

FLEX3 IN ACTION

Tariq Ahmed

with Jon Hirschi and Faisal Abid

FOREWORD BY Ryan Stewart



contents

foreword xxi
preface xxiii
acknowledgments xxv
about this book xxvii

PART 1 APPLICATION BASICS..... 1

- 1** **Introduction to Flex** 3
- 1.1 Why are web applications so prolific? 4
 - 1.2 Prolific, but at a price 4
 - 1.3 The RIA solution 5
 - They all want it all* 5 ■ *RIAs to the rescue* 6
 - How RIAs do it* 6
 - 1.4 The RIA contenders 6
 - Flex by Adobe* 7 ■ *Silverlight by Microsoft* 7
 - JavaFX by Sun Microsystems* 8

- 1.5 Flex vs. web applications 8
 - Web applications are based on documents* 9
 - *The role of browsers as transfer stations and document renderers* 9
 - *RWAs—the last stand* 10
 - *Cross-platform issues* 10
 - *Where’s the richness?* 11
 - *AJAX communication limits* 11
 - *Apples and oranges* 12
- 1.6 Becoming acquainted with Flex 13
 - Taking advantage of Adobe Flash* 13
 - *Flex and JavaScript can play together* 14
 - *The Flex ecosystem* 15
 - *How Flex works* 16
 - Events, events, events* 18
- 1.7 What’s new in Flex 3 19
- 1.8 Summary 20

2 Getting started 21

- 2.1 Flex on the cheap 22
 - Setting up the compile environment* 22
 - *Setting up the editing environment* 24
 - *Next steps (if you’re still interested)* 25
- 2.2 Get serious with Flex Builder 26
 - Product and pricing matrix* 26
 - *Getting Flex Builder* 27
- 2.3 Exploring Flex Builder 27
- 2.4 Views and perspectives 29
 - Out-of-the-box perspectives* 29
 - *Switching perspectives* 30
 - Customizing perspectives* 30
- 2.5 Our first project—Hello World! 31
 - Create the project* 31
 - *Entering code* 33
 - *Compile and run* 33
 - *Making it real* 33
- 2.6 Using design mode 34
- 2.7 Built-in help 35
 - Object-oriented languages and their APIs* 36
 - Accessing the API Reference* 36
 - *Perusing API Reference* 37
- 2.8 MXML and ActionScript in a nutshell 38
 - The structure of MXML* 38
 - How MXML and ActionScript relate* 39
 - Events are handled by ActionScript* 40
- 2.9 Summary 42

3 Working with ActionScript 43

3.1 Comments 44

Inline comments 44 ▪ *Block comments* 44

3.2 Variables 44

Variable names 45 ▪ *Strict data typing* 45 ▪ *Static vs. dynamic type checking* 45 ▪ *Primitive and complex data types* 45

3.3 Loops 48

For (starting value; valid condition; increment) 49 ▪ *For (property names in array/object)* 49 ▪ *For each (item in array/object)* 50
While (condition) 50 ▪ *Do while (condition)* 51

3.4 Conditional statements (if statements and switches) 51

If..else 51 ▪ *Switch* 53

3.5 Arrays 54

Indexed arrays 55 ▪ *Associative arrays* 57

3.6 ActionScript tidbits 58

Braces 58 ▪ *Logical operators and shortcuts* 59

3.7 Sneak peek at functions, classes, and packages 61

Your own functions 62 ▪ *Separating ActionScript to individual files* 65

3.8 Simple data binding 66

Once again, it is about events 66 ▪ *Bidirectional binding* 67

3.9 Summary 68

4 Layout and containers 69

4.1 Absolute layout 70

4.2 Constraint-based layout 72

Basic constraints 72 ▪ *Enhanced constraints* 74

4.3 Automatic layout 79

4.4 Variable and fixed sizing 81

Variable sizing 81 ▪ *Fixed Sizing* 81

4.5 Containers 82

Application container 82 ▪ *Canvas container* 83 ▪ *Box, HBox, and VBox containers* 84 ▪ *Panel container* 85
ApplicationControlBar and ControlBar containers 86
DividedBox, HDividedBox, and VDividedBox containers 86
Form container 88 ▪ *Grid container* 88 ▪ *Tile container* 89

