding 311
 - definition 109
 - exploring 354
 - importance 177
 - lists 166
 - recognizing 109
 - representation 178
 - reusing interactively 234
 - sample portion 369
 - untyped format 363
- data and behaviors 223
- data manipulation
 - application design 177
- data mining. *See* gathering information
- data parallel code 384
- data parallelism 386–390
- data representation
 - converting 188
 - multiple 178
- data set 370
- data source agnosticism 328
- data structure 6, 178
 - for chart drawing 86
 - context 187
 - conversions 188
- design 200
- designing 177
- immutable 33, 60
- keeping behaviors 205
- points 211
- potentially infinite 286
- traversal 198
- world simulation 408
 - See also* infinite data structure
- data type
 - converting 189
 - projection 148
- data types
 - functional point of view 305
- data vs. value 109
- data-centric applications
 - improving 234–240
 - parallelizing 408
- data-centric program 178
- data-driven programming 353–382
- database 287
 - asynchronous operations 354
 - join 330
- DataMember property 311
- DataSource property 311
- date and time
 - formatting 91
- DateTime type 91
- DateTimeOffset type 91
- deadlock 10, 490
- decision tree 223–229
 - building automatically 223
 - in C# 227
 - in F# 224
 - processing 226
- declaration
 - order 225
 - See also* value binding
- declarative
 - data parallelism 417
 - event handling 461
 - event handling in C# 467
 - event handling in Visual Basic 469
 - language 44
 - libraries 16
 - parallelism 11, 20
 - programming 12
- declarative code
 - visualizing 466
- declarative description
 - well-understood functions 436
- declarative library
 - as DSL 423
- declarative library. *See* composable library
- declarative programming
 - blur example 406
 - implementing technical details 422
 - parallelism 383
 - style 39–45
- declarative specification 427
 - executable 425
- declarative style
 - animations 16
 - composable design 421
 - execution details 386
 - and immutability 34
 - using .NET attributes 15
 - See also* LINQ
 - See also* XAML
- decomposing
 - argument 309
 - lists 87
 - result 347
 - tuples 87
- decomposition
 - of tuples 65
- deconstructing value 49
- decorator pattern 201
- deduction
 - of types 45
- deepest level of recursion 263
- default operator 294
- default value 111
- defaultof function 294
- degree (angle) 101
- delayed computation
 - evaluating 304
 - See also* lazy evaluation
- delayed execution 286
- delegate 132
 - automatic wrapping 214
 - function as argument 77
 - in C# 1.0 133
 - in C# 2.0 133
 - using events 462
 - using in F# 258
- delegate vs. function 133
- delegates
 - family of 133
- dependencies
 - hidden 5
- depth of the tree 391
- derived class 47
- design level 11

- design pattern
 - command 213
 - decorator 201
 - façade 403
 - functional monad 337
 - lifting 436
 - object-oriented 178
 - recursive discriminated unions 166
 - strategy 212
 - template method 228
 - visitor 119, 202
 - See also* composite design pattern
- design patterns 11, 199–204, 211–219
 - as functional concepts 8
- design principle
 - compositionality 420
- designer
 - Windows Forms 399
- designing type 298
- developing country indicators 369
- development
 - methodologies 286
 - phases 235
- development process 24, 222
 - explorative programming 369
- diagramming application 440
- dialog
 - opening a file 102
- dictionary
 - caching values 266
- Dictionary class 244
- Dictionary type
 - caching values 267
- directories
 - listing 175
- discoverability 235
- discriminated union 47
 - adding members 240
 - arguments 116
 - C# 393
 - compiled representation 118
 - creating value 115
 - document elements 182
 - encoding in C# 122
 - extracting values 117
 - financial contracts 458
 - as a language 424
 - mimicking in C# 117
 - pattern matching 116
 - processing using HOFs in F# 151
 - recursive 166, 187
 - representing messages 481
 - single-case 338, 430
 - specifying syntax 424
 - unnamed generic choice 489
 - working with 116
 - See also* option type
- discriminated union constructor
 - as function 485
- discriminated unions 115–122
 - single discriminator 308
 - structural equality 296
- discriminator 115
 - adding 119
 - single-case unions 338
 - type inference 164
- disk
 - asynchronous operations 354
- DisplayMember property 402
- Dispose method 245
 - automatic call 245
- disposing 97
- distance
 - calculating 415
- DivideByInt member
 - list average 407
- division with a remainder 109
- DivRem method 110
- do block 94
- do! keyword 348
- document 178
 - aggregating parts 197
 - annotated sample 187
 - class hierarchy 203
 - counting words 199, 203
 - decoration 201
 - displaying 186
 - drawing 178
 - drawing to a form 184
 - exploring structure 370
 - flat representation 182–187
 - list of elements 183
 - loading XML 191
 - manipulating 194
 - merging parts 196
 - parts 187
 - processing 178
 - recursive processing 190
 - structured
 - representation 187–194
 - terse representation 188
 - text elements 182
 - writing operations 194–199
 - XML representation 191
- document drawing
 - testing interactively 186
- documentation
 - comment 236
 - using types 51
- DocumentPart type 188
- DOM API 191
- domain 111
 - cross product 111
 - of values 109, 111
- domain-specific language. *See* DSL
- dot notation 145, 235
 - and pipelining 147
- downcast 254
 - final result to XElement 368
- download
 - asynchronous 354
- downloading data
 - reliability 366
- drag and drop
 - functional implementation 477
- drawing 440–445
 - animated circles 448
 - charts 322
 - composing 440, 444
 - creating 442
 - document 184
 - graphics 95
 - lifting to animations 448
 - representing 440
 - skewing, rotating, scaling 445
 - specifying location 443
- drawing a flowchart. *See* flowchart, drawing
- drawing changing over time 445
- drawing rectangles
 - accessing state 484
 - cancellation 489
 - collaboratively 480
 - UI 486
 - using mailbox 484
- drawing rectangles. *See* rectangle drawing
- drawing shapes problem 477
- drop-down
 - listing effects 402
- DSL 423
 - 123.Of.forever 433
 - embedded 423
- dual-core machine 389
- dynamic
 - loading 208

dynamic (*continued*)
 typing 6
 work distribution 384
 dynamic type 255
 test 118

E

eager evaluation 301
 efficiency
 array processing 277
 asynchronous operations 358
 of list processing 273
 sequence expressions 325
 Elapsed event 362
 elapsed time 403
 element
 removing 299
 XML 192
 elif keyword 387
 Elliott, Conal 428
 else clause
 inside sequence
 expressions 332
 embedded language 43, 424
 empty
 computation 346
 log 347
 empty list 69
 representing failure 373
 encapsulation 177
 higher-order operations 168
 of data types 235
 encoding sequence
 expressions 331
 end of the sequence 316
 EndGetResponse method 360
 Enumerable.Range method 316
 enumeration 114
 type 462
 enumerator object 326
 using directly 327
 equality
 customizing 243
See also reference equality
See also structural equality
 equality operator 296
 F# 297
 overloading 296
 Equals method 268, 297
 equation 285
 mathematics 286
 equivalence
 function declarations 133

Erlang
 concurrency 490
 errors
 when using mutable
 objects 289
 Esc key
 handling 489
 evaluation 32–39
 of arguments 38, 301
 step-by-step 37, 156
 strategy 302
See also eager evaluation
See also lazy evaluation
 evaluation order 300
 in Haskell 39
 in mainstream languages 301
 event 312
 Add method 102, 462
 AddHandler method
 102, 462
 in C# 462
 C# compatible 471
 carried value 466
 choosing one 488
 declarative handling 461
 declaring in F# 470
 in F# 462
 filtering 462
 handler 102
 keeping internal state 464
 merging 467
 multiple 488
 pattern matching 489
 Publish member 470
 publishing in F# 470
 raising 471
 reacting to 460
 in reactive animations 428
 registering handler 461
 RemoveHandler method
 102, 462
 similarity with lists 463
 time-varying values 472
 transforming 462
 Trigger member 471
 triggering 470
 using from F# 102
 waiting for 474
 waiting for occurrence 361
 Event class 470
 event handling
 declarative 462
 declarative in C# 467
 imperative 461
 using asynchronous
 workflows 475
 using LINQ 467
 event keyword 470
 event of behaviors 473
 event processing
 using queries in C# 468
 event-driven applications 460
 Event.filter function 463
 Event.listen function 463, 467
 Event.map function 464
 Event.merge function 465
 Event.scan function 464, 467
 EventArgs type 469
 evolution 222
 functions in C# 133
 Excel 83, 378–381
 chart 381
 writing data to 378
 exception
 handling 245
 throwing 84
 uncatchable 261
 Exception type 360
 executable computation
 composing 422
 executing
 as compiled code 315
 executing thread
 waiting in GUI 474
 execution
 controlling 460
 environment 315, 423
 of imperative programs 37
 order 6
 pattern 392
 recursive computation 262
 timing 394
 tracing on paper 37
 execution path
 selecting 75
 execution point
 lazy processing 410
 exn type. *See* Exception type
 expected value 298
 experimental code 83
 explicit
 cast 252
 class declaration 249
 conversions 88
 interface implementation 251
 lambda function 211
 mutable values 63
 parameter typing 134
 readonly fields 63

