

Contents

Introduction	1
What Is in This Book?.....	1
Reducing the Learning Curve.....	1
Excel VBA Power.....	2
Techie Stuff Needed to Produce Applications.....	2
Does This Book Teach Excel?.....	2
The Future of VBA and Windows Versions of Excel.....	4
Versions of Excel.....	4
Differences for Mac Users.....	4
Special Elements and Typographical Conventions.....	5
Code Files.....	5
Next Steps.....	5
1 Unleashing the Power of Excel with VBA	7
The Power of Excel.....	7
Barriers to Entry.....	7
The Macro Recorder Doesn't Work!.....	7
No One on the Excel Team Is Focused on the Macro Recorder.....	8
Visual Basic Is Not Like BASIC.....	8
Good News: Climbing the Learning Curve Is Easy.....	9
Great News: Excel with VBA Is Worth the Effort.....	9
Knowing Your Tools: The Developer Tab.....	9
Understanding Which File Types Allow Macros.....	10
Macro Security.....	12
Adding a Trusted Location.....	12
Using Macro Settings to Enable Macros in Workbooks Outside Trusted Locations.....	13
Using Disable All Macros with Notification.....	14
Overview of Recording, Storing, and Running a Macro.....	14
Filling Out the Record Macro Dialog.....	15
Running a Macro.....	16
Creating a Macro Button on the Ribbon.....	16
Creating a Macro Button on the Quick Access Toolbar.....	17
Assigning a Macro to a Form Control, Text Box, or Shape.....	18
Understanding the VB Editor.....	19
VB Editor Settings.....	20
The Project Explorer.....	20
The Properties Window.....	21
Understanding Shortcomings of the Macro Recorder.....	21
Recording the Macro.....	23
Examining Code in the Programming Window.....	23
Running the Macro on Another Day Produces Undesired Results.....	25
Possible Solution: Use Relative References When Recording.....	26
Never Use AutoSum or Quick Analysis While Recording a Macro.....	30

Four Tips for Using the Macro Recorder.....	31
Next Steps.....	32
2 This Sounds Like BASIC, So Why Doesn't It Look Familiar?.....	33
I Can't Understand This Code.....	33
Understanding the Parts of VBA "Speech".....	34
VBA Is Not Really Hard.....	37
VBA Help Files: Using F1 to Find Anything.....	38
Using Help Topics.....	38
Examining Recorded Macro Code: Using the VB Editor and Help.....	39
Optional Parameters.....	39
Defined Constants.....	40
Properties Can Return Objects.....	43
Using Debugging Tools to Figure Out Recorded Code.....	43
Stepping Through Code.....	43
More Debugging Options: Breakpoints.....	45
Backing Up or Moving Forward in Code.....	45
Not Stepping Through Each Line of Code.....	46
Querying Anything While Stepping Through Code.....	46
Using a Watch to Set a Breakpoint.....	49
Using a Watch on an Object.....	49
Object Browser: The Ultimate Reference.....	50
Seven Tips for Cleaning Up Recorded Code.....	51
Tip 1: Don't Select Anything.....	51
Tip 2: Use <code>Cells(2, 5)</code> Because It's More Convenient Than <code>Range("E2")</code>	52
Tip 3: Use More Reliable Ways to Find the Last Row.....	52
Tip 4: Use Variables to Avoid Hard-Coding Rows and Formulas.....	53
Tip 5: Use R1C1 Formulas That Make Your Life Easier.....	54
Tip 6: Copy and Paste in a Single Statement.....	54
Tip 7: Use <code>With...End With</code> to Perform Multiple Actions.....	54
Next Steps.....	57
3 Referring to Ranges.....	59
The Range Object.....	59
Syntax for Specifying a Range.....	60
Named Ranges.....	60
Shortcut for Referencing Ranges.....	60
Referencing Ranges in Other Sheets.....	61
Referencing a Range Relative to Another Range.....	61
Using the <code>Cells</code> Property to Select a Range.....	62
Using the <code>Offset</code> Property to Refer to a Range.....	63
Using the <code>Resize</code> Property to Change the Size of a Range.....	65
Using the <code>Columns</code> and <code>Rows</code> Properties to Specify a Range.....	66
Using the <code>Union</code> Method to Join Multiple Ranges.....	66

Using the <code>Intersect</code> Method to Create a New Range from Overlapping Ranges.....	67
Using the <code>IsEmpty</code> Function to Check Whether a Cell Is Empty.....	67
Using the <code>CurrentRegion</code> Property to Select a Data Range.....	68
Using the <code>Areas</code> Collection to Return a Noncontiguous Range.....	70
Referencing Tables.....	71
Next Steps.....	72
4 Looping and Flow Control.....	73
For . . . Next Loops.....	73
Using Variables in the For Statement.....	75
Variations on the For . . . Next Loop.....	76
Exiting a Loop Early After a Condition Is Met.....	77
Nesting One Loop Inside Another Loop.....	78
Do Loops.....	78
Using the while or Until Clause in Do Loops.....	81
The VBA Loop: For Each.....	82
Object Variables.....	83
Flow Control: Using If . . . Then . . . Else and Select Case.....	86
Basic Flow Control: If . . . Then . . . Else.....	86
Using Select Case . . . End Select for Multiple Conditions.....	88
Next Steps.....	91
5 R1C1-Style Formulas.....	93
Referring to Cells: A1 Versus R1C1 References.....	93
Toggling to R1C1-Style References.....	94
Witnessing the Miracle of Excel Formulas.....	95
Entering a Formula Once and Copying 1,000 Times.....	95
The Secret: It's Not That Amazing.....	96
Understanding the R1C1 Reference Style.....	97
Using R1C1 with Relative References.....	97
Using R1C1 with Absolute References.....	98
Using R1C1 with Mixed References.....	98
Referring to Entire Columns or Rows with R1C1 Style.....	99
Replacing Many A1 Formulas with a Single R1C1 Formula.....	99
Remembering Column Numbers Associated with Column Letters.....	101
Using R1C1 Formulas with Array Formulas.....	101
Next Steps.....	102
6 Creating and Manipulating Names in VBA.....	103
Global Versus Local Names.....	103
Adding Names.....	104
Deleting Names.....	105
Adding Comments.....	106
Types of Names.....	106
Formulas.....	106

Strings.....	107
Numbers.....	108
Tables.....	109
Using Arrays in Names.....	109
Reserved Names.....	110
Hiding Names.....	111
Checking for the Existence of a Name.....	111
Next Steps.....	114
7 Event Programming.....	115
Levels of Events.....	115
Using Events.....	116
Event Parameters.....	116
Enabling Events.....	117
Workbook Events.....	117
Workbook-Level Sheet and Chart Events.....	119
Worksheet Events.....	120
Chart Events.....	123
Embedded Charts.....	123
Embedded Chart and Chart Sheet Events.....	124
Application-Level Events.....	125
Next Steps.....	130
8 Arrays.....	131
Declaring an Array.....	131
Declaring a Multidimensional Array.....	132
Filling an Array.....	133
Retrieving Data from an Array.....	134
Using Arrays to Speed Up Code.....	135
Using Dynamic Arrays.....	136
Passing an Array.....	137
Next Steps.....	138
9 Creating Classes and Collections.....	139
Inserting a Class Module.....	139
Trapping Application and Embedded Chart Events.....	140
Application Events.....	140
Embedded Chart Events.....	141
Creating a Custom Object.....	143
Using a Custom Object.....	145
Using Collections.....	145
Creating a Collection.....	146
Creating a Collection in a Standard Module.....	146
Creating a Collection in a Class Module.....	148

Using Dictionaries	150
Using User-Defined Types to Create Custom Properties	153
Next Steps	156
10 Userforms: An Introduction	157
Input Boxes	157
Message Boxes	158
Creating a Userform	158
Calling and Hiding a Userform	159
Programming Userforms	160
Userform Events	160
Programming Controls	162
Using Basic Form Controls	163
Using Labels, Text Boxes, and Command Buttons	163
Deciding Whether to Use List Boxes or Combo Boxes in Forms	165
Adding Option Buttons to a Userform	167
Adding Graphics to a Userform	169
Using a Spin Button on a Userform	170
Using the <code>MultiPage</code> Control to Combine Forms	171
Verifying Field Entry	174
Illegal Window Closing	174
Getting a Filename	175
Next Steps	176
11 Data Mining with Advanced Filter	177
Replacing a Loop with <code>AutoFilter</code>	177
Using <code>AutoFilter</code> Techniques	180
Selecting Visible Cells Only	183
Advanced Filter—Easier in VBA Than in Excel	184
Using the Excel Interface to Build an Advanced Filter	185
Using Advanced Filter to Extract a Unique List of Values	186
Extracting a Unique List of Values with the User Interface	186
Extracting a Unique List of Values with VBA Code	187
Getting Unique Combinations of Two or More Fields	191
Using Advanced Filter with Criteria Ranges	192
Joining Multiple Criteria with a Logical OR	193
Joining Two Criteria with a Logical AND	194
Other Slightly Complex Criteria Ranges	194
The Most Complex Criteria: Replacing the List of Values with a Condition Created as the Result of a Formula	194
Using Filter in Place in Advanced Filter	201
Catching No Records When Using a Filter in Place	202
Showing All Records After Running a Filter in Place	202
The Real Workhorse: <code>xlFilterCopy</code> with All Records Rather Than Unique Records Only	203
Copying All Columns	203

Copying a Subset of Columns and Reordering	204
Excel in Practice: Turning Off a Few Drop-downs in the AutoFilter	209
Next Steps.....	210
12 Using VBA to Create Pivot Tables	211
Understanding How Pivot Tables Evolved Over Various Excel Versions.....	211
While Building a Pivot Table in Excel VBA	212
Defining the Pivot Cache	212
Creating and Configuring the Pivot Table.....	213
Adding Fields to the Data Area.....	214
Learning Why You Cannot Move or Change Part of a Pivot Report.....	216
Determining the Size of a Finished Pivot Table to Convert the Pivot Table to Values	217
Using Advanced Pivot Table Features.....	219
Using Multiple Value Fields.....	220
Grouping Daily Dates to Months, Quarters, or Years	221
Changing the Calculation to Show Percentages	222
Eliminating Blank Cells in the Values Area	225
Controlling the Sort Order with AutoSort.....	225
Replicating the Report for Every Product	225
Filtering a Data Set.....	228
Manually Filtering Two or More Items in a Pivot Field	228
Using the Conceptual Filters.....	229
Using the Search Filter.....	233
Setting Up Slicers to Filter a Pivot Table	235
Setting Up a Timeline to Filter an Excel 2016 Pivot Table.....	239
Using the Data Model in Excel 2016	242
Adding Both Tables to the Data Model.....	242
Creating a Relationship Between the Two Tables.....	243
Defining the PivotCache and Building the Pivot Table.....	243
Adding Model Fields to the Pivot Table.....	244
Adding Numeric Fields to the Values Area	244
Putting It All Together.....	245
Using Other Pivot Table Features	247
Calculated Data Fields	247
Calculated Items	247
Using ShowDetail to Filter a Record Set	248
Changing the Layout from the Design Tab	248
Settings for the Report Layout.....	248
Suppressing Subtotals for Multiple Row Fields.....	249
Next Steps.....	250
13 Excel Power	251
File Operations.....	251
Listing Files in a Directory	251
Importing and Deleting a CSV File.....	254
Reading a Text File into Memory and Parsing.....	254

Combining and Separating Workbooks	255
Separating Worksheets into Workbooks.....	255
Combining Workbooks.....	256
Filtering and Copying Data to Separate Worksheets.....	257
Copying Data to Separate Worksheets Without Using Filter.....	258
Exporting Data to an XML File.....	259
Working with Cell Comments	260
Resizing Comments.....	260
Placing a Chart in a Comment.....	261
Selecting Cells	263
Using Conditional Formatting to Highlight the Selected Cell.....	263
Highlighting the Selected Cell Without Using Conditional Formatting.....	264
Selecting/Deselecting Noncontiguous Cells.....	265
Creating a Hidden Log File.....	267
Techniques for VBA Pros.....	268
Creating an Excel State Class Module.....	268
Drilling-Down a Pivot Table.....	270
Filtering an OLAP Pivot Table by a List of Items.....	271
Creating a Custom Sort Order.....	273
Creating a Cell Progress Indicator.....	274
Using a Protected Password Box.....	275
Changing Case.....	277
Selecting with SpecialCells.....	279
Resetting a Table's Format.....	279
Cool Applications.....	280
Getting Historical Stock/Fund Quotes.....	280
Using VBA Extensibility to Add Code to New Workbooks.....	281
Next Steps.....	282
14 Sample User-Defined Functions.....	283
Creating User-Defined Functions.....	283
Sharing UDFs.....	286
Useful Custom Excel Functions.....	286
Setting the Current Workbook's Name in a Cell.....	286
Setting the Current Workbook's Name and File Path in a Cell.....	287
Checking Whether a Workbook Is Open.....	287
Checking Whether a Sheet in an Open Workbook Exists.....	287
Counting the Number of Workbooks in a Directory.....	288
Retrieving the User ID.....	289
Retrieving Date and Time of Last Save.....	291
Retrieving Permanent Date and Time.....	291
Validating an Email Address.....	292
Summing Cells Based on Interior Color.....	293
Counting Unique Values.....	294
Removing Duplicates from a Range.....	295