- 4.6 Dynamic layout with Repeaters 90
 - dataProvider sneak peek* 90
 - *Properties and events of a Repeater* 91
 - *Creating the Repeater* 92
 - *Working with Repeater events* 93
- 4.7 Summary 95

5 *Displaying forms and capturing user input* 96

- 5.1 The id attribute 97
- 5.2 Text controls 97
- 5.3 Date controls 99
- 5.4 Numeric controls 100
- 5.5 Buttons 102
 - Bars of Buttons* 104
 - *The PopUpButton and PopUpMenuButton* 104
- 5.6 Picklist controls 107
- 5.7 Accessing the control's value 109
 - Passing values to a function* 109
 - *Passing events to a function* 109
 - *Accessing properties directly* 111
 - *Which approach to use?* 112
- 5.8 Summary 112

6 *Validating user input* 113

- 6.1 Overview of validation 114
- 6.2 Built-in validators 115
 - Validator* 115
 - *StringValidator* 117
 - *NumberValidator* 118
 - DateValidator* 120
 - *EmailValidator* 123
 - CreditCardValidator* 125
 - *CurrencyValidator* 127
 - PhoneNumberValidator* 129
 - *RegExpValidator* 130
 - SocialSecurityValidator* 132
 - *ZipCodeValidator* 133
- 6.3 Real-time validation 134
- 6.4 Committed value validation 135
- 6.5 Pass-through validation 135
- 6.6 Scripted validation 136
- 6.7 Validation tidbits 137
 - Does a validator always check all criteria?* 137
 - *Validating what was entered vs. criteria matching* 137
 - *Controlling what triggers validation* 137
- 6.8 Summary 138

7 *Formatting data* 139

7.1 Built-in formatters 140

Formatter 140 ▪ *NumberFormatter* 141 ▪ *CurrencyFormatter* 143 ▪ *DateFormatter* 144 ▪ *PhoneFormatter* 146
ZipCodeFormatter 148 ▪ *SwitchSymbolFormatter* 149

7.2 Real-time formatting 151

7.3 Scripted formatting 151

Using a function with a formatter component 151
Using a function with a formatter class 152

7.4 Working with formatting errors 153

7.5 Summary 154

8 *DataGrids, lists, and trees* 155

8.1 List genealogy 155

ListBase and *AdvancedListBase*'s properties 156
ListBase events 157

8.2 The dataProvider and collections 158

Feeding the dataProvider 158 ▪ *Types of collections* 159
Users of collections 159

8.3 Initializing collections 159

8.4 Populating collections 160

List 161 ▪ *HorizontalList* 163 ▪ *TileList* 164
DataGrid 165 ▪ *Tree* 169 ▪ *AdvancedDataGrid* 172

8.5 Interacting with lists 178

List events 179 ▪ *Passing the event to a function* 179 ▪ *Passing data to a function* 181 ▪ *Accessing the selected row directly* 182
Binding to a selected row 182 ▪ *Using a different event as a trigger* 183

8.6 Summary 183

9 *List customization* 184

9.1 Label functions 184

Types of label functions 185 ▪ *Using a single-column label function* 186 ▪ *Using a multicolumn label function* 187
Ideas for label functions 188

- 9.2 Item renderers 188
 - Types of renderers* 189 ▪ *Creating a (regular) item renderer* 189
 - Adding logic to an item renderer* 190 ▪ *Creating an inline item renderer* 193 ▪ *Using drop-in item renderers* 193
- 9.3 Item editors 195
 - Enabling item editing* 195 ▪ *Creating an item editor* 196 ▪ *Item-editing events* 198
- 9.4 Combining an item editor and item renderer 204
- 9.5 Advanced item renderers 205
 - The AdvancedDataGridRendererProvider* 205 ▪ *Referencing the column* 206 ▪ *Spanning columns* 207 ▪ *Spanning an entire row* 208
- 9.6 Filter functions 209
- 9.7 Summary 211

PART 2 APPLICATION FLOW AND STRUCTURE213

10 Events 215

- 10.1 The event system 216
 - Event system—the Postal Service* 216 ▪ *Event-delivery system* 217 ▪ *Set your phases on stun* 218
- 10.2 Sending and receiving events 219
 - Adding event listeners* 219 ▪ *Keying into binding events* 221
 - Removing event listeners* 223
- 10.3 Custom events 226
 - Using the dispatcher to send an event* 226 ▪ *Creating custom events* 227 ▪ *Stopping event propagation* 230
 - Adding event metadata to components* 231
- 10.4 Summary 232