- explorative programming
 - 353, 362
 - accessing data 364
 - See also* data-driven programming
 - expose as member 238
 - expressing algorithms 11
 - expression 35
 - alternative interpretation 318
 - data structure 119
 - describing result 14
 - empty 99
 - let binding 56
 - programming using 14
 - scoping 246
 - sequencing 57
 - tree 119, 134, 423
 - type inference 128
 - See also* object expression
 - Expression Builder 428
 - Expression type 119
 - expression vs. statement 35
 - expressions
 - generating sequences 317
 - extending type 235
 - extensibility 118, 288
 - of the code 143
 - designing types 430
 - of a language 39
 - Extensible Application Markup Language. *See* XAML
 - extension
 - methods 42
 - property 433
 - extension method 115
 - for arrays 398
 - as custom operator 145
 - lifting behaviors 439
 - for numeric literals 432
 - supporting queries 344
 - for tuples 150
 - type extensions (F#) 239
 - external events 460
 - extracting value 344
- F**
-
- F#
 - class 26, 233
 - deconstruct the value 49
 - designer support 400
 - features 249
 - introducing 21
 - library project 256
 - module 399
 - native interoperability 397
 - object type 233
 - project 92
 - quotations 315, 423
 - redistributable 256
 - script file 22, 356
 - value treatment 56
 - F# abbreviation
 - seq type 316
 - F# code
 - evolution 222
 - how developed 82
 - standard meaning 423
 - F# compiler
 - closures 217
 - computation expression
 - compilation 340
 - deducing interface
 - declarations 241
 - discriminated unions 118
 - event treatment 471
 - generated constructor 257
 - measure attribute 374
 - merging type extensions 239
 - object expression
 - compilation 243
 - scoping 56
 - sequence expression
 - compilation 329
 - Struct attribute 395
 - type inference 47, 165
 - F# development process 24, 222
 - optimization 270
 - F# Interactive 23, 83
 - experimenting with
 - animations 446
 - measuring speed 388
 - printing sequences 316
 - shell 234
 - terminating commands 25
 - F# language
 - customizing 336
 - F# library
 - array processing 276
 - asynchronous
 - programming 357
 - calling from C# 255
 - function composition 161
 - option functions 154
 - option type 125
 - pipelining operator 144
 - F# object type 440
 - F# PowerPack
 - asynchronous I/O 361
 - units of measure 374
 - F# programming
 - iterative development 270
 - F# type
 - using from C# 256
 - See also* type
 - F# type declarations
 - attractive aspects 188
 - F# type extensions 238–240
 - façade pattern 403
 - factorial
 - generating sequence in
 - C# 316
 - generating sequence in
 - F# 320
 - iterative implementation 68
 - recursion 68
 - factorial function
 - optimizing using
 - memoization 268
 - failing operations 346
 - failure
 - detailed information 221
 - failwith function 84, 192
 - FastFunc type 258
 - field
 - declarations 34
 - immutable 60
 - public access 249
 - file reading 89
 - files
 - listing 175
 - filesystem
 - hiding in Haskell 346
 - FileSystemWatcher class 461
 - FillEllipse method 442
 - filter
 - blur 278
 - name 219
 - See also* color filter
 - filter function 131
 - tail recursive
 - implementation 271
 - filter operation 173
 - for documents 199
 - for events 462
 - filtering 130
 - filtering a list
 - strategy pattern 212
 - financial
 - application 214
 - financial contract
 - active date 456
 - combinators 456
 - evaluating trades 455

- financial contract (*continued*)
 - modeling 455
 - modeling language 457
 - primitives 455
 - using abstract values 458
- financial modeling
 - language 454
- finished application 222
- first argument 150
- first-class events
 - combining with behaviors 473
- first-class functions 41
- first-class value
 - events 468
- flags
 - pattern matching 192
- flat document
 - representation 182–187
- flattening projection
 - 329–334, 390
 - C# 332
 - nesting 332
 - replacing outer loop 331
 - SelectMany method 330
 - Seq.collect function 330
 - in sequence expressions 330
 - using directly 331
- flexibility
 - computation expressions 341
 - using interfaces 241
- flexible
 - income test 214
 - typing 289
- float function 88
- float type
 - with units of measure 52
- float32 function 88
- floating aggregate 465
- floating point
 - number 88
 - scientific 91
- flowchart
 - drawing 474
 - rectangle drawing 477
- fluency 159
- fluent integration 233
- fluent interface 428
 - for functional code 428
- fold operation 170
 - in C# 172
 - inverse operation 316
 - merging text parts 197
 - See also* Event.scan function
- folding
 - documents 198
 - See also* aggregation
- font
 - drawing 101
 - name 191
- for ... in construct 184
- for loop 42, 90, 152
 - array processing 276
 - capturing variable 218
 - parallel sequence
 - expression 389
 - parallelization 385
 - replacing with Seq.collect 331
- foreach loop
 - inside iterator 326
- foreach statement (C#) 90
- forestation percentages 374
- Forever method 16
- format string 23
- formatting strings 90
- Fowler, Martin 422, 428
- frame 439
- framework
 - creating animations 428
- Fran project 428
- from clause 328
 - custom computations 340
 - multiple 175, 333
 - translation to SelectMany 333
- FS0025 72
- FS0040 269
 - warning 322
- FSharp.Core.dll assembly 256
- FSharp.PowerPack.dll
 - library 355
- fst function 61, 113
- fsx file 22
- FSX files. *See* scripts
- fun keyword 133
- Func delegate 40, 77, 133, 206
 - interoperability 258
 - replacing interfaces 207
 - using from F# 258
- Func<T> delegate 133
- FuncList class 72
- FuncList.Cons method 74
- FuncList.Empty method 74
- function 40
 - aggregation 130
 - applying to behaviors 435
 - arguments 23
 - behavior 166
 - calling from C# 258
 - compilation 292
 - creating named 242
 - curried form 140
 - declaration 57
 - different setting 436
 - evolution in C# 133
 - from operator 78, 149
 - generalization 170
 - generic 87, 129
 - higher order 129–130
 - implicit conversions for
 - calls 367
 - instead of interface 41
 - as interface 132
 - joining 161
 - lambda notation 133
 - let binding 133
 - mathematical concept 31
 - mathematical notion 131
 - multiple arguments 137
 - name 235
 - nested 58, 99
 - optimizing 261–271
 - parameter syntax 97, 144
 - parameters with units 375
 - processing 160
 - pure 132
 - representing behavior 206
 - representing color filter 396
 - returning 136, 401
 - See also* currying
 - signature 57
 - similarity with interfaces 241
 - thinking using 40
 - time-varying values 429
 - type 77
 - type as a specification 45
 - type signature 133, 136
 - using as parameters 40
 - as value 130, 206
 - value in C# 76
 - value in F# 77
 - versioning 234
 - without arguments 57
 - See also* higher-order function
- function application
 - in lambda calculus 31
- function call
 - independence 299
 - nesting 261
 - reordering 299
 - tracking dependencies 390
- function composition 160, 435
- implementing 161
- in C# 162
- point-free style 211

function composition (*continued*)
 types 161
 using 161

function declaration
 inferring the type 164

function type 135
 equal 403
 relation with tuples 135
 turning to objects 240

function value 130–140
 defined 131
 encapsulated 455
 processing using HOFs 160
 refactoring 287
 representing
 computations 357
 wrapped 431

functional
 application design 418
 class 249
 collection 34
 design 177
 library 44
 reactive programming 471

functional C#
 option type 122

functional data structures 178
 implementing in C# 181

functional design
 thinking 178

functional languages 9
 fundamental features 4
 union type 115

functional libraries 420–459

functional list 165
 .NET support 410
 F# implementation 165
 immutability 34
See also list

functional list. *See* list

functional pattern
 lifting 436

functional program
 evaluation 32–39
 information
 representation 187
 parallelizing 383
 simplicity 190
See also computation

functional programming
 arguments 383
 closure 215
 composing functions 345
 defined 4
 DSL 423

foundations 31
 generic code 143
 influence 11
 lifting 436
 principles 6
 using HOFs in C# 159
 using mutable types 410

functional reactive animations.
See Fran project

functional skills
 understanding type 155

functional style
 in C# 159
 working with arrays 276

functional technique
 efficiency 260
 working with sequences 315

functional types 47

functionality
 adding 119
 localization 119

functions 55–60
 multiple 219
 using lists of 209
 as values 40, 75–79, 220

functions vs. interfaces 144

Future class 20

Future type 312

G

GAC 192

gallery of values 166

garbage collection 6

Gates, Bill 353

gathering information 372–378

generalizing code 406

generated comparisons 297

generating
 composed behaviors 221
 numbers 309
 random number
 generators 414
 reports 287

generating sequences 315–320
 implementing
 IEnumerable 315
 using higher-order
 functions 316
 using iterators 316
 using sequence
 expressions 317

generic 46
 C# 123
 F# 125

function 129
 higher-order functions 143
 list 73
 value 70, 122, 129–130

generic class
 initializing 63

generic code
 functional vs. object-
 oriented 143
 implicitly 165

generic functions 87
 in F# 144

generic method
 calling 163
 extensible code 143

generic parameter
 explicit 63

generic type 125
 declaration 122
 syntax options 126

generic type parameters
 units of measure 376

generic type signature
 understanding 149

geometrical transformation 399

get member 250

GetPixel method 278, 397

getter 249

Global Assembly Cache. *See* GAC

global mutable state
 logging 346
 using 480

global state 414
 functions 131

global value 55

Google MapReduce. *See*
 MapReduce algorithm

grammar rules 45

graphical designer 92

graphical editor 213, 219

graphical effect 395–408
 creating as lambda
 function 401
 creating from filters 401
 parallelizing 395
 representing 400

graphical filter 178, 205
 representation 206

graphical shapes
 representation 47

graphical user interface
See also GUI
See also UI

graphics. *See* drawings

grayscale filter 397

group clause 329
 grouping operation 329
 guard
 in match expression 49
 GUI 82, 460
 applications 92
 creating in F# 400
 thread 412, 475, 479
 See also UI
 guidelines
 using tuples 112