Finding the First Nonzero-Length Cell in a Range	296
Substituting Multiple Characters	297
Retrieving Numbers from Mixed Text	298
Converting Week Number into Date	299
Extracting a Single Element from a Delimited String	300
Sorting and Concatenating	300
Sorting Numeric and Alpha Characters	302
Searching for a String Within Text	303
Reversing the Contents of a Cell	304
Returning the Addresses of Duplicate Max Values	304
Returning a Hyperlink Address	305
Returning the Column Letter of a Cell Address	306
Using Static Random	306
Using <code>Select Case</code> on a Worksheet	307
Next Steps	308
15 Creating Charts	309
Contrasting the Good and Bad VBA to Create Charts	309
Planning for More Charts to Break	310
Using <code>.AddChart2</code> to Create a Chart	311
Understanding Chart Styles	312
Formatting a Chart	315
Referring to a Specific Chart	315
Specifying a Chart Title	316
Applying a Chart Color	317
Filtering a Chart	318
Using <code>SetElement</code> to Emulate Changes from the Plus Icon	319
Using the <code>Format</code> Method to Micromanage Formatting Options	324
Changing an Object's Fill	325
Formatting Line Settings	327
Creating a Combo Chart	327
Exporting a Chart as a Graphic	330
Considering Backward Compatibility	331
Next Steps	331
16 Data Visualizations and Conditional Formatting	333
VBA Methods and Properties for Data Visualizations	334
Adding Data Bars to a Range	335
Adding Color Scales to a Range	339
Adding Icon Sets to a Range	341
Specifying an Icon Set	341
Specifying Ranges for Each Icon	343
Using Visualization Tricks	343
Creating an Icon Set for a Subset of a Range	344
Using Two Colors of Data Bars in a Range	345

Using Other Conditional Formatting Methods.....	347
Formatting Cells That Are Above or Below Average.....	348
Formatting Cells in the Top 10 or Bottom 5.....	348
Formatting Unique or Duplicate Cells.....	349
Formatting Cells Based on Their Value.....	350
Formatting Cells That Contain Text.....	351
Formatting Cells That Contain Dates.....	351
Formatting Cells That Contain Blanks or Errors.....	351
Using a Formula to Determine Which Cells to Format.....	352
Using the New <code>NumberFormat</code> Property.....	353
Next Steps.....	354
17 Dashboarding with Sparklines in Excel 2016.....	355
Creating Sparklines.....	356
Scaling Sparklines.....	357
Formatting Sparklines.....	361
Using Theme Colors.....	361
Using RGB Colors.....	364
Formatting Sparkline Elements.....	365
Formatting Win/Loss Charts.....	368
Creating a Dashboard.....	369
Observations About Sparklines.....	369
Creating Hundreds of Individual Sparklines in a Dashboard.....	370
Next Steps.....	374
18 Reading from and Writing to the Web.....	375
Getting Data from the Web.....	375
Building Multiple Queries with VBA.....	377
Finding Results from Retrieved Data.....	378
Putting It All Together.....	379
Examples of Scraping Websites Using Web Queries.....	380
Using <code>Application.OnTime</code> to Periodically Analyze Data.....	381
Using Ready Mode for Scheduled Procedures.....	381
Specifying a Window of Time for an Update.....	382
Canceling a Previously Scheduled Macro.....	382
Closing Excel Cancels All Pending Scheduled Macros.....	383
Scheduling a Macro to Run <i>x</i> Minutes in the Future.....	383
Scheduling a Verbal Reminder.....	383
Scheduling a Macro to Run Every Two Minutes.....	384
Publishing Data to a Web Page.....	385
Using VBA to Create Custom Web Pages.....	386
Using Excel as a Content Management System.....	387
Bonus: FTP from Excel.....	389
Next Steps.....	390

19 Text File Processing	391
Importing from Text Files.....	391
Importing Text Files with Fewer Than 1,048,576 Rows.....	391
Dealing with Text Files with More Than 1,048,576 Rows.....	398
Writing Text Files.....	402
Next Steps.....	403
20 Automating Word	405
Using Early Binding to Reference a Word Object.....	406
Using Late Binding to Reference a Word Object.....	408
Using the New Keyword to Reference a Word Application.....	409
Using the CreateObject Function to Create a New Instance of an Object.....	409
Using the GetObject Function to Reference an Existing Instance of Word.....	410
Using Constant Values.....	411
Using the Watches Window to Retrieve the Real Value of a Constant.....	411
Using the Object Browser to Retrieve the Real Value of a Constant.....	412
Understanding Word's Objects.....	413
The Document Object.....	413
The Selection Object.....	415
The Range Object.....	416
Bookmarks.....	419
Controlling Form Fields in Word.....	420
Next Steps.....	422
21 Using Access as a Back End to Enhance Multiuser Access to Data	423
ADO Versus DAOs.....	424
The Tools of ADO.....	426
Adding a Record to a Database.....	427
Retrieving Records from a Database.....	429
Updating an Existing Record.....	431
Deleting Records via ADO.....	433
Summarizing Records via ADO.....	433
Other Utilities via ADO.....	434
Checking for the Existence of Tables.....	434
Checking for the Existence of a Field.....	435
Adding a Table On the Fly.....	436
Adding a Field On the Fly.....	436
SQL Server Examples.....	437
Next Steps.....	438
22 Advanced Userform Techniques	439
Using the UserForm Toolbar in the Design of Controls on Userforms.....	439
More Userform Controls.....	440
Checkbox Controls.....	440

Controls and Collections	447
Modeless Userforms	449
Using Hyperlinks in Userforms	449
Adding Controls at Runtime	450
Resizing the Userform On the Fly	452
Adding a Control On the Fly	452
Sizing On the Fly	452
Adding Other Controls	453
Adding an Image On the Fly	453
Putting It All Together	454
Adding Help to a Userform	456
Showing Accelerator Keys	456
Adding Control Tip Text	457
Creating the Tab Order	457
Coloring the Active Control	457
Creating Transparent Forms	460
Next Steps	461
23 The Windows Application Programming Interface (API)	463
Understanding an API Declaration	464
Using an API Declaration	465
Making 32-Bit- and 64-Bit-Compatible API Declarations	465
API Function Examples	467
Retrieving the Computer Name	467
Checking Whether an Excel File Is Open on a Network	467
Retrieving Display-Resolution Information	468
Customizing the About Dialog	469
Disabling the X for Closing a Userform	470
Creating a Running Timer	471
Playing Sounds	472
Next Steps	472
24 Handling Errors	473
What Happens When an Error Occurs?	473
A Misleading Debug Error in Userform Code	475
Basic Error Handling with the <code>On Error GoTo</code> Syntax	477
Generic Error Handlers	478
Handling Errors by Choosing to Ignore Them	479
Suppressing Excel Warnings	481
Encountering Errors on Purpose	481
Training Your Clients	481
Errors While Developing Versus Errors Months Later	482
Runtime Error 9: Subscript Out of Range	482
Runtime Error 1004: Method Range of Object Global Failed	483
The Ills of Protecting Code	484

More Problems with Passwords.....	485
Errors Caused by Different Versions.....	486
Next Steps.....	486
25 Customizing the Ribbon to Run Macros.....	487
Where to Add Code:The customui Folder and File.....	488
Creating a Tab and a Group.....	489
Adding a Control to a Ribbon.....	490
Accessing the File Structure.....	496
Understanding the RELS File.....	496
Renaming an Excel File and Opening a Workbook.....	497
Using Images on Buttons.....	497
Using Microsoft Office Icons on a Ribbon.....	498
Adding Custom Icon Images to a Ribbon.....	499
Troubleshooting Error Messages.....	500
The Attribute " Attribute Name " on the Element " customui Ribbon " Is Not Defined in the DTD/Schema.....	500
Illegal Qualified Name Character.....	501
Element " customui Tag Name " Is Unexpected According to Content Model of Parent Element " customui Tag Name ".....	501
Found a Problem with Some Content.....	502
Wrong Number of Arguments or Invalid Property Assignment.....	503
Invalid File Format or File Extension.....	503
Nothing Happens.....	503
Other Ways to Run a Macro.....	504
Using a Keyboard Shortcut to Run a Macro.....	504
Attaching a Macro to a Command Button.....	504
Attaching a Macro to a Shape.....	505
Attaching a Macro to an ActiveX Control.....	506
Running a Macro from a Hyperlink.....	507
Next Steps.....	508
26 Creating Add-ins.....	509
Characteristics of Standard Add-ins.....	509
Converting an Excel Workbook to an Add-in.....	510
Using Save As to Convert a File to an Add-in.....	511
Using the VB Editor to Convert a File to an Add-in.....	512
Having a Client Install an Add-in.....	512
Closing Add-ins.....	514
Removing Add-ins.....	514
Using a Hidden Workbook as an Alternative to an Add-in.....	515
Next Steps.....	516
27 An Introduction to Creating Office Add-ins.....	517
Creating Your First Office Add-in—Hello World.....	517
Adding Interactivity to an Office Add-in.....	521

A Basic Introduction to HTML.....	524
Using Tags.....	524
Adding Buttons.....	524
Using CSS Files.....	525
Using XML to Define an Office Add-in.....	525
Using JavaScript to Add Interactivity to an Office Add-in.....	526
The Structure of a Function.....	526
Variables.....	527
Strings.....	528
Arrays.....	528
JavaScript for Loops.....	529
How to Do an if Statement in JavaScript.....	530
How to Do a Select . . Case Statement in JavaScript.....	530
How to Do a For each . . next Statement in JavaScript.....	532
Mathematical, Logical, and Assignment Operators.....	532
Math Functions in JavaScript.....	534
Writing to the Content Pane or Task Pane.....	535
JavaScript Changes for Working in an Office Add-in.....	535
Napa Office 365 Development Tools.....	536
Next Steps.....	537
28 What's New in Excel 2016 and What's Changed.....	539
If It Has Changed in the Front End, It Has Changed in VBA.....	539
The Ribbon.....	539
Single Document Interface (SDI).....	540
Quick Analysis Tool.....	541
Charts.....	541
Pivot Tables.....	541
Slicers.....	541
SmartArt.....	542
Learning the New Objects and Methods.....	542
Compatibility Mode.....	542
Using the Version Property.....	543
Using the Excel8CompatibilityMode Property.....	543
Next Steps.....	544
Index.....	545

About the Authors

Bill Jelen, Excel MVP and the host of MrExcel.com, has been using spreadsheets since 1985, and he launched the MrExcel.com website in 1998. Bill was a regular guest on *Call for Help* with Leo Laporte and has produced more than 1,900 episodes of his daily video podcast, *Learn Excel from MrExcel*. He is the author of 44 books about Microsoft Excel and writes the monthly Excel column for *Strategic Finance* magazine. Before founding MrExcel.com, Bill Jelen spent 12 years in the trenches—working as a financial analyst for finance, marketing, accounting, and operations departments of a \$500 million public company. He lives in Merritt Island, Florida, with his wife, Mary Ellen.

Tracy Syrstad is a Microsoft Excel developer and author of eight Excel books. She has been helping people with Microsoft Office issues since 1997, when she discovered free online forums where anyone could ask and answer questions. Tracy found out she enjoyed teaching others new skills, and when she began working as a developer, she was able to integrate the fun of teaching with one-on-one online desktop sharing sessions. Tracy lives on acreage in eastern South Dakota with her husband, one dog, two cats, one horse (two, hopefully soon), and a variety of wild foxes, squirrels, and rabbits.

Dedications

For Robert K. Jelen
—Bill Jelen

For Marlee Jo Jacobson
—Tracy Syrstad

Acknowledgments

Thanks to Tracy Syrstad for being a great coauthor.

Bob Umlas is the smartest Excel guy I know and is an awesome technical editor. At Pearson, Joan Murray is an excellent acquisitions editor.

Along the way, I've learned a lot about VBA programming from the awesome community at the MrExcel.com message board. VoG, Richard Schollar, and Jon von der Heyden all stand out as having contributed posts that led to ideas in this book. Thanks to Pam Gensel for Excel macro lesson #1. Mala Singh taught me about creating charts in VBA, and Oliver Holloway brought me up to speed with accessing SQL Server. Scott Ruble and Robin Wakefield at Microsoft helped with the charting chapter.

My family was incredibly supportive during this time. Thanks to Mary Ellen Jelen, Robert F. Jelen, and Robert K. Jelen.

—Bill

Juan Pablo Gonzalez Ruiz and Zack Barresse are great programmers, and I really appreciate their time and patience showing me new ways to write better programs. Chris “Smitty” Smith has really helped me sharpen my business acumen.

Thank you to all the moderators at the MrExcel forum who keep the board organized, despite the best efforts of the spammers.

Programming is a constant learning experience, and I really appreciate the clients who have encouraged me to program outside my comfort zone so that my skills and knowledge have expanded.

And last, but not least, thanks to Bill Jelen. His site, MrExcel.com, is a place where thousands come for help. It's also a place where I, and others like me, have an opportunity to learn from and assist others.

—Tracy

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@quepublishing.com

Mail: Que Publishing
 ATTN: Reader Feedback
 800 East 96th Street
 Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at quepublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

As corporate IT departments have found themselves with long backlogs of requests, Excel users have discovered that they can produce the reports needed to run their businesses themselves using the macro language *Visual Basic for Applications* (VBA). VBA enables you to achieve tremendous efficiencies in your day-to-day use of Excel. VBA helps you figure out how to import data and produce reports in Excel so that you don't have to wait for the IT department to help you.

What Is in This Book?

You have taken the right step by purchasing this book. We can help you reduce the learning curve so that you can write your own VBA macros and put an end to the burden of generating reports manually.

Reducing the Learning Curve

This Introduction provides a case study about the power of macros. Chapter 1, "Unleashing the Power of Excel with VBA," introduces the tools and confirms what you probably already know: The macro recorder does not work reliably. Chapter 2, "This Sounds Like BASIC, So Why Doesn't It Look Familiar?" helps you understand the crazy syntax of VBA. Chapter 3, "Referring to Ranges," cracks the code on how to work efficiently with ranges and cells.

Chapter 4, "Looping and Flow Control," covers the power of looping using VBA. The case study in this chapter demonstrates creating a program to produce a department report and then wrapping that report routine in a loop to produce 46 reports.