11 Application navigation 233

- 11.1 Feeding the data provider for menus and menu bars 234
 - Nested arrays* 234 ▪ *Nested array collections* 235
 - Models* 235 ▪ *XML component and class* 236
 - XMLList component* 237 ▪ *XMLListCollection component and class* 238 ▪ *Choosing a data type for the data provider* 240

- 11.2 Working with menus 240
 - Creating a menu* 240
 - *Positioning the menu* 242
 - Customizing menu items* 242
- 11.3 Using a menu bar 246
 - Creating a menu bar* 247
 - *Positioning the menu bar* 248
 - Customizing items in the menu bar* 248
 - *Handling user interactions with menu bars* 249
- 11.4 Using view stacks 251
 - Creating a view stack* 251
 - *Adding navigation to the view stack* 253
 - *Determining which stack is selected* 254
 - Handling user interactions with view stacks* 255
- 11.5 TabNavigator 257
 - Creating a tab navigator* 258
 - *Handling user interactions with a tab navigator* 258
- 11.6 Accordion 260
 - Creating an accordion* 260
 - *Populating an accordion* 261
 - Handling user interactions with an accordion* 262
- 11.7 Summary 262

12 *Introduction to pop-ups* 263

- 12.1 Creating your first pop-up 263
 - First things first: create your title window* 264
 - *Using PopUpManager to open the window* 264
 - *Closing the pop-up* 266
- 12.2 Controlling the window location 266
 - Using the centerPopUp() method* 266
 - Calculating window placement* 267
- 12.3 Setting window transparency 270
- 12.4 Data integration with pop-ups 271
 - Getting data out of your pop-up window* 273
 - *Sending events* 273
 - *Getting data out* 274
 - *Sending data to the window* 275
- 12.5 Using alerts 276
 - Creating a simple alert* 276
 - *Doing more with alerts* 276
 - A more advanced alert* 277
 - *Pimp this alert* 278
- 12.6 Summary 280

13 *View states* 281

- 13.1 Understanding view states 281
- 13.2 View states in Flex 282
 - View states classes* 284
 - *View-state example* 285
 - *Defining a view state (<mx:State />)* 287
 - *View-state transitions* 289
 - States in components* 291
- 13.3 Summary 293

14 *Working with data services* 294

- 14.1 What the heck is a WSDL document? 295
- 14.2 Reading WSDL 295
 - Reading the WSDL document: operations* 295
 - *Reading the WSDL: input types* 296
 - *Reading the WSDL document: output* 297
- 14.3 Using WebService components 298
 - Creating a WebService component with ActionScript* 300
 - Calling the WebService component* 300
 - *Handling the result* 301
 - *The ResultEvent* 301
 - *Working with the result* 302
 - *Fault events* 303
 - *Using asynchronous tokens* 304
- 14.4 Using HTTPService to retrieve results 305
 - Connecting to an HTTP web service* 306
 - Explicit parameter binding* 307
- 14.5 Using the RemoteObject tag 308
- 14.6 Using the URLLoader 310
- 14.7 Using the Flex Import Web Service wizard 311
 - Using the Import Web Service wizard* 312
 - Working with generated web-service code* 313
- 14.8 Summary 315

15 *Working with XML* 316

- 15.1 XML primer 316
 - What is XML?* 317
 - *Benefits of XML* 317
 - *Drawbacks* 318
 - Syntax and rules* 318
 - *DTDs and XML Schema* 321
 - Namespaces* 322

- 15.2 XML components and classes 323
 - The XML component* 323 ▪ *The XML class* 324 ▪ *The XMLList component* 329 ▪ *The XMLList class* 329 ▪ *The XMLListCollection component and class* 330
- 15.3 Loading external XML source data 331
- 15.4 Binding in XML 332
 - Binding from XML* 332 ▪ *Binding to XML* 332
- 15.5 ECMAScript for XML (E4X) 333
 - E4X syntax* 334 ▪ *Working with XML structure* 335
- 15.6 Summary 339