H

Handler delegate 471
 handler. *See* event handler
 hard-to-use type 241
 hash
 code 244
 type 289, 331
 Haskell 6, 39
 animations 428
 evaluation strategy 301
 infinite lists 324
 lazy lists 308
 monads 334
 working with state 346
 head 70
 head (list) 86
 headache
 mutable value type 396
 heap allocation 396
 Hello world 22
 HelloWorld class 26
 hexadecimal format 91
 hidden class 243
 hiding
 mutation 277
 value 117
 hiding implementation
 computation type 339
 hierarchical data 191
 higher-order function
 41, 129–176
 array processing 276
 in C# 150
 combining 210
 Count method 209
 for discriminated union 151
 for documents 195
 encapsulation 168
 event processing 462
 in F# 144
 for functions 160–163

generating sequences 315
 generics 143
 implementing interfaces 441
 in lambda calculus 32
 measuring time 403
 nesting 156
 for options 158–159
 processing sequences 327
 recurring pattern 165
 refactoring 288
 for schedule values 151–154
 signature in C# 150
 simplified processing 154
 strategy pattern 212
 for tuples 147–151
 two-dimensional arrays 397
 type inference 163–165
 type signature 136, 195
 understanding type 166
 using from C# 258
 working with arrays 279
 wrapping code 185
 HOF. *See* higher-order function
 hole in the middle pattern 184
 Hollywood Principle 461
 host language
 extending 423
 HTML format 187
 HTTP communication 355
 HttpUtility class 364
 Hudak, Paul 428
 hundreds of tasks 358
 Hurt, Brian 185

I

IComparable interface 296
 ideal representation 187
 identifying commonality 221
 identity
 function 283
 monad 345
 IDEs 286
 idiomatic .NET solution 243
 idioms
 working with functions 211
 IDisposable interface 245
 using object expressions 246
 IEnumerable type 288
 IEnumerable<T>
 using in immutable types 410
 IEnumerable<T> interface
 167, 317
 implementing 315
 IEqualityComparer
 interface 244
 IEvent type 462
 if condition 300
 if expression
 using pattern matching 67
 if-then-else
 expression 37
 type inference 128
 IL 217
 image
 applying filter 397
 blurring 278
 processing 205
 imbalanced tree 280
 immediate testing. *See* interactive testing
 immutability 33–34
 in C# 181
 consequences 18
 data structures 33
 declarative syntax 427
 record type 180
 tracking code
 dependencies 289
 understanding code 17
 values 33
 immutable
 array 398
 data structures 12
 value 23
 immutable class
 creating 18
 implementing 181
 in C# 63
 initialization 95
 immutable data structure 60–68
 calculating with 64
 parallel processing 391
 parallelism 383
 suitable operations 274
 immutable data type
 implementing in C# 181
 members 237
 using arrays 277
 immutable list. *See* list
 immutable type
 changing values 181
 comparing 296
 syntax 426
 value or reference 108
 immutable value 33
 initializing 226
 immutable variables. *See* immutable value

- imperative
 - class 249
 - configuring objects 427
 - for loop 152
 - islands 385–386
 - languages 8
 - solution 209
- imperative program
 - executes 37
- imperative programming
 - array 275
 - function 131
- imperative style
 - scalability limitations 383
- imperial units of measure 51
- implementation
 - inheritance 241
 - level 11
 - parallel 384
 - sequential 384
- implicit
 - class syntax 253
 - constructor 248
- implicit conversion
 - in F# 254
 - to base class 367
- implicit type conversions 366
- implicitly generic code 165
- in keyword 56
- in-memory bitmap 184
- in-memory processing 369
- incompatible type 62
 - correcting code 345
- incomplete pattern 67
 - match 72, 197
 - matching 50
- indentation 128, 235
 - style 323
- independent
 - code blocks 390
 - iterations 385
 - lines 19
- independent components
 - animation library 429
- index
 - in projection 190
- indexer 85
- indicator
 - obtaining 371
- infinite
 - number of elements 314
 - recursion 322
- infinite data structure 309
 - sequence 322
- infinite list 308, 324
 - aggregating 310
 - of colors 308
 - of integers 309
 - projections 310
- infinite loop 69
 - asynchronous 475
- infinite sequence 314, 322–324
 - of colors 322
- InfiniteInts type 308
- infinity 309
- infix notation 145
- informal specification 223
 - transcription 188
- inherit keyword 447
- initialization
 - encapsulating 184
 - recursive values 226
 - soundness 269
 - values 294
 - of a variable 33
- input
 - array 385
 - classification 223
- instance members
 - calling 85
- instantiation
 - type inference 128
- int function 88
- Int32.Parse method 84
- integer 88
 - operator 35
- intelligent behavior 411
- IntelliSense 235, 256
 - computation builder
 - value 356
 - for F# objects 237
- interactive
 - execution 234
 - programming 23
 - shell 292
- interactive development
 - drawing documents 186
 - using mutation 447
- interactive shell 353
- INTERACTIVE symbol 293
- interactive test 293
 - evolving 293
- interactive testing 82, 364
 - asynchronous workflows 358
- interface
 - C# 222
 - declaring in F# 241
 - drawings in F# 440
 - explicit implementation 251
 - in F# 241–243
 - as function 132
 - implementing 242
 - implementing in a class 251
 - implementing using object
 - expressions 243
 - keyword 253
 - single method 207
 - type 222
 - using Func instead 207
 - writing extensible code 143
- interfaces
 - using from F# 243–248
- interference 300
- interleaving order 327
- intermediate language. *See* IL
- internal representation
 - hiding 430
- interpretation
 - of type signature 161
- Into keyword 469
- IntOption type 122
- intrinsic type extension 239
 - adding operations 411
 - See also* type extension
- invalid
 - state 182
 - value 108
- invariant culture 376
- inversion of control 461
- Invoke method 259, 412
- is operator 254
- it value 181
- iteration
 - local state 405
- iterative development 82
 - adding unit tests 92
 - creating project 92
 - interactive testing 82
- iterative development process
 - optimization 270
- iterative development style 235
- iterator
 - asynchronous
 - programming 362
 - local state 317

J

- JavaScript 8, 44, 255
- JIT 264
 - tail recursion 264
- join
 - clause 333
 - operation 330, 335

joining sequences 319
 Joins Concurrency Library 494
 just-in-time. *See* JIT

K

keeping values alive 234
 Kennedy, Andrew 376
 key
 customizing equality 243
 key data structure 219
 key-based access 244
 keyword
 and 225
 do! 348
 event 470
 for 336, 389
 fun 133
 in 56
 inherit 447
 interface 253
 Into 469
 let 55
 let rec 320
 let! 336–337, 356, 475
 lock (C#) 414
 match 71
 module 399
 mutable 250
 namespace 256
 rec 69, 225
 return 135
 return! 356, 478
 struct 10, 108, 395
 this 236
 type 115, 225, 374
 use 245–246
 using 245
 val 414
 var 63, 127
 with 180, 182, 243, 447
 yield 332, 369, 389
 yield break 317
 yield return 317
 yield! 319, 389
 kilometers per hour 52
 Kleene, Stephen C. 6
 km² 375

L

labeled tuple 179
 lambda
 Greek letter 31, 131
 object 223

lambda calculus 6, 31–32, 131
 typed 6
 lambda expression 77, 131
 closures 218
 See also lambda function
 lambda function 131–135
 as a code block 185
 conversion to delegate 132
 explicit 211
 function as parameter 209
 as a handler 312
 implementing operation 144
 laziness 304
 multiple arguments 133
 nested 138
 parallel for 386
 parameter type 134
 returning 401
 running on a thread 412
 statement block 135
 strategy pattern 212
 writing tests 207
 See also object expression
 lambda objects 242
 language
 built-in construct 386
 changing the meaning 423
 common 173
 creating animations 423
 extending 423
 multiparadigm 233
 specific domain 423
 Language Integrated Query. *See* LINQ
 language-oriented
 programming 43, 423
 languages
 See also functional languages
 See also imperative languages
 See also object-oriented
 languages
 large amount of data
 tail recursion 273
 large collections 271–279
 large data sets 260
 processing in parallel 390
 largest value 79
 last element 274
 last recursive call
 jumping out 263
 layer 99
 layout
 adaptive 196
 calculation 190
 Lazy class 394

lazy evaluation 302
 in Haskell 39
 I/O and user interface 301
 using functions 303
 using lazy values 304
 lazy keyword 304
 lazy list
 accessing values 309
 using sequences instead 322
 Lazy type 304, 306
 lazy value
 computation builder 350
 in C# 306
 in F# 304
 practical uses 307
 primitive functions 305
 primitive operations 394
 relation with tasks 394
 resized photos 310
 LazyCell discriminator 308
 LazyList type 308
 leaf 280
 counting primes 391
 length
 comparing 298
 let
 clause 340
 keyword 55
 let binding
 declaring functions 57
 factoring expressions 427
 inside classes 253
 nesting 58
 let rec ... and construct 226
 let rec binding 69
 let rec keyword 86, 320
 let! keyword 337
 asynchronous operation 356
 library
 compositional way 423
 functional 420–459
 immutable type 251
 loading dynamically 208
 shared 114
 lifting 436
 drawings to animations 448
 method 438
 nullable types 436
 numeric operators 451
 translation function 448
 using Behavior.map 437
 #light directive 56
 lightning filter 397
 lightweight process 480