Chapter 5, "R1C1-Style Formulas," covers, obviously, R1C1-style formulas. Chapter 6, "Creating and Manipulate Names in VBA," covers names. Chapter 7, "Event Programming," includes some great tricks that use event programming. Chapters 8, "Arrays,"

INTRODUCTION

IN THIS INTRODUCTION

What Is in This Book?.....	1
The Future of VBA and Windows Versions of Excel	4
Special Elements and Typographical Conventions	5
Code Files	5
Next Steps	5



and 9, “Creating Classes and Collections,” cover arrays, classes, and collections. Chapter 10, “Userforms: An Introduction,” introduces custom dialog boxes that you can use to collect information from a human using Excel.

Excel VBA Power

Chapters 11, “Data Mining with Advanced Filter,” and 12, “Using VBA to Create Pivot Tables,” provide an in-depth look at Filter, Advanced Filter, and pivot tables. Report automation tools rely heavily on these concepts. Chapters 13, “Excel Power,” and 14, “Sample User-Defined Functions,” include dozens of code samples designed to exhibit the power of Excel VBA and custom functions.

Chapters 15, “Creating Charts,” through 20, “Automating Word,” handle charting, data visualizations, web queries, sparklines, and automating Word.

Techie Stuff Needed to Produce Applications

Chapter 21, “Using Access as a Back End to Enhance Multiuser Access to Data,” handles reading and writing to Access databases and SQL Server. The techniques for using Access databases enable you to build an application with the multiuser features of Access while keeping the friendly front end of Excel.

Chapter 22, “Advanced Userform Techniques,” shows you how to go further with userforms. Chapter 23, “The Windows Application Programming Interface (API),” teaches some tricky ways to achieve tasks using the Windows API. Chapters 24, “Handling Errors,” through 26, “Creating Add-ins,” deal with error handling, custom menus, and add-ins. Chapter 27, “An Introduction to Creating Office Add-Ins,” provides a brief introduction to building your own JavaScript application within Excel. Chapter 28, “What’s New in Excel 2016 and What’s Changed,” summarizes the changes in Excel 2016.

Does This Book Teach Excel?

Microsoft believes that the ordinary Office user touches only 10% of the features in Office. We realize that everyone reading this book is above average, and MrExcel.com has a pretty smart audience. Even so, a poll of 8,000 MrExcel.com readers showed that only 42% of smarter-than-average users are using any 1 of the top 10 power features in Excel.

I regularly present a Power Excel seminar for accountants. These are hard-core Excelers who use Excel 30 to 40 hours every week. Even so, two things come out in every seminar. First, half of the audience gasps when they see how quickly you can do tasks with a particular feature, such as automatic subtotals or pivot tables. Second, someone in the audience routinely trumps me. For example, someone asks a question, I answer, and someone in the second row raises a hand to give a better answer.

The point? You and I both know a lot about Excel. However, I assume that in any given chapter, maybe 58% of the people have not used pivot tables before and maybe even fewer have used the Top 10 Filter feature of pivot tables. With this in mind, before I show how to

automate something in VBA, I briefly cover how to do the same task in the Excel interface. This book does not teach you how to make pivot tables, but it does alert you when you might need to explore a topic and learn more about it elsewhere.

CASE STUDY: MONTHLY ACCOUNTING REPORTS

This is a true story. Valerie is a business analyst in the accounting department of a medium-size corporation. Her company recently installed an overbudget \$16 million enterprise resource planning (ERP) system. As the project ground to a close, there were no resources left in the IT budget to produce the monthly report that this corporation used to summarize each department.

However, Valerie had been close enough to the implementation to think of a way to produce the report herself. She understood that she could export general ledger data from the ERP system to a text file with comma-separated values. Using Excel, Valerie was able to import the general ledger data from the ERP system into Excel.

Creating the report was not easy. As in many other companies, there were exceptions in the data. Valerie knew that certain accounts in one particular cost center needed to be reclassified as expenses. She knew that other accounts needed to be excluded from the report entirely. Working carefully in Excel, Valerie made these adjustments. She created one pivot table to produce the first summary section of the report. She cut the pivot table results and pasted them into a blank worksheet. Then she created a new pivot table report for the second section of the summary. After about three hours, she had imported the data, produced five pivot tables, arranged them in a summary, and neatly formatted the report in color.

Becoming the Hero

Valerie handed the report to her manager. The manager had just heard from the IT department that it would be months before they could get around to producing “that convoluted report.” When Valerie created the Excel report, she became the instant hero of the day. In three hours, Valerie had managed to do the impossible. Valerie was on cloud nine after a well-deserved “atta-girl.”

More Cheers

The next day, Valerie’s manager attended the monthly department meeting. When the department managers started complaining that they could not get the report from the ERP system, this manager pulled out his department’s report and placed it on the table. The other managers were amazed. How was he able to produce this report? Everyone was relieved to hear that someone had cracked the code. The company president asked Valerie’s manager if he could have the report produced for each department.

Cheers Turn to Dread

You can probably see what’s coming. This particular company had 46 departments. That means 46 one-page summaries had to be produced once a month. Each report required importing data from the ERP system, backing out certain accounts, producing five pivot tables, and then formatting the reports in color. It had taken Valerie three hours to produce the first report, but after she got into the swing of things, she could produce the 46 reports in 40 hours. Even after she reduced her time per report, though, this is horrible. Valerie had a job to do before she became responsible for spending 40 hours a month producing these reports in Excel.

VBA to the Rescue

Valerie found my company, MrExcel Consulting, and explained her situation. In the course of about a week, I was able to produce a series of macros in Visual Basic that did all the mundane tasks. For example, the macros imported the data, backed out certain accounts, made five pivot tables, and applied the color formatting. From start to finish, the entire 40-hour manual process was reduced to two button clicks and about 4 minutes.

Right now, either you or someone in your company is probably stuck doing manual tasks in Excel that can be automated with VBA. I am confident that I can walk into any company that has 20 or more Excel users and find a case just as amazing as Valerie's.

The Future of VBA and Windows Versions of Excel

Several years ago, there were many rumblings that Microsoft might stop supporting VBA. There is now plenty of evidence to indicate that VBA will be around in Windows versions of Excel through 2036. When VBA was removed from the Mac version of Excel 2008, a huge outcry from customers led to its being included in the next Mac version of Excel.

XLM macros were replaced by VBA in 1993, and 23 years later, they are still supported. Microsoft is making strides toward providing a JavaScript alternative to VBA, but it appears that Excel will support VBA for about another 23 years.

Versions of Excel

This fifth edition of *VBA and Macros* is designed to work with Excel 2016. The previous editions of this book covered code for Excel 97 through Excel 2013. In 80% of the chapters, the code for Excel 2016 is identical to the code in previous versions. However, there are exceptions. For example, the new AutoGroup functionality in pivot tables adds new options that were not available in Excel 2013.

Differences for Mac Users

Although Excel for Windows and Excel for the Mac are similar in terms of user interface, there are a number of differences when you compare the VBA environment. Certainly, nothing in Chapter 23 that uses the Windows API will work on the Mac. That said, the overall concepts discussed in this book apply to the Mac. You can find a general list of differences as they apply to the Mac at <http://www.mrexcel.com/macvba.html>. Development in VBA for Mac Excel 2016 is far more difficult than in Windows, with only rudimentary VBA editing tools. Microsoft actually recommends that you write all of your VBA in Excel 2016 for Windows and then use that VBA on the Mac.

Special Elements and Typographical Conventions

The following typographical conventions are used in this book:

- *Italic*—Indicates new terms when they are defined, special emphasis, non-English words or phrases, and letters or words used as words.
- `Monospace`—Indicates parts of VBA code, such as object or method names.
- **Bold monospace**—Indicates user input.

In addition to these typographical conventions, there are several special elements. Each chapter has at least one case study that presents a real-world solution to common problems. The case study also demonstrates practical applications of topics discussed in the chapter.

In addition to the case studies, you will see Notes, Tips, and Cautions.

NOTE

Notes provide additional information outside the main thread of the chapter discussion that might be useful for you to know.

TIP

Tips provide quick workarounds and time-saving techniques to help you work more efficiently.

CAUTION

Cautions warn about potential pitfalls you might encounter. Pay attention to the Cautions; they alert you to problems that might otherwise cause you hours of frustration.

Code Files

As a thank-you for buying this book, we have put together a set of 50 Excel workbooks that demonstrate the concepts included in this book. This set of files includes all the code from the book, sample data, additional notes from the authors, and 25 bonus macros. To download the code files, visit this book's web page at <http://www.quepublishing.com> or <http://www.mrexcel.com/getcode2016.html>.

Next Steps

Chapter 1 introduces the editing tools of the Visual Basic environment and shows why using the macro recorder is not an effective way to write VBA macro code.

This page intentionally left blank

This page intentionally left blank

Referring to Ranges

3

A *range* can be a cell, a row, a column, or a grouping of any of these. The `RANGE` object is probably the most frequently used object in Excel VBA; after all, you are manipulating data on a sheet. Although a range can refer to any grouping of cells on a sheet, it can refer to only one sheet at a time. If you want to refer to ranges on multiple sheets, you must refer to each sheet separately.

This chapter shows you different ways of referring to ranges, such as specifying a row or column. You'll also find out how to manipulate cells based on the active cell and how to create a new range from overlapping ranges.

The Range Object

The following is the Excel object hierarchy:

```
Application > Workbook > Worksheet >
Range
```

The `Range` object is a property of the `Worksheet` object. This means it requires that a sheet be active or else it must reference a worksheet. Both of the following lines mean the same thing if

`Worksheets(1)` is the active sheet:

```
Range("A1")
Worksheets(1).Range("A1")
```

There are several ways to refer to a `Range` object. `Range("A1")` is the most identifiable because that is how the macro recorder refers to it. However, all the following are equivalent when referring to a range:

```
Range("D5")
[D5]
Range("B3").Range("C3")
Cells(5,4)
Range("A1").Offset(4,3)
Range("MyRange") 'assuming that D5 has a
'Name of MyRange
```

IN THIS CHAPTER

The <code>Range</code> Object.....	59
Syntax for Specifying a Range	60
Named Ranges.....	60
Shortcut for Referencing Ranges	60
Referencing Ranges in Other Sheets	61
Referencing a Range Relative to Another Range.....	61
Using the <code>Cells</code> Property to Select a Range	62
Using the <code>Offset</code> Property to Refer to a Range.....	63
Using the <code>Resize</code> Property to Change the Size of a Range	65
Using the <code>Columns</code> and <code>Rows</code> Properties to Specify a Range.....	66
Using the <code>Union</code> Method to Join Multiple Ranges	66
Using the <code>Intersect</code> Method to Create a New Range from Overlapping Ranges	67
Using the <code>IsEmpty</code> Function to Check Whether a Cell Is Empty	67
Using the <code>CurrentRegion</code> Property to Select a Data Range	68
Using the <code>Areas</code> Collection to Return a Noncontiguous Range	70
Referencing Tables	71
Next Steps	72

Which format you use depends on your needs. Keep reading....It will all make sense soon!

Syntax for Specifying a Range

The `Range` property has two acceptable syntaxes. To specify a rectangular range in the first syntax, specify the complete range reference just as you would in a formula in Excel:

```
Range("A1:B5")
```

In the alternative syntax, specify the upper-left corner and lower-right corner of the desired rectangular range. In this syntax, the equivalent statement might be this:

```
Range("A1", "B5")
```

For either corner, you can substitute a named range, the `Cells` property, or the `ActiveCell` property. The following line of code selects the rectangular range from A1 to the active cell:

```
Range("A1", ActiveCell).Select
```

The following statement selects from the active cell to five rows below the active cell and two columns to the right:

```
Range(ActiveCell, ActiveCell.Offset(5, 2)).Select
```

Named Ranges

You probably have already used named ranges on your worksheets and in formulas. You can also use them in VBA.

Use the following code to refer to the range "MyRange" in Sheet1:

```
Worksheets("Sheet1").Range("MyRange")
```

Notice that the name of the range is in quotes—unlike the use of named ranges in formulas on the sheet itself. If you forget to put the name in quotes, Excel thinks you are referring to a variable in the program. One exception is if you use the shortcut syntax discussed in the next section. In that case, quotes are not used.

Shortcut for Referencing Ranges

A shortcut is available when referencing ranges. The shortcut involves using square brackets, as shown in Table 3.1.

Table 3.1 *Shortcuts for Referencing Ranges*

Standard Method	Shortcut
<code>Range("D5")</code>	<code>[D5]</code>
<code>Range("A1:D5")</code>	<code>[A1:D5]</code>
<code>Range("A1:D5, G6:I17")</code>	<code>[A1:D5, G6:I17]</code>
<code>Range("MyRange")</code>	<code>[MyRange]</code>

Referencing Ranges in Other Sheets

Switching between sheets by activating the needed sheet slows down your code. To avoid this, refer to a sheet that is not active by first referencing the `Worksheet` object:

```
Worksheets("Sheet1").Range("A1")
```

This line of code references Sheet1 of the active workbook even if Sheet2 is the active sheet.

To reference a range in another workbook, include the `Workbook` object, the `Worksheet` object, and then the `Range` object:

```
Workbooks("InvoiceData.xlsx").Worksheets("Sheet1").Range("A1")
```

To use the `Range` property as an argument within another `Range` property, identify the range fully each time. For example, suppose that Sheet1 is your active sheet and you need to total data from Sheet2:

```
WorksheetFunction.Sum(Worksheets("Sheet2").Range(Range("A1"), _  
Range("A7")))
```

This line does not work. Why not? Although `Range("A1"), Range("A7")` is meant to refer to the sheet at the beginning of the code line (Sheet2), Excel does not assume that you want to carry the `Worksheet` object reference over to these other `Range` objects and assumes that they refer to the active sheet, Sheet1. So what do you do? Well, you could write this:

```
WorksheetFunction.Sum(Worksheets("Sheet2").Range(Worksheets("Sheet2"). _  
Range("A1"), Worksheets("Sheet2").Range("A7")))
```

But this not only is a long line of code but also difficult to read! Thankfully, there is a simpler way, using `With...End With`:

```
With Worksheets("Sheet2")  
WorksheetFunction.Sum(.Range(.Range("A1"), .Range("A7")))  
End With
```

Notice now that there is a `.Range` in your code, but without the preceding object reference. That's because `With Worksheets("Sheet2")` implies that the object of the range is the worksheet. Whenever Excel sees a period without an object reference directly to the left of it, it looks up the code for the closest `With` statement and uses that as the object reference.