16 *Objects and classes* 340

- 16.1 OO theory in 5 minutes 341
 - The relationship between objects and classes* 341 ▪ *Objects have properties and methods* 341 ▪ *Inheritance* 342 ▪ *Encapsulation and coupling* 343 ▪ *Objects: they're everywhere!* 344
- 16.2 Playing with objects 344
 - A closer look at objects* 344 ▪ *Methods of objects* 345 ▪ *Method parameters* 346 ▪ *Methods return information* 346 ▪ *Objects have properties* 346
- 16.3 Creating a class 346
 - Creating the class file* 347 ▪ *Specifying a package* 347 ▪ *Naming classes* 347 ▪ *Class modifiers* 347 ▪ *Superclasses: extending a class* 348 ▪ *Interfaces* 348 ▪ *Looking at your class* 349
- 16.4 Adding properties 350
- 16.5 Adding getter/setter methods 351
- 16.6 Creating methods for your class 352
- 16.7 Summary 354

17 *Custom components* 355

- 17.1 What are custom components? 356
 - A custom component example* 356
- 17.2 How custom components work 357
 - Simple and composite types* 357 ▪ *Implementation choices* 358
- 17.3 Simple custom components 358

- 17.4 Passing parameters 359
 - Using a function to pass a value* 359
 - *Passing a value as a property* 360
 - *Using a setter function as a property proxy* 363
 - *Using bound parameters* 364
 - 17.5 Retrieving values 365
 - Using a function to retrieve a value* 365
 - *Using a property to retrieve a value* 366
 - *Using a getter function as a property proxy* 369
 - 17.6 Creating composite components 371
 - 17.7 Creating ActionScript components 372
 - 17.8 Custom events 374
 - Passing along the event object* 376
 - 17.9 Namespaces and structure 377
 - The MX namespace* 377
 - *The local namespace* 377
 - Directory structure and namespaces* 377
 - *Namespace in ActionScript* 378
 - 17.10 Summary 378
- 18** ***Advanced reusability in Flex*** 380
- 18.1 SWC files 381
 - 18.2 Runtime shared libraries 381
 - Standard RSLs* 382
 - Making your Flex application use RSL* 383
 - 18.3 Modular Flex application development 385
 - Components vs. modules* 385
 - *Creating a simple module* 385
 - Loading modules the MXML way with the Module API* 386
 - Loading and unloading modules with ActionScript* 387
 - Pros and cons of modules* 388
 - 18.4 Adding patches in your Flex application lifecycle 389
 - Using a SWC to update and replace a class* 389
 - 18.5 Refactoring 389
 - 18.6 Summary 390

PART 3 THE FINISHING TOUCHES391

19 Customizing the experience 393

19.1 Styles 393

Inline styles 394 ▪ *Local style definitions* 396 ▪ *External stylesheets* 400 ▪ *The Style Explorer* 401 ▪ *Working with color* 401 ▪ *Transparency* 403 ▪ *Using gradients* 404
Working with styles programmatically 405

19.2 Embedding fonts 407

Embedding via the font's system name 407 ▪ *Embedding using a font file* 408 ▪ *Leveraging CSS* 408

19.3 Images and icons 409

Image types 409 ▪ *To embed or not to embed* 409 ▪ *Images as variables* 410 ▪ *Icons* 411

19.4 Skins 413

Types of skins 413 ▪ *Graphical skins with images* 413
Graphical skins with SWFs 415 ▪ *Flex skin design extensions (SDE)* 416 ▪ *Image slicing* 417 ▪ *Programmatic skins* 418

19.5 Summary 419

20 Working with effects 420

20.1 What is an effect? 420

Cause and effect 421 ▪ *Out-of-the-box effects* 421
Composite effects 421

20.2 Triggered effects 421

20.3 Programmatically applying an effect 423

Creating effects with ActionScript 423
Using just ActionScript 424

20.4 Creating composite effects 425

Sequential effects 425 ▪ *Parallel effects* 426
Composite composites 426

20.5 Using out-of-the-box effects 427

The AnimateProperty effect 427 ▪ *The Blur effect* 428 ▪ *The Dissolve effect* 428 ▪ *The Fade effect* 429 ▪ *The Glow effect* 430 ▪ *The Iris effect* 431 ▪ *Move effect* 431 ▪ *The Pause effect* 432 ▪ *Resize effect* 433 ▪ *The Rotate effect* 434
The SoundEffect effect 434 ▪ *The Wipe effects* 435 ▪ *The Zoom effect* 436