- lightweight syntax 57, 128, 210
 - turning off 56
- line break 57
- LINQ 13, 110, 134
 - anonymous types 110
 - asynchronous
 - programming 362
 - collection processing 168
 - Count operator 209
 - creating animated values 440
 - expression trees 119
 - extension methods 42
 - goal 15
 - language-oriented
 - programming 423
 - method chaining 426
 - processing events 468
 - query operators 329
 - Select operator 279
 - standard terminology 438
 - in Visual Basic 469
 - working with behaviors 440
 - working with events 468
 - and XAML 12
- LINQ query
 - for option values 344
- LINQ to Objects 278, 288
 - implementing 326
- LINQ to SQL 315
 - language manipulation 423
- LINQ to XML 191, 366–369
- LISP 6, 44, 421
- List
 - module 387
 - type 290
- list 68–75
 - adding and removing 70
 - aggregating elements 76
 - of all integers 322
 - alternative value 166
 - appending elements 273
 - calculating sum 86
 - cell 308
 - comparison 298
 - creating in F# 70
 - decomposing 71, 87
 - diagram 69
 - efficient usage 273
 - empty 69
 - expression 321
 - F# implementation 165
 - filtering 287
 - of functions 206
 - generating 321
 - head 70
 - higher-order functions 131
 - hundreds of thousands
 - elements 262
 - immutability 70
 - implementing HOFs 170
 - implementing in C# 72
 - infinite 308
 - iterating over elements 184
 - of prime numbers 308
 - of operations 235
 - pattern matching 71
 - pipelining 146
 - processing 74, 288
 - processing using tail
 - recursion 271
 - representing documents 182
 - reverse 272
 - sorting 287
 - suitable operations 274
 - summing elements in C# 74
 - summing elements in F# 75
 - tail 70
 - using efficiently 271
 - using partial function
 - application 139
 - working with 42, 165–175
- list (LISP) 422
- list of functions 209
- list processing 130, 165–175
 - abstracting 76
 - composability 421
 - domain-specific language 423
 - example 167
 - filtering 166
 - language 43
 - operation order 169
 - parallelizing 387
 - projecting 166
 - schedule example 152
 - similarity with events 462
 - using function
 - composition 161
 - using pipeline 146
- list type
 - in F# 165
- List.average function 407
 - average color 407
- List.bind. *See* List.collect
- List.collect function 175
- List.concat function 190
- List.filter function 42
- List.fold function 171
- List.forall function 197
- List.hd function 146
- List.init function 265
- List.map function 43, 139
- List.mapi function 190
- List.ofseq function 84
- List.partition function 298
- List.rev function 146, 272
- List.sortBy function 288
- List<T> class 167
- literal collection
 - expressions 422
- little bit of code 354
- #load directive 388
- loading behaviors 208
- loan
 - testing using collection 208
- loan suitability
 - decision tree 224
- loan test
 - adding reporting 219
 - configurable 214
 - creating with reporting 220
 - default set in C# 207
 - default set in F# 210
 - executing 208
 - similar structure 221
 - using decision tree 223
 - using interfaces 242
- local function
 - additional argument 392
 - lambda 416
- local value 109
- localization
 - of functionality 119
 - of representation
 - processing 119
- location
 - at specified time 428
 - specifying 443
 - working with 414
- lock function (F#) 414
- lock keyword (C#) 414
- lock-free code 11
- LockBits method 397
- locking
 - minimizing 384
 - shared objects 413
- locks
 - minimizing 414
- log messages 346
- logger
 - state 346
- logging 346
 - primitive function 347
- logging computation 346
 - using 347

- logic
 - expressed explicitly 142
- logical operators 145
- loop
 - busy 411
 - using recursion 35
- lower-level
 - parallel programming 391
- M**

- machine learning 223
- mailbox
 - accessing concurrently 492
 - ignoring messages 492
 - Receive method 482
 - Scan method 492
- mailbox processor 480–493
 - AsyncPostAndReply method 483
 - communicating 490
 - concurrency 483
 - creating 481
 - encapsulating 487
 - encoding state machine 491
 - implementing 482
 - Post method 483
 - PostAndReply method 483
 - Receive method 484
 - Scan method 484
- MailboxProcessor
 - members 483
 - type 482
- main
 - application 399
 - window 93
- maintainability 222
- map function
 - tail recursive
 - implementation 271
- Map method 345
 - See also* Select method
- map operation 148
 - for behavior values 435
 - for behaviors in C# 438
 - for documents 194
 - for events 462, 464
 - merging document parts 196
 - for options 155
 - for schedule 151
 - for tuple 148
- map task 22
- Map type 244
- Map.contains function 377
- Map.ofSeq function 376
- mapFirst function 148
- MapReduce algorithm 22
- mapSecond function 148
- Marshal class 397
- massive amount of data 354
- match
 - construct 66
 - expression 49, 67
- match construct 84, 366
 - C# analogy 118
 - nesting 154
- match keyword
 - processing lists 71
- MatchFailureException
 - exception 72
- MatchNone method 124
- MatchSome method 124
- Math.Pow method 414
- Math.Sqrt method 414
- mathematical definition
 - correspondence 68
- mathematical operators 31, 145
- mathematical purity
 - in Haskell 37, 39
- mathematics
 - function 131
 - monad 337
 - relation 131
- mature code
 - using members 240
- max function 79
- MaxBy extension method 417
- maximum
 - finding 79
- maybe monad 345
- McCarthy, John 6
- meaning
 - of the code 315
 - preservation 300
 - of symbols 422
- Measure attribute 374
- MeasureString method 101
- MeasureTime method 404
- measuring the speedup 388
- member
 - adding to F# types 235
 - call syntax 236
 - calling 236
 - declaration 235
 - keyword 236
 - See also* abstract keyword
- memoization 266–271
 - in C# 267
 - first call 269
 - of recursive functions 269
 - reusable function 267
- memoize function 268
- memory
 - continuous block 275
 - limitations 308
- merging
 - events 465, 467
 - routines 287
- message
 - communicating 483
 - queueing 483
 - replying to 481
 - skipping 492
 - type 481
 - unprocessed 484
 - See also* mailbox, Scan method
- message passing 481
 - concurrency 490
 - encapsulation 487
- message type 491
- method
 - adding 145
 - converting to delegates 397
 - declaring 132
 - in F# 248
 - group 402
 - overriding in F# 447
 - See also* member
- method call
 - type inference 73
- method chain 426
- parallel query 388
- method chaining 182
- method signature 129
 - adapting 402
 - See also* currying
- metric system 51
- micro-optimization 273
- Microsoft Excel 378–381
- Microsoft Research. *See* MSR
- Microsoft Robotics studio 362
- miles per hour 52
- Milner, Robin 6
- minimum
 - finding 79
- mistake
 - avoiding using units 375
- mixing concepts 227
- ML language 6
- modeling
 - financial contract 454
- modeling language
 - using 457

- modern programming
 - reusability 143
- modifying
 - list 289
 - unfamiliar programs 285
- module 258
 - automatic naming 292
 - compilation 292
 - keyword 399
 - unit tests 295
- monad 334–350
 - bind and return 341
 - bind operation 337
 - identity 345
 - maybe 345
 - return operation 338
 - unwrapping value 340
- monadic type 337
 - See also* computation type
- monadic value binding 337
- monads. *See* computation
 - expressions
- Mondrian, Piet 486
- Monitor class 414
- mouse clicks
 - counting 475
- mouse location 469
- MouseMove event
 - handling 477
- movement
 - composing 427
- MoveNext method 315
- MSR 7, 94, 384
- multicore processor 6, 383
- multiline
 - commands 25
 - comment 145
- multiparadigm 233
 - language 6
- multiple
 - arguments 97, 144
 - data sources 318
 - data structures 178
 - functions 219
 - output elements 330
 - sources 327
- multiple elements
 - returning 330
- multiple representations
 - transition 223
- multiple threads
 - communicating 492
- multiple values 109–114
 - higher-order functions 147
 - returning 109

- multithreading 10
- mutable
 - by default 218
 - class 108
 - field 249
 - list 299
 - property 216
- mutable data structure
 - array 275
 - problems 290
- mutable keyword
 - class declaration 250
- mutable objects
 - method chaining 427
- mutable state
 - avoiding in classes 249
 - documenting 215
 - handling events 461
 - hiding 214
 - modifying 299
 - reference cells 216
 - top-level 412
 - user interface 466
- mutable types 34
 - GUI interaction 401
- mutable value 59–60
 - in F# 39
 - limitations 216
- mutation
 - accidental 290
 - hiding in C# 279
 - initialization 226
 - using interactively 447
 - using internally 277
- mutually recursive
 - functions 269
 - types 224
 - values 225