Referencing a Range Relative to Another Range

Typically, the `Range` object is a property of a worksheet. It is also possible to have `Range` be the property of another range. In this case, the `Range` property is relative to the original range, which makes for unintuitive code. Consider this example:

```
Range("B5").Range("C3").Select
```

This code actually selects cell D7. Think about cell C3, which is located two rows below and two columns to the right of cell A1. The preceding line of code starts at cell B5. If we assume that B5 is in the A1 position, VBA finds the cell that would be in the C3 position relative to B5. In other words, VBA finds the cell that is two rows below and two columns to the right of B5, which is D7.

Again, I consider this coding style to be very unintuitive. This line of code mentions two addresses, and the actual cell selected is neither of these addresses! It seems misleading when you are trying to read this code.

You might consider using this syntax to refer to a cell relative to the active cell. For example, the following line of code activates the cell three rows down and four columns to the right of the currently active cell:

```
Selection.Range("E4").Select
```

I mention this syntax only because the macro recorder uses it. Recall that when you recorded a macro in Chapter 1, “Unleashing the Power of Excel with VBA,” with relative references on, the following line was recorded:

```
ActiveCell.Offset(0, 4).Range("A2").Select
```

This line found the cell four columns to the right of the active cell, and from there it selected the cell that would correspond to A2. This is not the easiest way to write code, but it is the way the macro recorder does it.

Although a worksheet is usually the object of the `Range` property, occasionally, such as during recording, a range may be the property of a range.

Using the `Cells` Property to Select a Range

The `Cells` property refers to all the cells of the specified `Range` object, which can be a worksheet or a range of cells. For example, this line selects all the cells of the active sheet:

```
Cells.Select
```

Using the `Cells` property with the `Range` object might seem redundant:

```
Range("A1:D5").Cells
```

This line refers to the original `Range` object. However, the `Cells` property has an `Item` property that makes the `Cells` property very useful. The `Item` property enables you to refer to a specific cell relative to the `Range` object.

The syntax for using the `Item` property with the `Cells` property is as follows:

```
Cells.Item(Row, Column)
```

You must use a numeric value for `Row`, but you may use the numeric value or string value for `Column`. Both of the following lines refer to cell C5:

```
Cells.Item(5, "C")
Cells.Item(5, 3)
```

Because the `Item` property is the default property of the `Range` object, you can shorten these lines as follows:

```
Cells(5, "C")
Cells(5, 3)
```

The ability to use numeric values for parameters is particularly useful if you need to loop through rows or columns. The macro recorder usually uses something like `Range("A1").`

Select for a single cell and `Range("A1:C5").Select` for a range of cells. If you are learning to code only from the recorder, you might be tempted to write code like this:

```
FinalRow = Cells(Rows.Count, 1).End(xlUp).Row
For i = 1 to FinalRow
    Range("A" & i & ":E" & i).Font.Bold = True
Next i
```

This little piece of code, which loops through rows and bolds the cells in columns A through E, is awkward to read and write. But how else can you do it? Like this:

```
FinalRow = Cells(Rows.Count, 1).End(xlUp).Row
For i = 1 to FinalRow
    Cells(i, "A").Resize(, 5).Font.Bold = True
Next i
```

Instead of trying to type the range address, the new code uses the `Cells` and `Resize` properties to find the required cell, based on the active cell. See the “Using the `Resize` Property to Change the Size of a Range” section later in this chapter, for more information on the `Resize` property.

You can use the `Cells` properties for parameters in the `Range` property. The following refers to the range A1:E5:

```
Range(Cells(1,1), Cells(5,5))
```

This is particularly useful when you need to specify variables with a parameter, as in the previous looping example.

Using the Offset Property to Refer to a Range

You have already seen a reference to `Offset` when you recorded a relative reference. `Offset` enables you to manipulate a cell based on the location of another cell, such as the active cell. Therefore, you do not need to know the address of the cell you want to manipulate.

The syntax for the `Offset` property is as follows:

```
Range.Offset (RowOffset, ColumnOffset)
```

For example, the following code affects cell F5 from cell A1:

```
Range("A1").Offset (RowOffset:=4, ColumnOffset:=5)
```

Or, shorter yet, you can write this:

```
Range("A1").Offset (4, 5)
```

The count of the rows and columns starts at A1 but does not include A1.

If you need to go over only a row or a column, but not both, you don't have to enter both the row and the column parameters. To refer to a cell one column over, use one of these lines:

```
Range("A1").Offset (ColumnOffset:=1)
Range("A1").Offset (, 1)
```

Both of these lines mean the same, so the choice is yours. If you use the second line, make sure to include the comma so Excel knows that the 1 refers to the `ColumnOffset` argument. Referring to a cell one row up is similar:

```
Range("B2").Offset(RowOffset:=-1)
Range("B2").Offset(-1)
```

Once again, you can choose which one to use. It is a matter of readability of the code.

Suppose you have a list of produce in column A, with totals next to the produce items in column B. If you want to find any total equal to zero and place `LOW` in the cell next to it, do this:

```
Set Rng = Range("B1:B16").Find(What:="0", LookAt:=xlWhole, _
    LookIn:=xlValues)
Rng.Offset(, 1).Value = "LOW"
```

When used in a Sub and looping through a data set, it would look like this:

```
Sub FindLow()
With Range("B1:B16")
    Set Rng = .Find(What:="0", LookAt:=xlWhole, LookIn:=xlValues)
    If Not Rng Is Nothing Then
        firstAddress = Rng.Address
        Do
            Rng.Offset(, 1).Value = "LOW"
            Set Rng = .FindNext(Rng)
        Loop While Not Rng Is Nothing And Rng.Address <> firstAddress
    End If
End With
End Sub
```

The `LOW` totals are noted by the program, as shown in Figure 3.1.

Figure 3.1
Find the produce with zero totals.

	A	B	C
1	Apples	45	
2	Oranges	12	
3	Grapefruit	86	
4	Lemons	0	LOW

NOTE

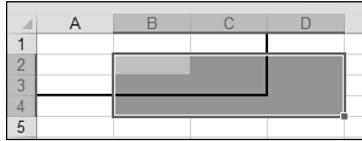
Refer to the section "Object Variables" in Chapter 4, "Looping and Flow Control," for more information on the `Set` statement.

Offsetting isn't only for single cells; you can use it with ranges. You can shift the focus of a range over in the same way you can shift the active cell. The following line refers to B2:D4 (see Figure 3.2):

```
Range("A1:C3").Offset(1,1)
```

Figure 3.2

Offsetting a range:
`Range("A1:C3").`
`Offset(1,1).`
`Select.`



Using the `Resize` Property to Change the Size of a Range

The `Resize` property enables you to change the size of a range based on the location of the active cell. You can create a new range as needed. This is the syntax for the `Resize` property:

```
Range.Resize(RowSize, ColumnSize)
```

To create the range B3:D13, use the following:

```
Range("B3").Resize(RowSize:=11, ColumnSize:=3)
```

Here's a simpler way to create this range:

```
Range("B3").Resize(11, 3)
```

But what if you need to resize by only a row or a column—not both? You don't have to enter both the row and the column parameters.

To expand by two columns, use either of the following:

```
Range("B3").Resize(ColumnSize:=2)
```

or

```
Range("B3").Resize(,2)
```

Both lines mean the same thing. The choice is yours. If you use the second line, make sure to include the comma so Excel knows the `2` refers to the `ColumnSize` argument. Resizing just the rows is similar. You can use either of the following:

```
Range("B3").Resize(RowSize:=2)
```

or

```
Range("B3").Resize(2)
```

Once again, the choice is yours. It is a matter of readability of the code.

From the list of produce, say that you want to find the zero totals and color the cells of the total and corresponding produce (see Figure 3.3). Here's what you do:

```
Set Rng = Range("B1:B16").Find(What:="0", LookAt:=xlWhole, _  
    LookIn:=xlValues)  
Rng.Offset(, -1).Resize(, 2).Interior.ColorIndex = 15
```

Figure 3.3

Resizing a range to extend the selection.

	A	B
1	Apples	45
2	Oranges	12
3	Grapefruit	0
4	Lemons	0
5	Tomatoes	58

Notice that the `Offset` property first moves the active cell over to the produce column. When you are resizing, the upper-left-corner cell must remain the same.

Resizing isn't only for single cells; you can use it to resize an existing range. For example, if you have a named range but need it and the column next to it, use this:

```
Range("Produce").Resize(, 2)
```

Remember, the number you resize by is the total number of rows/columns you want to include.

Using the Columns and Rows Properties to Specify a Range

The `Columns` and `Rows` properties refer to the columns and rows of a specified `Range` object, which can be a worksheet or a range of cells. They return a `Range` object referencing the rows or columns of the specified object.

You have seen the following line used, but what is it doing?

```
FinalRow = Cells(Rows.Count, 1).End(xlUp).Row
```

This line of code finds the last row in a sheet in which column A has a value and places the row number of that `Range` object into the variable called `FinalRow`. This can be useful when you need to loop through a sheet row by row; you will know exactly how many rows you need to go through.

NOTE

Some properties of columns and rows require contiguous rows and columns in order to work properly. For example, if you were to use the following line of code, 9 would be the answer because only the first range would be evaluated:

```
Range("A1:B9, C10:D19").Rows.Count
```

However, if the ranges were grouped separately, the answer would be 19. Excel takes the top, left-most cell address, A1, and the bottom, rightmost cell address, D19, and counts the cells in the range A1:D19:

```
Range("A1:B9", "C10:D19").Rows.Count
```

Using the Union Method to Join Multiple Ranges

The `Union` method enables you to join two or more noncontiguous ranges. It creates a temporary object of the multiple ranges, which enables you to affect them together:

```
Application.Union(argument1, argument2, etc.)
```

The expression `Application` is not required. The following code joins two named ranges on the sheet, inserts the `=RAND()` formula, and bolds them:

```

Set UnionRange = Union(Range("Range1"), Range("Range2"))
With UnionRange
    .Formula = "=RAND()"
    .Font.Bold = True
End With

```

Using the Intersect Method to Create a New Range from Overlapping Ranges

The `Intersect` method returns the cells that overlap between two or more ranges. If there is no overlap, an error will be returned:

```
Application.Intersect(argument1, argument2, etc.)
```

The expression `Application` is not required. The following code colors the overlapping cells of the two ranges:

```

Set IntersectRange = Intersect(Range("Range1"), Range("Range2"))
IntersectRange.Interior.ColorIndex = 6

```

Using the IsEmpty Function to Check Whether a Cell Is Empty

The `IsEmpty` function returns a Boolean value that indicates whether a single cell is empty: `True` if empty, `False` if not. The cell must truly be empty for the function to return `True`. If it contains even just a space that you cannot see, Excel does not consider the cell to be empty:

```
IsEmpty(Cell)
```

Say that you have several groups of data separated by a blank row. You want to make the separations a little more obvious. The following code goes down the data in column A. When it finds an empty cell in column A, it colors in the first four cells of that row (see Figure 3.4):

```

LastRow = Cells(Rows.Count, 1).End(xlUp).Row
For i = 1 To LastRow
    If IsEmpty(Cells(i, 1)) Then
        Cells(i, 1).Resize(1, 4).Interior.ColorIndex = 1
    End If
Next i

```

Figure 3.4
Colored rows separating data.

	A	B	C	D
1	Apples	Oranges	Grapefruit	Lemons
2	45	12	86	15
3	71%	53%	82%	52%
4				
5	Tomatoes	Cabbage	Lettuce	Green Peppers
6	58	24	31	0
7	30%	43%	68%	1%
8				
9	Potatoes	Yams	Onions	Garlic
10	10	61	26	29
11	18%	19%	22%	82%

Using the CurrentRegion Property to Select a Data Range

`CurrentRegion` returns a `Range` object that represents a set of contiguous data. As long as the data is surrounded by one empty row and one empty column, you can select the data set by using `CurrentRegion`:

```
RangeObject.CurrentRegion
```

The following line selects A1:D3 because this is the contiguous range of cells around cell A1 (see Figure 3.5):

```
Range("A1").CurrentRegion.Select
```

This is useful if you have a data set whose size is in constant flux.

Figure 3.5

Use `CurrentRegion` to select a range of contiguous data around the active cell.

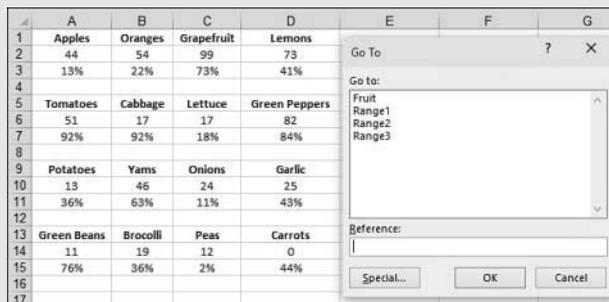
	A	B	C	D
1	Apples	Oranges	Grapefruit	Lemons
2	14	97	84	21
3	31%	47%	29%	77%
4				

CASE STUDY: USING THE SPECIALCELLS METHOD TO SELECT SPECIFIC CELLS

Even Excel power users might not have encountered the Go To Special dialog box. If you press the F5 key in an Excel worksheet, you get the normal Go To dialog box (see Figure 3.6). In the lower-left corner of this dialog is a button labeled Special. Click this button to get to the super-powerful Go To Special dialog box (see Figure 3.7).

Figure 3.6

Although the Go To dialog doesn't seem useful, click the Special button in the lower-left corner.