- 20.6 Easing functions 437
 - Out-of-the-box easing functions* 437
 - Making your own easing functions* 438
- 20.7 Fonts and effects 439
- 20.8 Summary 439

21 *Drag-and-drop* 440

- 21.1 The drag-and-drop process 441
- 21.2 Drag-and-drop events 442
- 21.3 Components that support drag-and-drop 443
- 21.4 Enabling D&D on List-based components 443
- 21.5 Moving versus copying 445
- 21.6 Multi-item drag 445
- 21.7 Two-way drag-and-drop 447
- 21.8 Using D&D for user-controlled sorting 448
- 21.9 Enter the DragManager 449
 - Operational values* 449 ▪ *DragManager functions* 449
- 21.10 Accepting or denying a drop 450
 - Limiting who gets into the party* 450 ▪ *Preventing event propagation* 451 ▪ *Use the DragEvent object to find the drop target* 452
- 21.11 Applying your own drop 453
 - Adding to the component's dataProvider explicitly* 453
 - Adding to the component's dataProvider implicitly* 454
- 21.12 Adding D&D to non-List components 454
 - Setting up the example* 455 ▪ *Initiating the drag* 455 ▪ *Adding visual feedback* 456 ▪ *Handling the drop* 456 ▪ *Handling the exit* 457 ▪ *Putting it all together* 457
- 21.13 Customizing the drag-and-drop experience 459
 - Changing the drag image* 459
 - Changing the drag proxy Icons* 460
- 21.14 Summary 461

22 *Charting* 462

- 22.1 Introduction to charting 462
 - Chart parts* 463 ▪ *types overview* 463
- 22.2 Setting the stage with series and data 464
- 22.3 Creating charts 465
 - Invoking a chart* 465 ▪ *Adding a legend* 466
 - Changing chart types* 467
- 22.4 Stacking charts 469
- 22.5 Exploring chart types 470
 - Area charts* 470 ▪ *Bar charts and column charts* 471 ▪ *Line charts* 472 ▪ *Bubble charts* 473 ▪ *Candlestick and HLOC charts* 475 ▪ *Pie charts* 477 ▪ *Plot charts* 479
- 22.6 Customizing charts 481
 - Series strokes* 481 ▪ *Series fills* 481
- 22.7 Summary 483

23 *Debugging and testing* 484

- 23.1 Debugging 485
 - Using the Flash Debug Player* 485 ▪ *Configuring logging* 485
 - Using the trace() function* 486 ▪ *Trace-log viewers* 487
 - Converting objects to strings* 487 ▪ *FxSpy* 489 ▪ *Monitoring network activity* 489 ▪ *Using the Debugger* 490
- 23.2 Testing 493
 - Types of tests* 493 ▪ *Flex Profiler* 494 ▪ *FlexUnit (unit testing)* 497 ▪ *Fluint (unit testing)* 500 ▪ *FunFX (functional testing)* 503 ▪ *RIATest (functional testing)* 503 ▪ *HP QuickTest Pro (functional testing)* 504 ▪ *IBM Rational Functional Tester (functional testing)* 504
- 23.3 Summary 505

24 *Wrapping up a project* 506

- 24.1 Printing 506
 - Flex's approach to printing* 507 ▪ *Tools of the trade* 507
 - Printing things* 507 ▪ *Scaling things for print* 508
 - The art of adding objects* 509 ▪ *Printing lists* 512 ▪ *Catching when a user cancels* 514 ▪ *FlexReport* 515

- 24.2 Customizing the wrapper 515
 - Wrapper files* 515
 - *Wrapper templates* 516
 - *The bare essentials* 516
 - *Embedding into a web application* 517
 - Passing parameters in a wrapper* 517
- 24.3 Deployment 519
 - Create a production build* 519
 - *Positioning client-side files* 520
 - Positioning server-side files* 521
 - *Testing that it works* 521
- 24.4 Summary 521
 - resources* 523
 - index* 527