N

- name
 - find longest 290
 - find multiword 290
- named
 - arguments 95
 - class 242
 - function 242
- namespace
 - keyword 256
 - opening 84
- naming convention 258
 - interface in F# 441
 - interfaces 241
- NASA Climate Orbiter 51
- nested
 - document parts 187
 - function 99
 - function calls 261
 - function declaration 58
 - pattern 50, 87
 - sequence processing 332
- nested element
 - finding 367
- nested for loops
 - parallelization 405
- nesting tuples 112
- .NET
 - attributes 15
 - class library 208
 - coding style 442
 - enumeration 462
 - events 461
 - exceptions 84
 - generics 6
 - integration 233
 - object model 26
 - programming guidelines 222
 - thread pool 358
 - See also* interfaces
- .NET 2.0
 - handling stack overflow 261
- .NET 3.0
 - WPF part of 92
- .NET 3.5
 - Func delegate 77
- .NET 4.0 20
 - Parallel Extensions to .NET 384
 - Zip method 326
- .NET APM. *See* asynchronous
 - programming model
- .NET applications
 - developing in F# 81
- .NET collections
 - using from F# 244
- .NET languages
 - interoperability 255
- .NET libraries
 - callable from C# 241
 - mutable types 410
- .NET object model 222
 - features 249
- .NET Reactive Framework 468
- .NET types
 - interoperation 120
 - null value 120
 - thread-safety 405

- network
 - communication 481
 - operation 365
 - traffic flow 429
 - new keyword
 - optional 152
 - newly calculated value 317
 - newton 376
 - Nil discriminator 166
 - nil list 69
 - NodeType property 119
 - None discriminator 120
 - nonempty log 347
 - nongeneric utility class 73
 - nonparallel version 392
 - nonstandard behavior 350
 - notation
 - See also* type constructor
 - null value 120, 294
 - checking 120
 - type inference 127
 - Nullable type 121
 - nullable type
 - lifting 436
 - nullable types (C# 2.0) 121
 - nullable values
 - addition 437
 - NullReferenceException
 - exception 109, 120
 - number
 - converting 87
 - of failing tests 221
 - floating-point 88
 - literals 87
 - parsing 111
 - of requests 369
 - numeric
 - conversions 88
 - range 40
 - types 87
 - numeric literals
 - extension methods 432
- O**
-
- O notation 274
 - object
 - equality 296
 - identity 296
 - keeping state 487
 - state 177
 - object expression 223, 242
 - complex code 251
 - drawing objects 441
 - key equality 244
 - lambda function analogy 242
 - using with .NET types 243
 - object initializers
 - composing values 424
 - object initializers (C# 3.0) 94
 - object tree 424
 - object type 233
 - inheritance in F# 447
 - object-oriented
 - design 11, 177
 - features 248
 - and functional 249
 - ideas 233
 - languages 8
 - organization 237
 - refactoring 286
 - representing behavior 207
 - object-oriented languages
 - method chaining 427
 - object-oriented programming.
 - See* OOP
 - object, notion of 8
 - Object.Equals 296
 - Object.ReferenceEquals
 - method 297
 - observation
 - evaluation of lazy values 305
 - obvious code 260
 - OCaml 6
 - generic types 126
 - string libraries 90
 - syntax 126
 - See also* records of functions
 - Office 2007 381
 - online rectangle drawing 480
 - OnPaint method 447
 - OOP 8, 30
 - fluent interface 428
 - generic code 143
 - representing alternatives 114
 - open directive 26
 - OpenFileDialog class 102
 - operation
 - adding 114
 - adding in OOP 202
 - belonging to a type 235
 - composing 160
 - for composition 420
 - designing 177, 194, 411
 - discoverability 235
 - failing 346
 - recurring 184
 - reordering 300
 - separated 234
 - sequencing 426
 - specifying 143
 - structure 148
 - operator 35, 57
 - custom 145
 - as function 78
 - implementing or 303
 - lifting for behaviors 437
 - overloading 395
 - provided by a type 396
 - using as function 149
 - operator (Haskell) 325
 - operators as functions
 - point-free style 211
 - optimization 300
 - functional techniques 260
 - using lazy evaluation 39
 - optimizing functions 261–271
 - Option class 123
 - declaration 123
 - pattern matching 124
 - Option module 154
 - Option type
 - supporting queries 344
 - option type 109, 120–127, 294
 - in C# 122
 - extracting value 129
 - nested pattern matching 154
 - nongeneric version 122
 - processing using HOFs 154
 - processing using HOFs in C# 159
 - processing using queries 335
 - option value
 - implementing computation builder 343
 - processing 170
 - Option.bind function
 - implementing 158
 - Option.map function 155
 - behavior analogy 435
 - implementing 158
 - or-pattern 199
 - order
 - of arguments 163
 - of magnitude 275
 - of parameters 152
 - of statements 19
 - of operations 169
 - recursive processing 198
 - orderby clause 329
 - out parameters
 - calling from F# 111
 - outer loop 331
 - output array 385

- output elements
 - multiple 330
 - Output Type 92
 - overloaded
 - constructor 249
 - delegate 133
 - overloaded operator 396
 - for behaviors 450
 - for vector 410
 - working with vectors 415
 - overloading
 - number of type parameters 64
 - overridden method 447
- P**
-
- paged data 370
 - paradigm
 - combining 8, 249
 - functional 7
 - parallel
 - array processing 405
 - combining workflows 358
 - execution 10
 - programming 20
 - running asynchronous workflow 357
 - Parallel Extensions to .NET 20, 384
 - Parallel LINQ. *See* PLINQ
 - Parallel module 387
 - Parallel.For method 404
 - Parallel.ForEach method 386
 - Parallel.ofSeq function 388
 - parallelism 383–419
 - amount of 371
 - downloading data 371
 - important concepts 383
 - independent code 20
 - state 480
 - parallelization 12
 - bearing in mind 396
 - image processing 404
 - techniques 384–395
 - parallelizing
 - animal movement 415
 - simulation 417
 - ParallelQuery 388
 - parameter
 - name 57
 - specifying units 375
 - parameterization 76
 - parameterizing functions 143
 - using functions 288
- parameterized function
 - benefits 79
 - parameterized task 212
 - parameterizing functions 287
 - parameterless constructor 93
 - parameters
 - comma separated 97, 144
 - order 150
 - refactoring 288
 - space separated 97, 144, 442
 - using tuples 236
 - params keyword (C#) 84
 - parentheses
 - avoiding 145
 - Parsec library 426
 - parser combinators 426
 - parsing numbers 111
 - parsing values 372
 - partial function application 139
 - composition operator 161
 - creating graphical effects 402
 - document drawing 186
 - filtering collections 289
 - point-free style 211
 - schedule example 152
 - working with behaviors 436
 - partitioning 298
 - Pascal case 258
 - pattern 65
 - asynchronous methods 361
 - command 213
 - composite 200
 - decorator 201
 - drawing primitives 443
 - explorative development 369
 - hole in the middle 184
 - lifting 436
 - list processing 99
 - in match expression 49
 - multiple 67
 - processing composed values 143
 - processing events and lists 463
 - recursive function 69
 - recursive workflow 476
 - repeated 5
 - strategy 212
 - template method 228
 - temporary state change 247
 - underscore 66
 - using immutable types 64
 - visitor 202
 - with conditions 49
 - WithXyz method 250
 - See also* recursion pattern
 - pattern matching 48, 55
 - active patterns 201
 - Boolean flags 192
 - checking conditions 197
 - complete 121
 - complete pattern 67
 - decomposing lists 71
 - decomposing tuples 65
 - discriminated union 116
 - discriminated unions 308
 - on events 489
 - exceptions 366
 - extracting values 75
 - for ... in construct 90
 - function parameters 97
 - incomplete 197
 - incomplete pattern 67
 - lists 84, 86, 295
 - mimicking in C# 124
 - nested 87
 - or-pattern 199
 - order of patterns 99
 - on parameters 197
 - single-case discriminated union 338, 342
 - testing threshold 392
 - tuples 87
 - and type inference 164
 - underscore 87
 - using visitor pattern 203
 - per-pixel
 - filter 396
 - processing 398
 - performance
 - asynchronous operations 354
 - measuring 389
 - phase
 - application design 177
 - operation design 177
 - sinusoidal movement 452
 - photo browser 310
 - physical units 375
 - PIA 378
 - pie chart 82, 308
 - calculating angle 99
 - drawing 99
 - drawing labels 101
 - drawing segments 96
 - pipe 168
 - pipeline
 - array processing 278
 - event processing 466