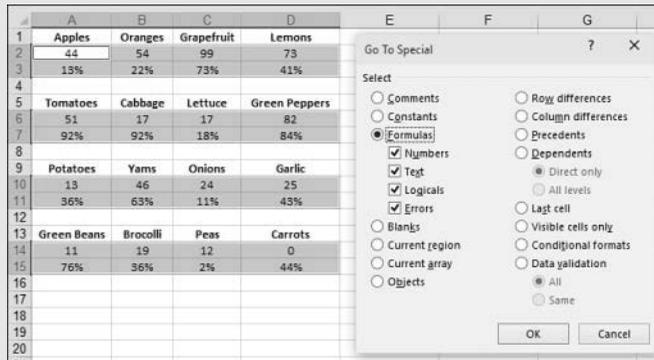
In the Excel interface, the Go To Special dialog enables you to select only cells with formulas, only blank cells, or only the visible cells. Selecting only visible cells is excellent for grabbing the visible results of AutoFiltered data.

To simulate the Go To Special dialog in VBA, use the `SpecialCells` method. This enables you to act on cells that meet certain criteria, like this:

```
RangeObject.SpecialCells(Type, Value)
```

Figure 3.7

The Go To Special dialog has many incredibly useful selection tools, such as one for selecting only the formulas on a sheet.



This method has two parameters: `Type` and `Value`. `Type` is one of the `xlCellType` constants:

```
xlCellTypeAllFormatConditions
xlCellTypeAllValidation
xlCellTypeBlanks
xlCellTypeComments
xlCellTypeConstants
xlCellTypeFormulas
xlCellTypeLastCell
xlCellTypeSameFormatConditions
xlCellTypeSameValidation
xlCellTypeVisible
```

`Value` is optional and can be one of the following:

```
xlErrors
xlLogical
xlNumbers
xlTextValues
```

The following code returns all the ranges that have conditional formatting set up. It produces an error if there are no conditional formats and adds a border around each contiguous section it finds:

```
Set rngCond = ActiveSheet.Cells.SpecialCells(xlCellTypeAllFormatConditions)
If Not rngCond Is Nothing Then
    rngCond.BorderAround xlContinuous
End If
```

Have you ever had someone send you a worksheet without all the labels filled in? Some people think that the data shown in Figure 3.8 looks neat. They enter the `Region` field only once for each region. This might look aesthetically pleasing, but it is impossible to sort.

Figure 3.8

The blank cells in the `Region` column make it difficult to sort data sets such as this.

	A	B	C
1	Region	Product	Sales
2	North	ABC	766,469
3		DEF	776,996
4		XYZ	832,414
5	East	ABC	703,255
6		DEF	891,799
7		XYZ	897,949

Using the `SpecialCells` method to select all the blanks in this range is one way to fill in all the blank region cells quickly with the region found above them:

```
Sub FillIn()
On Error Resume Next 'Need this because if there aren't any blank
'cells, the code will error
Range("A1").CurrentRegion.SpecialCells(xlCellTypeBlanks).FormulaR1C1 _
    = "=R[-1]C"
Range("A1").CurrentRegion.Value = Range("A1").CurrentRegion.Value
End Sub
```

In this code, `Range("A1").CurrentRegion` refers to the contiguous range of data in the report. The `SpecialCells` method returns just the blank cells in that range. This particular formula fills in all the blank cells with a formula that points to the cell above the blank cell. (You can read more about R1C1-style formulas in Chapter 5, "R1C1-Style Formulas.") The second line of code is a fast way to simulate doing a Copy and then Paste Special Values. Figure 3.9 shows the results.

Figure 3.9

After the macro runs, the blank cells in the Region column have been filled in with data.

	A	B	C
1	Region	Product	Sales
2	North	ABC	766,469
3	North	DEF	776,996
4	North	XYZ	832,414
5	East	ABC	703,255
6	East	DEF	891,799
7	East	XYZ	897,949

Using the `Areas` Collection to Return a Noncontiguous Range

The `Areas` collection is a collection of noncontiguous ranges within a selection. It consists of individual `Range` objects representing contiguous ranges of cells within the selection. If a selection contains only one area, the `Areas` collection contains a single `Range` object that corresponds to that selection.

You might be tempted to loop through the rows in a sheet and check the properties of a cell in a row, such as its formatting (for example, font or fill) or whether the cell contains a formula or value. Then you could copy the row and paste it to another section. However, there is an easier way. In Figure 3.10, the user enters the values below each fruit and vegetable. The percentages are formulas. The following line of code selects the cells with numeric constants and copies them to another area:

```
Range("A:D").SpecialCells(xlCellTypeConstants, xlNumbers).Copy _
    Range("I1")
```

Figure 3.10

The `Areas` collection makes it easier to manipulate noncontiguous ranges.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Apples	Oranges	Grapefruit	Lemons					45	12	86	15
2	45	12	86	15					58	24	31	0
3	6%	65%	78%	45%					10	61	26	29
4									46	64	79	95
5	Tomatoes	Cabbage	Lettuce	Green Peppers								
6	58	24	31	0								
7	22%	31%	70%	65%								
8												
9	Potatoes	Yams	Onions	Garlic								
10	10	61	26	29								
11	18%	49%	57%	86%								
12												
13	Green Beans	Broccoli	Peas	Carrots								
14	46	64	79	95								
15	27%	56%	21%	42%								

Referencing Tables

A table is a special type of range that offers the convenience of referencing named ranges. However, tables are not created in the same manner as other ranges. For more information on how to create a named table, see Chapter 6, “Creating and Manipulating Names in VBA.”

Although you can reference a table by using `Worksheets(1).Range("Table1")`, you have access to more of the properties and methods that are unique to tables if you use the `ListObjects` object, like this:

```
Worksheets(1).ListObjects("Table1")
```

This opens the properties and methods of a table, but you can't use that line to select the table. To do that, you have to specify the part of the table you want to work with. To select the entire table, including the header and total rows, specify the `Range` property:

```
Worksheets(1).ListObjects("Table1").Range.Select
```

The table part properties include the following:

- `Range`—Returns the entire table.
- `DataBodyRange`—Returns the data part only.
- `HeaderRowRange`—Returns the header row only.
- `TotalRowRange`—Returns the total row only.

What I really like about coding with tables is the ease of referencing specific columns of a table. You don't have to know how many columns to move in from a starting position or the letter/number of the column, and you don't have to use a `FIND` function. Instead, you can use the header name of the column. For example, to select the data of the `Qty` column of the table, but not the header or total rows, do this:

```
Worksheets(1).ListObjects("Table1").ListColumns("Qty")._
    .DataBodyRange.Select
```

NOTE

For more details on coding with tables, check out *Excel Tables: A Complete Guide for Creating, Using, and Automating Lists and Tables* by Zack Barresse and Kevin Jones (ISBN: 978-1615470280).

Next Steps

Chapter 4 describes a fundamental component of any programming language: loops. If you have taken a programming class, you will be familiar with basic loop structures. VBA supports all the usual loops. That chapter also describes a special loop, `For Each...Next`, which is unique to object-oriented programming such as VBA.

Index

Symbols

{ } (curly braces), JavaScript interactivity in Office (MS) add-ins, 527

:= (parameters), VBA syntax, 35-37

; (semicolons), JavaScript interactivity in Office (MS) add-ins, 527

Numbers

32-bit compatible API declarations, 465-466

64-bit compatible API declarations, 465-466

A

A1 formulas

A1 versus, 93-97

autofilling data, 95-96

case study, 96-97

replacing multiple A1 formulas with one R1C1 formula, 99-101

About dialog, customizing via API declarations, 469-470

above/below average rules, 334, 348

absolute references and R1C1 formulas, 98

accelerator keys (userforms), showing, 456

Access (MS)

ADO, 424, 426-427, 434

adding fields “on the fly,” 436-437

adding records to MDB, 427-428

adding tables “on the fly,” 436

connections, 426

- cursors, 426
- deleting records from MDB, 433
- lock type, 426
- record sets, 426
- retrieving records from MDB, 429-430
- summarizing records, 433-434
- updating existing MDB records, 431-432
- verifying field existence, 435-436
- verifying table existence, 434-435
- MDB, 423-424
 - adding records to, 427-428
 - deleting records via ADO, 433
 - retrieving records from, 429-430
 - summarizing records via ADO, 433-434
 - updating existing records, 431-432
- shared access databases, creating (case study), 425-426
- SQL server and, 437-438
- accounting reports case study, 3-4**
- Action option (Advanced Filter), 186**
- ActiveX Controls, attaching macros to, 506-507**
- AddChart2 and chart creation, 311-312**
- add-ins, 509**
 - case study, 515-516
 - characteristics of, 509-510
 - client installations, 512-514
 - closing, 514
 - Office (MS) add-ins, 517
 - buttons (HTML), 524-525
 - creating, 517-521
 - .CSS files, 525
 - “Hello World” add-in, 517-521
 - HTML in, 524-525
 - interactivity in, 521-524
 - JavaScript interactivity in, 526-536
 - Napa Office 365 development tools, 536-537
 - tags (HTML), 524
 - writing to content pane, 535
 - writing to task pane, 535
 - XML in, 525-526
- removing, 514-515
- security, 513-514
- workbooks
 - converting to add-ins, 510-512
 - hidden workbooks as alternative to add-ins, 515-516
- Add-ins group (Developer tab), 10**
- addresses**
 - cell addresses, returning column letters from, 306
 - duplicate max values, returning the addresses of, 304-305
 - email addresses, validating, 292-293
 - hyperlink addresses, returning, 305-306
- ADO (ActiveX Data Objects), 426-427, 434**
 - connections, 426
 - cursors, 426
 - location, 426
 - types of, 426
 - DAO versus, 424-425
 - fields
 - adding fields “on the fly,” 436-437
 - verifying existence of, 435-436
 - lock type, 426
 - MDB
 - deleting records from MDB, 433
 - summarizing records, 433-434
 - record sets, 426

- tables
 - adding tables “on the fly,” 436
 - verifying existence of, 434-435
- Advanced Filter, 177, 184-185**
 - Action option, 186
 - building via Excel interface, 185-186
 - case studies
 - criteria ranges, 194-197
 - multiple Advanced Filters, 206-209
 - criteria ranges, 186, 192-194
 - case study, 194-197
 - complex criteria, 194-195
 - formula-based conditions, 196-201
 - joining multiple ranges via Logical AND, 194
 - joining multiple ranges via Logical OR, 193-194
 - Filter in Place, 186, 201-202
 - catching no records with filter in place, 202
 - showing all records, 202
 - multiple Advanced Filters, 206-209
 - unique lists, 186
 - extracting values via user interface, 186-187
 - extracting values via VBA code, 187-191
 - unique combinations of two or more fields, 191-192
 - xlFilterCopy, 203
 - copying columns, 203-204
 - copying subsets of columns, 204-206
 - reordering columns, 204-206
- alpha characters, sorting, 302-303**
- analyzing data via Application.OnTime method, 381**
 - macros
 - canceling all pending scheduled macros, 383
 - canceling previously scheduled macros, 382
 - scheduling macros to run every 2 minutes, 384-385
 - scheduling macros to run x minutes in the future, 383
 - scheduling verbal reminders, 383-384
 - Ready mode for scheduled procedures, 381-382
 - updates, scheduling window of time for, 382
- Anderson, Ken, 390**
- API declarations, 463-464**
 - 32-bit compatible declarations, 465-466
 - 64-bit compatible declarations, 465-466
 - checking Excel file open status on network, 467-468
 - computer names, 467
 - creating running timers, 471
 - customizing About dialog, 469-470
 - disabling X button for closing user-forms, 470-471
 - display resolution information, 468-469
 - example of, 464-465
 - playing sounds, 472
 - private versus public status, 464-465
 - types of, 464
- application-level events**
 - class modules, 125, 140-141
 - list of, 125-130

Application.OnTime method and data analysis, 381

macros

- canceling all pending scheduled macros, 383

- canceling previously scheduled macros, 382

- scheduling macros to run every 2 minutes, 384-385

- scheduling macros to run x minutes in the future, 383

- scheduling verbal reminders, 383-384

- Ready mode for scheduled procedures, 381-382

- updates, scheduling window of time for, 382

Archibald, Rory, 254**Areas collection, returning noncontiguous ranges, 70-71****arrays, 131**

- declaring, 131-132

- dynamic arrays, 136-137

- filling, 133-134

- JavaScript arrays in Office (MS) add-ins, 528-529

- multidimensional arrays, declaring, 132-133

- names, 109-110

- passing, 137-138

- R1C1 formulas and, 101-102

- retrieving data from, 134-135

- speeding up code, 135-136

- variant variables, 133

assigning macros to

- Form Controls, 18-19

- shapes, 18-19

- text boxes, 18-19

assignment operators (JavaScript) and Office (MS) add-ins, 532-533**Atlas Programming Management, 260****audio (sound), playing via API declarations, 472****autofilling data**

- A1 formulas, 95-96

- R1C1 formulas, 96

AutoFilter, 180

- color, filtering by, 181

- dynamic date ranges, selecting, 182-183

- icon, filtering by, 181-182

- loops, replacing, 177-179

- multiple items, selecting, 180

- search box, selecting via, 180-181

- turning off drop-downs in, 209-210

- visible cells, selecting, 183-184

AutoSort, controlling pivot table sort orders, 225**AutoSum, recording macros, 30-31**

B

backing up/moving forward in code (debugging), 45-46**backward compatibility and charts, 331****Barresse, Zack, 71, 258, 279****BASIC**

- example of, 33-34

- VBA versus, 8, 33-34

binary event tracking in win/loss charts (sparklines), 368-369**binding Word objects**