- pipelining 43
 - carried values of events 466
 - list processing example 168
 - processing tuples 149
 - schedule example 152
 - using dot notation 147
 - pipelining operator 43, 146, 168, 211
 - composition 421
 - reading time-varying values 434
 - supporting 152
 - syntax 144
 - type inference 164
 - PLINQ 20, 384
 - using method calls 386
 - point-free style 211
 - examples 211
 - function composition 211
 - operators as functions 211
 - partial function
 - application 211
 - population
 - representing using tuple 61
 - PostAndReply method 487
 - PostSharp 44
 - PowerPack 308
 - library 85
 - PowerThreading library 362
 - pragmatic choice
 - mutable state 480
 - pragmatism 215
 - predicate 131
 - specifying 131
 - prepending an element 272
 - primary concern
 - behavior 178
 - data 178
 - Primary Interop Assemblies. *See* PIA
 - primes
 - counting 386
 - primitive
 - adding 425
 - animation in LISP 422
 - components 200
 - composing 420
 - creating rotation 452
 - custom computations 339
 - defining in LISP 421
 - drawings 442
 - execution specification 386
 - logging a message 347
 - moving drawings 443
 - type 108
 - workflows 361
 - primitive behavior
 - squaring 435
 - primitive functions
 - custom computations 339
 - primitive operation
 - as an argument 212
 - of lazy values 305
 - printf function 23, 90
 - format specifiers 91
 - printing output 89
 - private
 - fields 249
 - utilities 238
 - private setter
 - immutability 73
 - process
 - inside application 480
 - is terminated... 262
 - UI component 479
 - processing
 - language 173
 - messages 482
 - processing sequences 325–334
 - filtering and projection 327
 - using higher-order functions 327
 - using iterators 326
 - productivity 20
 - program
 - correctness 261
 - as an expression 31, 36
 - See also* behavior-centric program
 - See also* data-centric program
 - program data
 - representation 178
 - program state. *See* state
 - programming
 - See also* asynchronous programming
 - See also* data-driven programming
 - programming style
 - point-free 211
 - programming techniques
 - mimicking 44
 - project properties 92
 - projection
 - with index 190
 - projection operation 148
 - for documents 194
 - See also* flattening projection
 - projection operation. *See* map operation
 - property
 - accessing 85
 - in F# 248
 - initializing 93
 - of type Func 431
 - See also* member
 - property declaration
 - C# 250
 - prototype 222
 - prototype-based object
 - systems 44
 - pseq computation builder 390
 - public
 - API 242
 - composition 201
 - publishing events in F# 470
 - pure function
 - lazy values 305
 - optimization 266
 - purely functional
 - class declarations 249
 - members 237
 - Python 255
- ## Q
-
- queries 328–334
 - query 110
 - asynchronous
 - programming 362
 - events in Visual Basic 469
 - in LINQ 21
 - method chaining 426
 - processing option values 335
 - returning multiple values 110
 - translation 333
 - writing behaviors 440
 - query expression 13, 169, 328
 - customizing 335
 - internals 328
 - meaning 328
 - validity 328
 - working with events 468
 - Query Generator 363
 - query operators 329
 - for value computations 342
 - question mark symbol 254
 - queue
 - mailbox messages 491
 - of messages 483
 - quotations 423

R

- #r directive 192, 293, 356
- race condition 10
 - avoiding 418
 - avoiding using message passing 483
- radian (angle) 101
- raising an event 471
- random
 - color 95
 - tree 391
- Random class
 - thread-safety 413
- random numbers
 - mutable state 131
- randomBrush
 - function with side effects 96
- randomly generated array 278
- reactive animations 471
- reactive application 358, 466
 - storing state 480
 - understanding 474
- Reactive LINQ 468
- reactive programming 460–494
- Reactive.Attach method 468
- read-eval-print loop. *See* REPL
- read-only property 249
- readability 285
 - animation in LISP 422
 - list processing 168
 - point-free style 211
 - tupled parameters 97
 - tuples 113
 - using extension methods 146
 - using infinite sequences 322
 - using records 219
 - wrapping functions 399
- readonly modifier 18, 34, 63, 181
- real-world
 - creating animations 428
 - discriminated unions 119
 - document processing
 - example 182–204
 - drawing with infinite sequences 324
 - financial modeling 454
 - first application 81
 - graphical effects 395
 - logging 347
 - parallelization 395
 - simulation 409
 - testing loan suitability 206
 - time-varying values 429
 - using F# members 240
 - XML processing 191
- reasoning
 - about code 6, 37
 - point-free style 211
- rec keyword 69, 225
- recompilation
 - avoiding 118
- recomputing a value 266
- record type 179
 - adding members 236
 - changing value 180
 - cloning 180
 - representing clients 209
 - storing functions 219
 - succinct syntax 210
- records
 - accessing elements 181
 - combining data and behavior 225
 - per page 365
 - structural equality 296
- records of functions 219, 240
 - converting to interfaces 241
- Rect type 179
- rectangle
 - deflating 180
- rectangle drawing 477–490
 - online 480
- RectangleF class 180
- recurring pattern 87, 165
 - value processing 173
- recurring task
 - decomposing values 143
- recursion 34, 68–75, 261
 - asynchronous workflow 475
 - asynchronous workflows 365
 - busy loop 412
 - data type 68
 - depth 280
 - encoding loops 35
 - hiding 39
 - infinite list processing 310
 - keeping state 189
 - limitations 261
 - order 198
 - processing XML 192
 - returning result 262
 - termination 262
 - transformation 189
 - tree processing 280
 - types 166
 - using explicitly 39
 - See also* tail recursion
- recursion pattern
 - working with lists 87
- recursive
 - let bindings 226
 - sequence expressions 320
 - type 166
 - workflow 475
- recursive call
 - before and after 263
 - continuation argument 281
 - inside sequence
 - expression 320
 - jumping out 263
 - operation before 264
 - summing list 74
 - work after 263
- recursive discriminated
 - union 187
 - composite 200
- recursive function 69
 - asynchronous 482
 - list processing 86
 - processing messages 482
 - rewriting 263
 - stack overflow 260
- recursive part
 - hiding 75
 - reusing 76
- recursive processing
 - parallel 391
- recursive reference
 - run-time checking 322
- recursive value
 - compilation error 226
 - invalid code 270
 - memoization 270
 - self-reference 270
- reduce task 22
- ref
 - function 216
 - type 216
- refactoring 19, 39, 286
 - common behaviors 221
 - errors 292
 - evaluation order 300
 - lazy values 311
 - using computation
 - expressions 349
 - using functions in C# 289
 - valid 292
- reference
 - equality 297
 - type 108
- reference cell 216
 - in C# 216
 - capturing 217

- region codes 368
- region information
 - exploring 368
- registering callback 462
- relation 131
- releasing a thread 358
- remote data source 363
- RemoveAll method 290–291
- removing duplication 221
- reordering operations 300
- repeated schedule 114
- repetition
 - avoiding 185
- repetition anti-pattern 87
- REPL 23
- replacing values 157
- reply
 - waiting for 483
- reply channel 483
 - using in a message 485
- reporting details 219
- representation
 - adding types 119
 - runnable function 422
 - tree-like data-structure 422
 - See also* document representation
- requests
 - number of 369
- required income
 - modifying 218
- resizing photos 310
- resources
 - cleaning 245
 - consuming 355
- rest of the computation 340
- Result property
 - blocking behavior 392
- retry attempts 365
- return keyword
 - inside lambda function 135
- return keyword (F#) 337
- Return member 341
- return operation 338
- return statement 36
- return! keyword 356
 - asynchronous return 356
 - recursive looping 478
- returning
 - a new instance 249
 - new value 64
- returning result
 - using continuations 282
- reusability 131
 - of control structures 43
 - event processing 466

- generic functions 143
- generic types 125
- reusable .NET libraries 241
- reusing instances 414
- reverse list 146
- RGB components 395
- Richter, Jeffrey 362
- right associativity 70
- robustness
 - accessing values 75
- root
 - decision tree 224
 - element 192
- rotate primitive 425
- rotation 452
- rotation speed 452
 - changing with clicks 473
- routine 131
- Ruby 255
- running threads 358
- runtime
 - problems 255
 - stack optimization 263
- runtime error
 - recursive reference 270

S

- SaveFileDialog class 102
- scalability 383
- scaling time 451
- Scan method 492
- scene
 - animation 429
- schedule
 - adding members 238
 - processing 151
 - processing using HOFs in C# 153
 - processing using HOFs in F# 151
- Schedule type 114
- ScheduleType enumeration 117
- Scheme 44
- scheme. *See* LISP
- scientific format 91
- scope 55
 - explicit specification 245
 - implicit specification 245
 - of mutable state 215
 - nested 58
 - specifying using use 246
- ScreenElement type 183
- scripts 293
- search function
 - arguments 109
- seed
 - random numbers 413
- select
 - clause 328
 - operator 14
- Select method 139, 168
 - for behaviors 438
 - for events 468
 - implementing 326
 - for option values 345
 - using with arrays 278
- Select operation
 - for value computations 342
- SelectMany method 175
 - implementing using bind 345
 - for option values 345
 - See also* flattening projection
- SelectMany operation
 - for value computation 342
- SelectMany operator
 - implementing 343
- self-explanatory code 399
- semicolon
 - separating properties 210
- separate class declaration 251
- separate line
 - record properties 210
- separation
 - algorithm parts 322
 - of concerns 323
 - of varying functionality 39
- separation of concerns
 - event handling 462
- separator 171, 210
- seq block 317
 - preceding code 319
- seq identifier 170, 317
- Seq module 193, 316, 327
- seq type 288
- Seq.cache function 325
- Seq.collect function
 - bind operator 334
 - importance 332
 - instance of monad 338
 - See also* flattening projection
- Seq.filter function 327
- Seq.fold function 367
- Seq.groupBy function 329
- Seq.hd function 193
- Seq.iter function 323
- Seq.map function 327
- Seq.maxBy function 415
- Seq.sortBy function 329
- Seq.take function 318

- Seq.unfold function 316
- Seq.zip function 323
- seq<'a> type 184, 316
- sequence 314
 - caching 324
 - end of 316
 - event streams 464
 - of factorials in C# 316
 - of factorials in F# 320
 - finite 314
 - generated dynamically 314
 - IEnumerable interface 314
 - mathematics 314
 - of operations 160
 - of tuples 323
 - of values 463
 - parallel 388
 - See also* generating sequences
 - See also* processing sequences
- sequence expression 153, 170
 - caching 325
 - composing 319
 - filtering 328
 - generalization 315
 - if ... then ... else 332
 - internals 328
 - introducing syntax 317
 - lists and arrays 321
 - nested for loops 330
 - optimization 329
 - parallel implementation 389
 - projection 328
 - recursive calls 320
 - side effects 318
 - type 318
 - unifying concepts 315
 - using flattening
 - projection 331
 - using recursion 320
 - value binding 318
 - yielding empty list 373
 - yielding list elements 319
- sequence expressions 317–324, 328–332
 - processing regions 369
- sequence processing
 - using queries 328
 - using sequence expressions 328
- sequencing
 - of expression 57
 - operations 332, 426
- sequential code
 - parallelizing 384
 - waiting for events 474
- set member 250
- SetPixel method 278, 397
- shape
 - functional representation 48
- shared library 114
 - modifying 118
- shared memory 384
- shared object
 - accessing safely 413
- shared processing functions 173
- shared state 299
 - tracking dependencies 390
- sharing code 95, 185
- shifting time 451
- short-circuiting behavior 303
- side effects
 - avoiding 291
 - in C# and F# 301
 - explicit 291
 - in Haskell 39
 - initialization 93
 - object-oriented code 237
 - order 301
 - random numbers 96
 - See also* pure function
- signature
 - See also* type signature
- Silverlight 378
- simplicity
 - corresponding representations 193
- simplification
 - mathematics 285
 - as refactoring 286
- simulation 408–418
 - designing operations 411
 - drawing operation 411
 - parallelizing 417
 - running 411
 - state 409
- simulation state
 - in C# 410
- single argument function 109
- single expression 14
 - event processing 465
- single method 207
 - interface 132, 213
- single stack frame 263
- single-threaded
 - application 480
 - GUI 479
- skeleton
 - application 413
- Sleep method 361
- slicing 376
- slot, immutable 60
- smoothing values 277
- snd function 61, 111
- solar system 452
 - declarative specification 427
 - running simulation 453
- solution
 - refinement 82
- Solution Explorer 92, 192
- solving problems
 - using existing library 44
- Some discriminator 120
- Sort method 41
- sorting a list 40
- source code
 - representing 119
- sources
 - multiple 327
- special case
 - using pattern matching 68
- specification
 - rewriting to code 224
 - transcription 188
 - using method chaining 427
- speedup 393
 - measuring 388
- sprintf function 91
- SQL 15, 131
- stable codebase 114
- stack
 - size limited 261
- stack frame 37, 261
 - dropping 263
 - never used 263
 - throw away 263
- stack limit 262
- stack overflow 69, 260
 - avoiding 261
 - list processing 271
 - tree processing 280
- stack space
 - using continuations 282
- StackOverflowException
 - exception 261
- standalone application 27, 312
- standard function
 - combining 210
- standard notation
 - asynchronous methods 488
- starting with simplicity 222
- StartNew method 392
- state
 - affected by an operation 9
 - capturing 214
 - changing 34

- state (*continued*)
 - changing with messages 481
 - encapsulating 487
 - in GUI applications 479
 - iterator code 317
 - keeping as function
 - parameter 482
 - of a logger 346
 - new 347
 - of object 9
 - reactive applications 480
 - restoring using use 246
 - tracing 37
 - working with safely 480
- state machine
 - as mailbox processor 482, 491
- state parameter
 - unfolding operation 316
- statement 35
 - in F# 318
 - sequencing 56
 - switch order 286
 - See* currently executing statement
- statement block 135
 - type inference 128
- STAThreadAttribute
 - attribute 94
- static class 146
- static method 145–146
 - accessing in F# 84
 - generic 64
- static property 259
- static typing 45
 - in a functional language 45
- statically typed language 45, 255
- statistics 377
 - sampling 433
- step
 - of an evaluation 37
- step-by-step
 - automatic generalization 164
 - computation 156
- Stopwatch class 403
- strategy
 - data selection 288
 - pattern 212
- streams 309
- strict typing 289
- String
 - immutable type 10
- string
 - alignment and padding 91
 - concatenating 172
 - concatenation 146, 197
 - convert array to list 84
 - formatting 90, 171
 - in F# 84
 - indexing 85
 - sprint function 91
- String class 84
- string comparison
 - ignoring spaces 244
- string type
 - immutability 34
- String.Concat method 85
- String.Format method 47
- String.Join method 85
- String.Split method 84
- StringBuilder class 172
- Struct attribute 395, 409
- struct keyword 10, 108, 395
- struct type (C) 179
- structural
 - comparison 268
 - equality 296
 - patterns 200
- structure
 - encapsulation 178
 - exposed 178
 - information 178
 - projection 148
 - revealing 114
 - of tuples 113
- structured document
 - representation 187–194
- structuring application 11
- subexpression
 - evaluation 157
- substitution
 - mathematics 286
 - as refactoring 286
 - in unit testing 286
- substring 192
- Substring method 34
- subtree
 - counting primes 391
- subtree processing
 - independent 391
- success flag 111
- succinct syntax 11
- suitable operations 274
- summing
 - event values 467
- summing list 261
 - in C# 74
 - in F# 75
 - using tail recursion 265
- SumNumbers method 40
- surface area 373
- switch keyword
 - pattern matching 118
 - using inside HOF 153
- switch statement 48
- symbol 31
 - defining meaning 422
 - in LISP 44
- symbol (LISP) 422
- Syme, Don 7
- synchronization
 - minimizing 384
- syntactic sugar
 - asynchronous workflows 356
 - creating lists 70
 - disposal 245
 - queries 468
- syntactic transformation
 - parallelizing for loops 404
- syntax
 - calling operators 149
 - creating libraries 423
 - creating objects 152
 - declaring immutable class 73
 - default 56
 - flexibility in LISP 44
 - higher-order functions 144
 - lists and symbols 422
 - member call 236
 - nested calls 146
 - OCaml-compatible 56
 - of LISP 421
 - significant whitespace 56
 - using discriminated unions 424
 - using extension members 433
 - using immutable types 426
 - working with collections 170
- system callback 358
- System namespace 84
- System.Collections.Generic namespace 244
- System.Drawing namespace 92, 180
- System.Drawing.dll
 - assembly 234
- System.IO namespace 90
- System.Numerics namespace 88
- System.Reflection namespace 208
- System.Threading namespace 385
- System.Web namespace 364
- System.Xml.Linq.dll 192

T

- 'T type 126
- Tag property 49, 114, 118
- tagged union. *See* discriminated union
- tail 70
- tail (list) 86
- tail call
 - when using continuations 282
- tail recursion 261, 263–266, 271–273
 - sequence expressions 321
 - tree processing 280
 - See also* accumulator argument
- target representation 193
- task
 - for one subtask 392
 - ideal number of 393
 - incomplete 392
 - limiting the number 392
 - map and reduce 22
 - optimizing number 405
 - parallelization 384
 - primitive operations 394
- Task class 392
- Task Parallel Library. *See* TPL
- task-based parallelism 11, 20, 390–395, 417
 - in C# 393
 - execution pattern 392
 - nonparallel version 392
 - overhead 392
 - speedup 393
- Task.Factory property 392
- TaskFactory class 392
- technical details
 - hiding 14
 - revealing 422
- technologies
 - declarative 13
- template method pattern 228
 - functional implementation 228
- temporarily changing
 - context 247
- termination of recursion 69
- test
 - data 87
 - request 364
- testing 292
 - complicated data structures 296
 - composition 300
 - imperative code 299
 - input 265
 - interactively 293
 - primitive pieces 300
 - side effect free functions 299
 - suitability 206
 - unit testing 294
 - using structural equality 298
- text labels 101
- TextContent type 183
- The Haskell School of Expression 37
- theory
 - infinite sequences 324
- thinking about
 - problems using functions 40
- this
 - keyword 236
 - modifier 146
 - referencing inside constructor 447
- thread
 - current 357
 - expensive creation 355
 - pool 357
 - program state 37
 - releasing 358
 - switching during workflow execution 360
 - used for callback 357
- thread-safety 405, 481
- Thread.Sleep method 361
- threading model 94
- threshold
 - choosing 393
 - parallelization 392
- throwing an exception 84
- tick-type. *See* apostrophe-type
- time
 - addition 451
 - measuring 403
 - multiplication 451
 - of an operation 389
 - shifting and scaling 451
- #time directive 273, 388
- time-dependent value 428
- time-varying
 - drawing 445
 - value 428
- time-varying value
 - See also* behavior
- Time.Current property 432
- Time.Forever method 432
- Time.Wiggle property 16, 432
- timer
 - waiting asynchronously 361
- ToArray method 278
- tools
 - data-oriented 353
- ToolStrip control 399
- ToolStripComboBox control 399
- TPL 384
- tracking dependencies. *See* code dependencies
- trade
 - evaluating 455
 - primitive 456
- trading
 - window 457
- traditional .NET 241
- transformation
 - between representations 178, 188
 - calculating information 189
 - function 187
 - of the iterator code 317
 - of a sequence 328
- transforming
 - document 190
 - sequences 326
- transition 222
 - GUI control states 478
 - to parallel version 389
 - from simple to complex types 108
- TranslateTransform
 - method 185, 443
- translating representations 190
- translation
 - transformation 443
 - variable animations 448
- traversal 198
- tree 119
 - balanced and imbalanced example 280
 - depth of processing 392
 - height 281
 - large 283
 - processing 279–283
 - processing using continuations 282
 - recursive processing in C# 393
 - summing values 279
 - See also* decision tree
- trigger event 470
- trigonometric functions 101
- try ... with block 366
- try-finally block 245
- TryParse method 88, 111