- early binding, 406-408

- early binding using New keyword, 409

- late binding, 408-409

blanks/errors formatting in cells, 351-352

bookmarks (Word), 419-420

breakpoints (debugging code), 45, 49

Bricklin, Dan, 93

bug fixes. See debugging code

buttons

command buttons, attaching macros to, 504-505

HTML buttons and Office (MS) add-ins, 524-525

images, adding to buttons, 497

custom icon images, 499-500

Microsoft Office icons, 498-499

macro buttons

creating on Quick Access Toolbar, 17

creating on Ribbon, 16

userforms

option buttons, 167-169

spin buttons, 170-171

C

calculations, changing to show percentages, 222-224

calling userforms, 159-160

case (text), changing, 277-278

Case statements, complex expressions in (Select Case.End Select loops), 89

case studies

A1 formulas, 96-97

accounting reports, 3-4

add-ins, 515-516

Advanced Filter

criteria ranges, 196-197

multiple Advanced Filters, 206-209

cells in ranges, selecting specific, 68-70

charts, 327-330

cleaning up recorded code, 55-57

criteria ranges (Advanced Filter), 194-197

error handling (troubleshooting), 480

events, 122

Go to Special versus looping, 184

help buttons, 151-153

looping

through file directories, 84-85

versus Go to Special, 184

macros

AutoSum, 30-31

Quick Analysis, 30-31

recording, 21-24

relative references when recording macros, 26-30

testing, 24-25

named ranges for Vlookup, 112-113

password cracking and code security, 484-485

pivot tables

data visualization, 249-250

filtering case study, 233-235

R1C1 formulas, 96-97

ranges

named ranges for Vlookup, 112-113

selecting specific cells in, 68-70

relative references when recording macros, 26-30

shared access databases, creating, 425-426

troubleshooting (error handling), 480

UDF, creating, 284-285

userforms

controls, 162-163

multicolumn list boxes, 459

Cell Masters website, 254**cells**

- above/below average rules, 334, 348
- case studies
 - entering military time in cells, 122
 - selecting specific cells in ranges, 68-70
- column letters, returning from cell addresses, 306
- comments
 - placing charts in comments, 261-262
 - resizing, 260-261
- data bars, 334
 - adding to ranges, 335-339
 - multiple colors of data bars in ranges, 345-347
- duplicate cell formatting, 349-350
- empty cells
 - checking for empty cells in ranges, 67
 - deleting empty cells from values area, 225
- file paths, setting in cells, 287
- formatting
 - above/below average rules, 348
 - blanks/errors formatting, 351-352
 - date formatting, 351
 - duplicate cell formatting, 349-350
 - formulas to determine cells to format, 352-353
 - text formatting, 351
 - top/bottom rules, 348-349
 - unique cells, 349-350
 - value-based formatting, 350-351
- highlighting, 334
- military time, entering (case study), 122

- noncontiguous cells, selecting, 265-267
- nonzero-length cells, finding in ranges, 296-297
- progress indicators, creating, 274-275
- ranges
 - cells versus ranges when cleaning up code, 52
 - finding nonzero-length cells in, 296-297
 - selecting specific cells in, 68-70
 - selecting via cells, 62-63
- reversing cell contents, 304
- selecting, 263
 - creating hidden log files, 267-268
 - highlighting selected cells using conditional formatting, 263-264
 - highlighting selected cells without using conditional formatting, 264-265
 - noncontiguous cells, 265-267
 - via SpecialCells, 279
- SpecialCells, selecting cells via, 279
- specific cells, selecting in ranges, 68-70
- summing based on interior color, 293-294
- top/bottom rules, 334, 348-349
- unique cells, formatting, 349-350
- visible cells, selecting via AutoFilter, 183-184
- workbook names, setting in cells, 286-287

charts, 309

- AddChart2 and chart creation, 311-312
- backward compatibility, 331
- case study, 327-330
- combo charts, creating, 327-330

- creating via AddChart2, 311-312
- embedded charts
 - class modules, 141-143
 - events and class modules, 123
 - events list, 124
- events, 119, 123
 - embedded charts and class modules, 123, 141-143
 - embedded charts events, 124
- Excel
 - changes to, 541
 - planning for migration, 310-311
- exporting as graphics, 330
- filtering, 318
- formatting, 312-315
 - applying color, 317-318
 - changing object fills, 325-327
 - emulating Plus icon changes via SetElement, 319-323
 - filtering charts, 318
 - line settings, 327
 - micromanaging formatting changes, 324-325
 - specific chart references, 315-316
 - specifying titles, 316
- good/bad of the VBA creation process, 309-310
- placing in cell comments, 261-262
- styles of, 312-315
- types of, 313-315
- win/loss charts (sparklines), 355
 - binary event tracking, 368-369
 - formatting, 361-369
 - RGB colors, 364-365
 - scaling, 357-361
 - theme colors, 361-364
- Checkbox controls (userforms), 440-441**
- class modules, 139**
 - application-level events, 125, 140-141
 - collections, creating in, 148-149
 - embedded chart events, 123, 141-143
 - Excel State class modules, creating, 268-270
 - inserting into objects, 139-140
- cleaning up code, 51**
 - case study, 55-57
 - cells versus ranges, 52
 - copying/pasting in statements, 54
 - With.End With blocks for multiple actions, 54-55
 - formulas
 - avoiding hard-coding formulas, 53-54
 - R1C1 formulas, 54
 - multiple actions in recorded code, 54-55
 - ranges versus cells, 52
 - rows
 - avoiding hard-coding rows, 53-54
 - finding the last row, 52-53
 - selecting things, 51
- clients**
 - add-in installations, 512-514
 - error handling (troubleshooting) procedures, training in, 481-482
- closing**
 - add-ins, 514
 - userforms
 - closing windows illegally, 174-175
 - disabling X button via API declarations, 470-471
- code files, 5**
- Code group**
 - Developer tab, 10
 - Macro Security icon, 10, 12

- Macros icon, 10
- Record Macro icon, 10
- Use Relative References icon, 10
- Visual Basic Editor, opening, 10
- Visual Basic icon, 10

collections (VBA), 35-37, 139, 145. *See also dictionaries*

- Areas collection, returning noncontiguous ranges, 70-71
- collections, 37
- creating, 146
 - in class modules, 148-149
 - in standard modules, 146-147
- userform controls and, 447-449

color

- charts, applying to, 317-318
- color scales, 334, 339-340
- data bars, multiple colors of, 345-347
- filtering by (AutoFilter), 181
- ranges
 - adding color scales to, 339-340
 - multiple colors of data bars in ranges, 345-347
- sparkline formatting
 - RGB colors, 364-365
 - theme colors, 361-364
- summing cells based on interior color, 293-294
- userforms, coloring active controls, 457-459

columns

- column letters, returning from cell addresses, 306
- copying
 - subsets of columns via xlFilterCopy (Advanced Filter), 204-206
 - xlFilterCopy (Advanced Filter), 203-204

- multicolumn list boxes in userforms (case study), 459

R1C1 formulas

- column number/letter associations, 101
- column references, 99
- ranges, specifying via columns, 66
- reordering via xlFilterCopy (Advanced Filter), 204-206
- spark columns, 355
 - creating, 356-357
 - formatting, 361-369
 - RGB colors, 364-365
 - scaling, 357-361
 - theme colors, 361-364

combining

- userforms, 171-173
- workbooks, 256-257

combo charts, creating, 327-330

command buttons

- macros, attaching to command buttons, 504-505
- userforms, 163-165

comments

- cell comments
 - placing charts in, 261-262
 - resizing, 260-261
- JavaScript interactivity in Office (MS) add-ins, 527
- names, adding to, 106

compatibility

- backward compatibility and charts, 331
- Excel compatibility issues, 542-543
 - Excel8CompatibilityMode property, 543-544
 - Version property, 543

concatenating data (sorting and), 300-302

conditional formatting

- data visualization, 334
- highlighting selected cells, 263-264

configuring

- pivot tables, 213-214
- slicers for pivot table filtering, 235-239

connections (ADO), 426**constants (defined) in VBA, 40-43****content management (web page data) via Excel, 387-389****content pane (Office add-ins), writing to, 535****controls**

- ActiveX Controls, attaching macros to, 506-507
- Ribbon, customizing for running macros
 - adding controls to Ribbon, 490-491
 - control attributes, 490
 - required arguments, 491
- userforms, 440
 - adding, 453
 - adding at runtime, 450-456
 - adding controls to existing forms, 162-163
 - adding “on the fly,” 452
 - adding tip text, 457
 - case study, 162-163
 - Checkbox controls, 440-441
 - collections and, 447-449
 - coloring active controls, 457-459
 - designing via toolbar, 439-440
 - MultiPage control and combining userforms, 171-173
 - programming, 162
 - RefEdit controls, 444

Scrollbar controls, 446

TabStrip controls, 442-443

ToggleButton controls, 444

Controls group (Developer tab), 10**converting**

- files to add-ins
 - Save As method, 511
 - VB Editor, 511-512
- pivot tables to values, 217-219
- week numbers to dates, 299
- workbooks to add-ins, 510-511
 - Save As method, 511
 - VB Editor, 511-512

copying

- columns via xlFilterCopy (Advanced Filter), 203-204
- data to
 - separate worksheets, 257-258
 - separate worksheets without filters, 258-259
- reports for every product (pivot tables), 225-228
- within statements (cleaning up recorded code), 54
- subsets of columns via xlFilterCopy (Advanced Filter), 204-206

counting unique values, 294-295**CreateObject function, creating new instances of Word objects, 409****criteria ranges (Advanced Filter), 186, 192-194**

- case studies, 184, 196-197
- complex criteria, 194-195
- formula-based conditions
 - case study, 196-197
 - returning above-average records, 201
 - setting up conditions, 196
- VBA, 197-201

joining multiple ranges

Logical AND, 194

Logical OR, 193-194

.CSS files and Office (MS) add-ins, 525

.CSV files

deleting, 254

importing, 254

curly braces ({}), JavaScript interactivity in Office (MS) add-ins, 527

CurrentRegion property, selecting data ranges, 68

cursors

ADO, 426

location, 426

types of, 426

debugging code

hovering the cursor, 47-48

running to cursor, 46

custom functions. See UDF

custom objects

creating, 143-145

using, 145

custom properties, creating via UDT (User-Defined Types), 153-156

customizing Ribbon for running macros, 487-488

accessing Excel file structure, 496

adding controls to Ribbon, 490-491

creating

groups, 489-490

tabs, 489-490

customui folder and file, 488-489

images on buttons, 497-500

.RELS files, 496-497

renaming Excel files, 497

troubleshooting (error handling), 500-503

D

daily dates, grouping to months, quarters, years via pivot tables, 221-222

DAO (Data Access Objects) versus ADO, 424-425

dashboards, creating via sparklines, 369-373

data analysis via Application.OnTime method, 381

macros

canceling all pending scheduled macros, 383

canceling previously scheduled macros, 382

scheduling macros to run every 2 minutes, 384-385

scheduling macros to run x minutes in the future, 383

scheduling verbal reminders, 383-384

Ready mode for scheduled procedures, 381-382

updates, scheduling window of time for, 382

data bars, 334

adding to ranges, 335-339

multiple colors of data bars in ranges, 345-347

Data Model, 242

creating, 245-247

loading large text files to Data Model via Power Query, 401-402

pivot tables

adding model fields to pivot tables, 244

adding numeric fields to value area, 244-245

adding tables to Data Model, 242-243

- building pivot tables, 243
- creating relationships between pivot tables, 243
- defining pivot caches, 243
- data sets**
 - duplicate value rules, 334
 - navigating in Macro Recorder, 31
- data visualization, 333**
 - above/below average rules, 334, 348
 - cells
 - blanks/errors formatting, 351-352
 - date formatting, 351
 - duplicate cell formatting, 349-350
 - formulas to determine cells to format, 352-353
 - text formatting, 351
 - unique cell formatting, 349-350
 - value-based formatting, 350-351
 - color scales, 334, 339-340
 - conditional formatting, 334
 - data bars, 334
 - adding to ranges, 335-339
 - multiple colors of data bars in ranges, 345-347
 - duplicate value rules, 334
 - highlight cell rules, 334
 - icon sets, 334
 - adding to ranges, 341-343
 - creating for subsets of ranges, 344-345
 - methods, 334-335
 - NumberFormat property, 353-354
 - pivot tables case study, 249-250
 - properties, 334-335
 - ranges
 - adding data bars to, 335-339
 - creating icon sets for subsets of ranges, 344-345
 - highlighting unique values in ranges, 352-353
 - multiple colors of data bars in ranges, 345-347
 - rows, highlighting for the largest value, 353
 - top/bottom rules, 334, 348-349
- databases**
 - MDB, 423-426
 - shared access databases, creating (case study), 425-426
- date and time**
 - cells dates, formatting, 351
 - daily dates, grouping to months, quarters, years via pivot tables, 221-222
 - retrieving
 - from last save, 291
 - permanent date and time, 291
 - week numbers, converting to dates, 299
- date ranges (dynamic), selecting via AutoFilter, 182-183**
- debugging code**
 - backing up/moving forward in code, 45-46
 - breakpoints, 45, 49
 - controls, adding to userforms, 162-163
 - error handling (troubleshooting) and code security, 484
 - querying while stepping through code, 46
 - hovering the cursor, 47-48
 - Immediate window, 46-47
 - watches, 48
 - running to cursor, 46
 - stepping through code, 43-45

declarations (API)

- 32-bit compatible declarations, 465-466
- 64-bit compatible declarations, 465-466
- About dialog, customizing, 469-470
- computer names, retrieving, 467
- display resolution information, retrieving, 468-469
- example of, 464, 465
- Excel file open status, checking in network, 467-468
- private versus public status, 464-465
- running timers, creating, 471
- sounds, playing, 472
- types of, 464
- usage example, 465
- X button for closing userforms, disabling, 470-471

declaring

- arrays, 131-132
- multidimensional arrays, 132-133

defined constants (VBA), 40-43**deleting**

- empty cells from values area (pivot tables), 225
- names, 105-106
- records from MDB, 433

delimited files (imported), opening, 395-397**delimited strings, extracting a single element from, 300****Developer tab**