- tuple 60–68
 - calculating with 64
 - choosing representation 113
 - comparison 297
 - compatibility 113
 - complexity 113
 - compositionality 112
 - decomposing 65
 - in C# 61
 - incomplete pattern 67
 - nested 112, 138
 - of functions 217, 219
 - processing elements 148
 - processing using HOFs in C# 150
 - processing using HOFs in F# 149
 - reconstructing 147
 - structural equality 296
 - type 111
 - using interactively 366
 - using records instead 113
 - working with 61, 147, 149
- Tuple class 62
- Tuple class 150
 - value equality 268
- tuple type
 - relation with functions 135
- tupled parameters 97
 - in DSL 442
- tuples 109–114
 - decomposing 87
 - vs. records 113
 - See also* multiple values
- turning into
 - complex product 222
- two-dimensional array 380, 397
 - higher-order function 397
 - processing in F# 398
- # type 331
- type
 - adding 115
 - adding members 235
 - alias 98
 - in C# 45
 - complexity 108
 - complicated structure 142
 - composing 108, 420
 - composing functions 161
 - constraint mismatch 254
 - conversion 254
 - of a drawing function 98
 - enumeration 462
 - of an expression 45
 - extensibility 118
 - function 135
 - in functional languages 47
 - in functional programming 45
 - future extensibility 430
 - giving a name 430
 - as grammar rules 45
 - meaning of value 374
 - partially specified 129
 - recursive 166
 - recursive definition 224
 - recursive structure 68
 - scheme 112
 - specification 161, 165
 - specifying explicitly 63
 - of a value 109
 - See also* F# type
 - type ... and construct 225
 - type annotation
 - aggregation function 78
 - for arrays 277
 - function calls 163
 - for functions 134
 - not needed 210
 - and pipelining operator 164
 - record 181
 - two ways 129
 - unnecessary 405
 - type argument
 - automatic deduction 163
 - cannot be inferred 128
 - inference 73
 - specifying 128
 - type cast
 - in F# 253
 - type checking
 - invalid units of measure 375
 - type constructor
 - composing types 420
 - discriminated unions 115
 - function 135
 - tuples 111
 - what is 111
 - type declaration 115
 - generic 122
 - members 235
 - as specification 188
 - type definition
 - recursive 187
 - type extension
 - adding operators 451
 - for numeric type 433
 - intrinsic 239
 - type extensions 238–240
 - type inference 11, 24, 45, 127–129, 163–165
 - accessing records 210
 - algorithm 130
 - in C# 3.0 46
 - construction 63
 - discriminated union
 - creation 126
 - in F# 128
 - for function calls 163
 - for functions 144
 - for generic methods (C#) 124
 - for local variables 46
 - for loop 289
 - for method calls 163
 - generic method calls 64
 - insufficient information 128
 - missing in C# 63
 - order of arguments 163
 - from pattern matching 164
 - record type 179
 - summing list 75
 - tricky examples 129
 - tuple type 60
 - from type signature 164
 - using object expressions 245
 - for values 127
 - See also* automatic generalization
 - type inference algorithm
 - in C# 163
 - in F# 163
 - type information
 - loss 60
 - understanding 98
 - type keyword 115, 225
 - declaring interfaces 241
 - declaring units 374
 - type mismatch
 - units of measure 51
 - type parameter
 - automatic naming 129, 149
 - inferred 149
 - of generic method 129
 - substitution 402
 - type signature 57, 98
 - adapting 402
 - aggregation 78
 - analyzing 156
 - bind and return 341
 - bind operation 335
 - continuation passing style 283
 - creating graphical effects 402
 - deducing behavior 166
 - function 136

- type signature (*continued*)
 - generating behaviors 222
 - graphical form 78
 - higher-order functions 148
 - interface member 241
 - list processing functions 166
 - memoization 268
 - possible interpretations 161
 - processing schedules 152
 - projecting documents 195
 - reactive animations 472
 - reading 78
 - similarities 173
 - translation function 189
 - of tuples 113
 - understanding 155, 166
 - understanding bind 174
 - working with locations 414
 - See also* generic type signature
 - type system 108
 - compositionality 420
 - type-safety 125, 165
 - types 45–52
- U**
-
- UI 101
 - browsing photos 311
 - control flow 460
 - initialization 226
 - using asynchronous workflow 474
 - undefined
 - result 120
 - value 120, 344
 - underscore
 - pattern 66, 87
 - type 245
 - understanding code 300
 - using units of measure 375
 - unfold operation 316
 - unification
 - iterators and queries 326
 - uniformity
 - sequence processing 328
 - union
 - in the C language 48
 - unit of work 213
 - unit testing 6, 19, 292
 - composition 299
 - setup and teardown 292
 - unit tests
 - failure 296
 - pointless for immutable data 299
 - separating 295
 - unit type 37, 56, 95
 - units of measure 51–52, 373–377
 - ^ (power) 374
 - / (division) 374
 - simplification 375
 - unmanaged memory 397
 - unprocessed messages 484
 - unreachable
 - branch 197
 - code 153
 - unresponsiveness 354
 - untyped data format 363
 - unwrapping value 340
 - upcast 254
 - returned element to XContainer 368
 - updatable field 250
 - upper bound
 - inclusive 385
 - URL
 - building programatically 363
 - UrlEncode method 364
 - use keyword 245
 - programming 246
 - useful information 372
 - user interface interaction
 - threading 480
 - using keyword 245
 - utility function 58, 238
- V**
-
- val keyword 414
 - valid state transition 182
 - valid value 294
 - value 166
 - abstract description 242
 - accessibility 55
 - animated 428
 - available operations 143
 - binding 55
 - captured 215
 - complexity 108
 - composing 424
 - constructor 111
 - deconstructing 49
 - domain 111
 - hiding 117
 - instead of variables 34
 - missing 109, 294
 - mutable 59
 - not available immediately 463
 - pulling 317
 - recognizing 109
 - recomputing 266
 - recursion 225
 - recursive definition 225
 - scope 55
 - shared by closures 217
 - type 108
 - type as meaning 374
 - using from C# 258
 - valid 294
 - what is? 108
 - See also* animated value
 - See also* behavior
 - See also* immutable value
 - See also* value binding
 - value binding 23, 33
 - customized 337
 - inside sequence expression 318
 - monadic 337
 - value hiding 198
 - Value property
 - lazy values 304
 - value structure
 - abstracting 143
 - value type
 - immutability 396
 - implementing in F# 395
 - value vs. type 109
 - value wrapper computation 338
 - values 55–60
 - ensuring consistent use 45
 - values vs. variables 33
 - var keyword 46, 63, 127
 - variable
 - declaration 23
 - declaring new 33
 - initialization 33
 - mutating 33
 - variable reference
 - avoiding 111
 - VB.NET
 - integration 233
 - vector
 - calculating with 415
 - graphics 206
 - Vector type
 - implementing 409
 - vent
 - merging 465
 - verbosity 163
 - verification
 - partial 298
 - of systems 31
 - virtual method
 - adding 114, 119
 - visitor pattern 119, 202

Visual Basic
 event processing 469
 Visual Studio 83, 192, 286
 designers 400
 using F# in 47
See also F# Interactive
 Visual Studio 2010 4, 7
 Visual Studio project 92
 visualization 378–381
 declarative programs 466
 vocabulary
 extending 42
 of the computer 8
 void
 keyword (C#) 37
 type 135

W

waiting asynchronously 361
 walking down
 XML tree 368
 warning
 FS0025 72
 FS0040 269, 322
 incomplete pattern
 match 197
 web page
 downloading 354
 website
 photo browsing demo 310
 weighted average 397
 when clause 50, 197
 exception handling 366
 processing events 490
 WHERE clause 131
 where clause 328
 filtering events 469
 Where method 131, 168, 387
 for events 468
 implementing 326
 where operator 14
 while loop
 asynchronous GUI
 processes 479
 while(true) loop 322

whitespace
 syntax 56
 Windows application 92
 Windows Forms 92, 102,
 193, 311
 declarative event
 handling 466
 designer 399
 drawing shapes 477
 GUI thread 412
 in F# 400
 switch function 474
 thread-safety 412
 Windows Presentation Founda-
 tion. *See* WPF
 with keyword 180, 243, 447
 mimicking in C# 182
 WithXyz method 250
 WithXYZ methods 181
 work
See also computation
 worker thread 384, 394
 in the GUI 480
 workflow 334–350
See also asynchronous
 workflow
 working with lists 165–175
See also list processing
 working with values
 using functions 143
 worksheet
 creating 378
 world
 simulation 408
 World Bank 362–372
 downloading data 370
 registration 363
 World Development
 Indicators 363
 WPF 14, 92
 wrapper type
 configuring mutable
 objects 428
 wrapping
 code 240
 value 342
See also lazy keyword

X

XAML 14
 XContainer class 367
 XDocument class 193, 368
 XElement class 192, 367
 XML
 comment 236
 document representation 191
 following a path 368
 helper functions 366
 parsing attributes 191
 reading attributes 368
 reading values 368
 syntax (C) 94
 working with 191–194,
 366–369
 XML data
 exploring 368
 reading values 372
 XML element
 recursive processing 192
 XSLT 15
 xUnit.net framework 292

Y

yield
 elements 319
 multiple 390
 yield break keyword 317
 yield keyword 318
 replacing with flattening
 projection 332
 yield return keyword 317
 yield! keyword 319
 lazy evaluation 319
 tail-call optimization 321

Z

Zero member 346, 396
 list average 407
 Zip operation
 implementing in C# 326
 zip operation 323