- accessing, 9-10
- Add-ins group, 10
- Code group, 10
- Controls group, 10
- Modify group, 10
- XML group, 10

development stage errors versus errors months later (error handling/troubleshooting), 482

- Runtime Error 9: Subscript Out of Range, 482-483
- Runtime Error 1004: Method Range of Object Global Failed, 483-484

dictionaries. *See also* collections (VBA), 150-151**digital signatures, Disable All Macros Except Digitally Signed Macros option (macro security), 14****directories**

- listing files in, 251-253
- looping case study, 84-85
- workbooks, counting number of workbooks in a directory, 288-289

Disable All Macros Except Digitally Signed Macros option (macro security), 14**Disable All Macros with Notification option (macro security), 13-14****Disable All Macros Without Notification option (macro security), 13****Do loops, 78-80**

- Until clause and, 81-82
- While clause and, 81-82
- While.Wend loops, 82

Document object (Word), 413-415**downloading code files, 5****duplicate cells, formatting, 349-350****duplicate max values, returning the addresses of, 304-305****duplicate value rules, 334****duplicates, removing from ranges, 295-296****dynamic arrays, 136-137****dynamic date ranges, selecting via AutoFilter, 182-183**

E

early binding, referencing Word objects via, 406-409

editing macros, 19

- defined constants, 40-43
- optional parameters, 40
- Project Explorer, 20
- properties that return objects, 43
- Properties window, 21
- settings, 20

email addresses, validating, 292-293

embedded charts and events

- class modules, 123, 141-143
- list of events, 124

empty cells, deleting from values area (pivot tables), 225

Enable All Macros option (macro security), 14

error handling (troubleshooting), 473

- blanks/errors formatting, 351-352
- case study, 480
- clients, training, 481-482
- code security
 - debugging and, 484
 - locking code, 485-486
 - password cracking (case study), 484-485
- debug errors in userforms, 475-477
- development stage errors versus errors months later, 482
- encountering errors on purpose, 481
- On Error GoTo syntax, 477-478
- Excel warnings, suppressing, 481
- generic error handlers, 478-479
- ignoring errors, 479
- Ribbon, customizing for running macros, 500-503

Runtime Error 9: Subscript Out of Range, 482-483

Runtime Error 1004: Method Range of Object Global Failed, 483-484

VBA, 473-475

versioning errors, 486

events, 115

- accessing, 116
- application-level events
 - class modules, 125, 140-141
 - list of, 125-130
- case study, 122
- chart events, 119, 123
 - embedded charts and class modules, 123, 141-143
 - embedded charts events, 124
- enabling, 117
- frame control events in userforms, 167-169
- graphic control events, 169
- levels of, 115-116
- military time, entering in cells, 122
- parameters, 116
- QueryClose events, userform windows, 174-175
- userform events, 160-161
- viewing, 116
- workbooks
 - events, 117
 - sheet events, 119
- worksheet events, 120

Excel

- Advanced Filter, building, 185-186
- charts
 - planning for Excel migration, 310-311
 - version changes to, 541

- compatibility issues, 542-543
 - Excel8CompatibilityMode property, 543-544
 - Version property, 543
- content management (web page data), 387-389
- file structure, accessing, 496
- FTP, 389-390
- Macro Recorder, 8
- pivot tables, version changes to, 541
- power programming
 - combining workbooks, 256-257
 - copying data to separate worksheets, 257-258
 - copying data to separate worksheets without filters, 258-259
 - creating hidden log files, 267-268
 - deleting .CSV files, 254
 - exporting data to .XML files, 259-260
 - filtering data to separate worksheets, 257-258
 - highlighting selected cells using conditional formatting, 263-265
 - importing .CSV files, 254
 - listing files in directories, 251-253
 - parsing text files, 254-255
 - placing charts in comments, 261-262
 - reading text files into memory, 254-255
 - resizing cell comments, 260-261
 - selecting noncontiguous cells, 265-267
 - separating worksheets into workbooks, 255-256
- Quick Analysis tool, version changes to, 541
- .RELS files and ribbon customization, 496-497
- renaming files, 497
- Ribbon, version changes to, 539
- SDI, version changes to, 540
- slicers, version changes to, 541
- SmartArt, version changes to, 542
- VBEVB Editor
 - learning new methods, 542
 - learning new objects, 542
- versioning errors, 486
- versions of, 4
 - chart changes, 541
 - learning new methods, 542
 - learning new objects, 542
 - pivot table changes, 541
 - Quick Analysis tool changes, 541
 - Ribbon changes, 539
 - SDI changes, 540
 - slicer changes, 541
 - SmartArt changes, 542
 - warnings, suppressing, 481
- Excel 97-2003 Workbook (.xls) files and macros, 11**
- Excel Binary Workbook (.xlsb) files and macros, 11**
- Excel Macro-Enable Workbook (.xlsm) files**
 - concerns with, 11
 - macros, 11
 - public perception of, 11
 - saving files in, 11
- Excel State class modules, creating, 268-270**
- Excel Tables: A Complete Guide for Creating, Using, and Automating Lists and Tables, 71***
- Excel Workbook (.xlsx) files and macros, 10-11**
- Excel8CompatibilityMode property, Excel compatibility issues, 543-544**

ExcelMatters.com website, 254

exiting loops early, 77-78

exp.com, 271

exporting

charts as graphics, 330

data to .XML files, 259-260

extracting single element from a delimited string, 300

F

field entry, verifying in userforms, 174

files

converting to add-ins

Save As method, 511

VB Editor, 511-512

.CSS files and Office (MS) add-ins, 525

.CSV files

deleting, 254

importing, 254

delimited files (imported), opening, 395-397

directories

listing files in, 251-253

looping case study, 84-85

Excel files

accessing file structure, 496

renaming, 497

fixed-width files (imported), opening, 392-395

hidden log files, creating, 267-268

naming, userforms and, 175-176

paths, setting in cells, 287

.RELS files and ribbon customization, 496-497

text files, 391

delimited files (imported), 395-397

fixed-width files (imported), 392-395

importing, 391-402

importing files with less than 1,048,576 rows, 391-397

importing files with more than 1,048,576 rows, 398-402

loading large files to Data Model via Power Query, 401-402

parsing, 254-255

reading into memory, 254-255

running files a row at a time, 398-400

writing, 402

.XML files, exporting data to, 259-260

fills (object), changing in charts, 325-327

Filter in Place (Advanced Filter), 186, 201-202

catching no records with filter in place, 202

showing all records, 202

filtering

Advanced Filter, 184-185

Action option, 186

building via Excel interface, 185-186

criteria ranges, 186

Filter in Place, 186, 201-202

multiple Advanced Filters (case study), 206-209

unique lists, 186-192

xlFilterCopy, 203-206

AutoFilter

filtering by color, 181

filtering by icon, 181-182

turning off drop-downs in, 209-210

charts, 318

data to separate worksheets, 257-258

pivot tables

- case study, 233-235
- conceptual filters, 229-230
- configuring filtering timelines, 239-242
- configuring slicers for filtering, 235-239
- filtering OLAP pivot tables by lists of items, 271-273
- manually filtering multiple items in pivot fields, 228-229
- search filter, 233
- types of filters, 230

finding

- first nonzero-length cell in a range, 296-297
- last row (cleaning up recorded code), 52-53

fixed-width files (imported), opening, 392-395**flow control**

- If.ElseIf.End loops, 87-88
- If.Then.Else loops, 86
- If.Then.Else.End If loops, 87
- If.Then.End If loops, 86-87
- Select Case.End Select loops, 88
 - complex expressions in Case statements, 89
 - nesting If statements, 89-91

For Each. loops, 82-83**For each.next statements, JavaScript interactivity in Office (MS) add-ins, 532****for loops, JavaScript interactivity in Office (MS) add-ins, 529-530****Form Controls, assigning macros to, 18-19****form fields, automating Word, 420-422****Format method, micromanaging formatting changes in charts, 324-325****formatting****cells**

- above/below average rules, 348
- blanks/errors formatting, 351-352
- date formatting, 351
- duplicate cell formatting, 349-350
- formulas to determine cells to format, 352-353
- text formatting, 351
- top/bottom rules, 348-349
- unique cells, 349-350
- value-based formatting, 350-351

charts, 312-315

- applying color, 317-318
- changing object fills, 325-327
- emulating Plus icon changes via SetElement, 319-323
- filtering charts, 318
- line settings, 327
- micromanaging formatting changes, 324-325
- specific chart references, 315-316
- specifying titles, 316

conditional formatting

- data visualization, 334
- highlighting selected cells, 263-264

sparklines, 361

- RGB colors, 364-365
- theme colors, 361-364
- tables, resetting formatting, 279-280

formula-based conditions and criteria ranges (Advanced Filter)

- case study, 196-197
- returning above-average records, 201
- setting up conditions, 196
- VBA, 197-201

formulas

A1 formulas

autofilling data, 95-96

case study, 96-97

R1C1 versus, 93-97

replacing multiple A1 formulas
with one R1C1 formula, 99-101array formulas and R1C1 formulas,
101-102hard-coding formulas, avoiding
(cleaning up recorded code), 53-54

names, 106-107

R1C1 formulas

A1 versus, 93-97

absolute references and, 98

accessing, 94-95

array formulas and, 101-102

autofilling data, 96

case study, 96-97

cleaning up recorded code, 54

column number/letter
associations, 101

column references, 99

mixed references and, 98-99

relative references and, 97-98

replacing multiple A1 formulas
with one R1C1 formula, 99-101

row references, 99

For.Next loops, 73-76

exiting early, 77-78

nested loops, 78

running backwards, 77

For statement variables, 76

Step clause and, 76-77

variations on, 76-77

**frame control events in userforms,
167-169****Frankston, Bob, 93****FTP from Excel, 389-390****functions**CreateObject function, creating new
instances of Word objects, 409custom functions. *See* UDFGetObject function, referencing
existing instances of Word objects,
410-411IsEmpty function, checking for empty
cells in ranges, 67JavaScript functions in Office (MS)
add-ins, 526, 534-535

UDF, 283, 286

alpha characters, 302-303

case study, 284-285

checking existence of a worksheet,
287-288

checking workbook open status, 287

concatenating data (sorting and),
300-302converting week numbers to
dates, 299counting number of workbooks in
a directory, 288-289

counting unique values, 294-295

creating

extracting a single element from a
delimited string, 300finding the first nonzero-length
cell in a range, 296-297removing duplicates from ranges,
295-296retrieving date and time from last
save, 291retrieving numbers from mixed
text, 298-299retrieving permanent date and
time, 291

retrieving user ID, 289-290

- returning addresses of duplicate max values, 304-305
- returning column letters from cell addresses, 306
- returning hyperlink addresses, 305-306
- reversing cell contents, 304
- searching for a string within text, 303-304
- Select.Case statements in worksheets, 307
- setting workbook names and file paths in cells, 287
- setting workbook names in cells, 286
- sharing, 286
- sorting and concatenating data, 300-302
- sorting numeric characters, 302-303
- static random, 306
- substituting multiple characters, 297-298
- summing cells based on interior color, 293-294
- validating email addresses, 292-293

fund/stock quotes (power programming techniques), 280-281

G

- GetObject function, referencing existing instances of Word objects, 410-411**
- GetOpenFilename, 175-176, 478**
- GetSaveAsFilename, 176**
- global names versus local names, 103-104**
- Go to Special versus looping (case study), 184**
- graphics**
 - buttons, adding graphics to, 497
 - custom icon images, 499-500
 - Microsoft Office icons, 498-499

- exporting charts as graphics, 330
- userforms, adding to, 169, 453-454

H

hard-coding rows/formulas, avoiding (cleaning up recorded code), 53-54

“Hello World” add-in (Office), 517-521

help

- Cell Masters website, 254
- Excel Tables: A Complete Guide for Creating, Using, and Automating Lists and Tables*, 71
- ExcelMatters.com website, 254
- help buttons case study, 151-153**
- JKP Application Development Services, 466
- Juanpg.com website, 268-270
- OpenXMLDeveloper.org website, 497
- RibbonX Visual Designer, 497
- userforms, adding to, 456
 - adding control tip text, 457
 - coloring active controls, 457-459
 - creating tab order, 457
 - showing accelerator keys, 456
- Wallentin blog, Dennis, 257
- XcelFiles website, 263

Help (VBA), 38-40

- defined constants, 40-43
- optional parameters, 40

hiding

- hidden log files, creating, 267-268
- hidden workbooks
 - as alternative to add-ins, 515-516
 - storing macros in, 515-516
 - storing userforms in, 515-516
- names, 111
- userforms, 160

highlighting

- cells, 334
 - selected cells via conditional formatting, 263-264
 - selected cells without conditional formatting, 264-265
- rows for the largest value, 353
- unique values in ranges, 352-353

historical stock/fund quotes (power programming techniques), 280-281**HTML (Hypertext Markup Language)**

- Office (MS) add-ins, 524
 - .CSS files, 525
 - HTML buttons, 524-525
 - HTML tags, 524

hyperlinks

- addresses, returning, 305-306
- macros, running, 507-508
- userforms, 449-450

I**icon sets, 334**

- ranges, adding to, 341-343
- subsets of ranges, creating for, 344-345

icons

- adding to buttons
 - custom icon images, 499-500
 - Microsoft Office icons, 498-499
- filtering by (AutoFilter), 181-182

If statements

- JavaScript interactivity in Office (MS) add-ins, 530
- nesting If statements (Select Case. End Select loops), 89-91

If.Else.If.End loops and flow control, 87-88**If.Then.Else loops and flow control, 86****If.Then.Else.End If loops and flow control, 87****If.Then.End If loops and flow control, 86-87****ignoring errors as a way of error handling/troubleshooting, 479****images**

- buttons, adding images to, 497
 - custom icon images, 499-500
 - Microsoft Office icons, 498-499
- exporting charts as images, 330
- userforms, adding to, 453-454

importing text files, 391

- delimited files, 395-397
- files with less than 1,048,576 rows, 391-397
- files with more than 1,048,576 rows, 398-402
- fixed-width files, 392-395
- loading large files to Data Model via Power Query, 401-402
- running files a row at a time, 398-400

initializing add-ins (Office) via JavaScript, 536**input boxes (userforms), 157-158****interactivity in Office (MS) add-ins, 521, 526, 535**

- arrays, 528-529
- assignment operators, 532-533
- For each.next statements, 532
- functions, 526-527
- if statements, 530
- initializing add-ins, 536
- logical operators, 532-533
- for loops, 529-530
- math functions, 534-535
- mathematical operators, 532-533
- reading/writing to add-ins, 536

Select.Case statements, 530-531
 strings, 528
 variables, 527-528
 writing to content pane, 535

Internet data

analyzing via Application.OnTime method, 381
 canceling all pending scheduled macros, 383
 canceling previously scheduled macros, 382
 Ready mode for scheduled procedures, 381-382
 scheduling macros to run every 2 minutes, 384-385
 scheduling macros to run x minutes in the future, 383
 scheduling verbal reminders, 383-384
 scheduling window of time for updates, 382
 content management via Excel, 387-389
 publishing to web pages, 385-386
 content management via Excel, 387-389
 custom web pages, 386-387
 FTP from Excel, 389-390
 retrieving, 375-380
 building multiple queries, 377-378
 examples of, 380
 finding results from retrieved data, 378-379
 interrupting macros, 117
 Intersect method, creating new ranges from overlapping ranges, 67
 IsEmpty function, checking for empty cells in ranges, 67

J

JavaScript and Office (MS) add-ins, 526, 535
 arrays, 528-529
 assignment operators, 532-533
 For each.next statements, 532
 functions, 526-527
 if statements, 530
 initializing, 536
 logical operators, 532-533
 for loops, 529-530
 math functions, 534-535
 mathematical operators, 532-533
 reading/writing to, 536
 Select.Case statements, 530-531
 strings, 528
 variables, 527-528
 writing to
 content pane, 535
 task pane, 535

Jiang, Wei, 273

JKP Application Development Services, 466

joining multiple ranges, 66-67

Jones, Kevin, 71, 258

Juanpg.com website, 268-270

K

Kaji, Masaru, 254

Kapor, Mitch, 93

keyboard shortcuts, running macros from, 504

Klann, Daniel, 275

L

labels (userforms), 163-165

largest value, highlighting rows for, 353

last row, finding (cleaning up recorded code), 52-53

late binding, referencing Word objects via, 408-409

layouts (pivot tables)
changing via Design tab (VBA), 248
report layout settings, 248-249

line breaks, JavaScript interactivity in Office (MS) add-ins, 527

lines (sparklines)
creating, 356-357
formatting, 361
 RGB colors, 364-365
 sparkline elements, 365-367
 theme colors, 361-364
scaling, 357-361

list boxes
multicolumn list boxes in userforms (case study), 459
MultiSelect property, 166-167
userforms, 165-167

lists
file lists in directories, 251-253
pivot tables, filtering OLAP pivot tables by lists of items, 271-273
unique lists (Advanced Filter), 186

local names versus global names, 103-104

locking code, 485-486

log files (hidden), creating, 267-268

logical operators (JavaScript) and Office (MS) add-ins, 532-533

loops, 73

Do loops, 78-80
 Until clause and, 81-82
 While clause and, 81-82
 While.Wend loops, 82

For Each. loops, 82-83
 exiting early, 77-78
 file directory case study, 84-85

flow control
 If.ElseIf.End loops, 87-88
 If.Then.Else loops, 86
 If.Then.Else.End If loops, 87
 If.Then.End If loops, 86-87
 Select Case.End Select loops, 88-91

Go to Special versus looping (case study), 184

If.ElseIf.End loops and flow control, 87-88

If.Then.Else loops and flow control, 86

If.Then.Else.End If loops and flow control, 87

If.Then.End If loops and flow control, 86-87

for loops, JavaScript interactivity in Office (MS) add-ins, 529-530

For.Next loops, 73-76
 exiting early, 77-78
 nested loops, 78
 running backwards, 77
 For statement variables, 76
 Step clause and, 76-77
 variations on, 76-77

nested loops, 78

- object variables
 - For Each. loops, 83-85
 - VBA loops (For Each. loops), 83-85
- replacing via AutoFilter, 177-179
- Select Case.End Select loops and flow control, 88
 - complex expressions in Case statements, 89
 - nesting If statements, 89-91
- VBA loops (For Each. loops), 82-83
- While.Wend loops, 82

Lotus 1-2-3

- Macro Recorder, 7-8
- R1C1 formulas, 93

M

Macro Recorder

- AutoSum while recording macros, 30-31
- case study
 - AutoSum while recording macros, 30-31
 - Quick Analysis while recording macros, 30-31
 - recording macros, 21-24
 - relative references when recording macros, 26-30
 - running macros on another day produces undesired results, 25-26
 - testing macros, 24-25
- Excel and, 8
- flaws in, 7-8
- Lotus 1-2-3, 7-8
- navigating data sets, 31
- Quick Analysis while recording macros, 30-31

Macro Security icon (Code group), 10-12**macros, 19**

- ActiveX Controls, attaching macros to, 506-507
- assigning to
 - Form Controls, 18-19
 - shapes, 18-19
 - text boxes, 18-19
- buttons, creating on
 - Quick Access Toolbar, 17
 - Ribbon, 16
- case studies
 - AutoSum, 30-31
 - Quick Analysis, 30-31
 - recording macros, 21-24
 - relative references, 26-30
 - running macros on another day produces undesired results, 25-26
 - testing macros, 24-25
- command buttons, attaching macros to, 504-505
- data analysis via Application.OnTime method
 - canceling all pending scheduled macros, 383
 - canceling previously scheduled macros, 382
 - scheduling macros to run every 2 minutes, 384-385
 - scheduling macros to run x minutes in the future, 383
 - scheduling verbal reminders, 383-384
- debugging
 - backing up/moving forward in code, 45-46
 - breakpoints, 45, 49
 - querying while stepping through code, 46-48
 - running to cursor, 46
 - stepping through code, 43-45

- editing
 - defined constants, 40-43
 - optional parameters, 40
 - Project Explorer, 20
 - properties that return objects, 43
 - Properties window, 21
 - settings, 20
- Form Controls, assigning macros to, 18-19
- hyperlinks, running macros from, 507-508
- interrupting, 117
- keyboard shortcuts, running macros from, 504
- pausing, 117
- Personal Macro Workbook (Personal.xlsm), 15
- Record Macro dialog, filling out, 14-16
- recording, 14-16
 - AutoSum, 30-31
 - case study, 21-24
 - navigating data sets, 31
 - Quick Analysis, 30-31
 - relative references, 26-30, 31
 - using different methods while recording, 31
- recording macros case study, 21-24
- relative references, 26-31
- restarting, 117
- Ribbon, customizing for running macros, 487-488
 - accessing Excel file structure, 496
 - adding controls to Ribbon, 490-491
 - creating groups, 489-490
 - creating tabs, 489-490
 - customui folder and file, 488-489
 - images on buttons, 497-500
 - .RELS files, 496-497
 - renaming Excel files, 497
 - troubleshooting (error handling), 500-503
- running, 16
 - case study, 25-26
 - creating macro buttons on Quick Access Toolbar, 17
 - creating macro buttons on Ribbon, 16
 - running macros on another day produces undesired results, 25-26
- running macros on another day produces undesired results, 25-26
- scheduling
 - canceling all pending scheduled macros, 383
 - canceling previously scheduled macros, 382
 - scheduling macros to run every 2 minutes, 384-385
 - scheduling macros to run x minutes in the future, 383
 - scheduling verbal reminders, 383-384
- security
 - Disable All Macros Except Digitally Signed Macros option, 14
 - Disable All Macros with Notification option, 13-14
 - Disable All Macros Without Notification option, 13
 - Enable All Macros option, 14
 - Macro Security icon (Code group), 10-12
 - using macros outside of trusted locations, 13-14
- shapes
 - assigning macros to, 18-19
 - attaching macros to, 505-506

- storing in hidden workbooks, 515-516
- testing, 24-25
- text boxes, assigning macros to, 18-19
- tips for using, 31
- trusted locations, using macros outside of, 13-14
- using different methods while recording, 31
- .xls files, 11
- .xlsb files, 11
- .xlsm files, 11
- .xlsx files, 10-11
- XML macros, 4
- Macro icon (Code group), 10**
- Macs (Apple) and VBA, 4**
- math functions (JavaScript), Office (MS) add-ins, 534-535**
- mathematical operators (JavaScript) and Office (MS) add-ins, 532-533**
- max values (duplicate), returning addresses of, 304-305**
- MDB (Multidimensional Databases), 423-424**
 - records
 - adding to MDB, 427-428
 - deleting via ADO, 433
 - retrieving from MDB, 429-430
 - summarizing via ADO, 433-434
 - updating existing records, 431-432
 - shared access databases, creating (case study), 425-426
- memory, reading text files into, 254-255**
- message boxes (userforms), 158**
- methods (VBA), 34-37**
 - Application.OnTime method and data analysis, 381
 - canceling all pending scheduled macros, 383
 - canceling previously scheduled macros, 382
 - Ready mode for scheduled procedures, 381-382
 - scheduling macros to run every 2 minutes, 384-385
 - scheduling macros to run x minutes in the future, 383
 - scheduling verbal reminders, 383-384
 - scheduling window of time for updates, 382
- data visualization, 334-335
- Format method, micromanaging formatting changes in charts, 324-325
- Intersect method, creating new ranges from overlapping ranges, 67
- new methods, learning, 542
- parameters, 35-37
- Save As method, converting workbooks to add-ins, 511
- SetElement method, emulating Plus icon changes, 319-323
- SpecialCells method, selecting specific cells in ranges, 68-70
- Union method, joining multiple ranges, 66-67
- Microsoft Office icons, adding to buttons, 498-499**
- Miles, Tommy, 255-256**
- military time, entering in cells (case study), 122**
- mixed references and R1C1 formulas, 98-99**
- mixed text, retrieving numbers from, 298-299**
- Moala, Ivan F., 263-264, 277-279**
- modeless userforms, 449**
- Modify group (Developer tab), 10**
- monthly accounting reports case study, 3-4**

months, grouping daily dates to, 221-222

moving

forward/backing up in code (debugging), 45-46

pivot tables, 216-217

multidimensional arrays, declaring, 132-133

MultiPage control, combining userforms, 171-173

multiple actions in recorded code, 54-55

multiple characters, substituting via UDF, 297-298

multiple items, selecting via AutoFilter, 180

MultiSelect property, userform list boxes, 166-167

N

names, 103

adding comments to, 106

arrays, 109-110

checking existence of, 111-112

creating, 104-105

deleting, 105-106

formulas, 106-107

global names

creating, 104-105

local names versus, 103-104

hiding, 111

local names

creating, 105

global names versus, 103-104

Name Manager

adding comments to names, 106

global names versus local names, 103-104

local names versus global names, 103-104

named ranges, 60

numbers, 108-109

renaming Excel files, 497

reserved names, 110-111

strings, 107-108

tables, 109

types of names, 106-111

Vlookup, named ranges for (case study), 112-113

workbooks

setting names and file paths in cells, 287

setting names in cells, 286

Napa Office 365 development tools and Office (MS) add-ins, 536-537

navigating

data sets in Macro Recorder, 31

Object Browser, 50

nested If statements (Select Case.End Select loops), 89-91

nested loops, 78

New keyword, referencing Word objects, 409

noncontiguous cells, selecting, 265-267

noncontiguous ranges, returning via Areas collection, 70-71

nonzero-length cells, finding in ranges, 296-297

NumberFormat property, 353-354

numbers

names, 108-109

retrieving from mixed text, 298-299

week numbers, converting to dates, 299

numeric characters, sorting, 302-303

O

Object Browser

- navigating, 50
- opening, 50
- Word constant values, retrieving, 412-413

object fills, changing in charts, 325-327**object variables**

- For Each. loops, 83-85
- VBA loops (For Each. loops), 83-85

objects (VBA), 34-37

- class modules, inserting into objects, 139-140
- collections, 35
- custom objects
 - creating, 143-145
 - using, 145
- hierarchy of, 59
- new objects, learning, 542
- properties, 36-37
- Range object, 59-60
- returning objects via properties, 43
- watches and, 49-50

objects (Word), 405-406, 413

- bookmarks, 419-420
- constant values, retrieving via late binding, 411-413
- Document object, 413-415
- new instances, creating via CreateObject function, 409
- Range object, 416-419
- referencing
 - early binding, 406-408
 - early binding using New keyword, 409
 - late binding, 408-409
 - referencing existing instances via GetObject function, 410-411
- Selection object, 415-416

Office (MS)

- add-ins, 517
 - buttons (HTML), 524-525
 - creating, 517-521
 - .CSS files, 525
 - “Hello World” add-in, 517-521
 - HTML in, 524-525
 - interactivity in, 521-524
 - JavaScript interactivity in, 526-536
 - Napa Office 365 development tools, 536-537
 - tags (HTML), 524
 - writing to content pane, 535
 - writing to task pane, 535
 - XML in, 525-526
- icons, adding to buttons, 498-499

Offset property, referencing ranges, 63-65**OLAP pivot tables, filtering by lists of items, 271-273****Oliver, Nathan P.251, 280****On Error GoTo syntax (error handling/ troubleshooting), 477-478****online resources**

- Cell Masters website, 254
- code files, 5
- ExcelMatters.com website, 254
- exp.com, 271
- JKP Application Development Services, 466
- Juanpg.com website, 268-270
- OpenXMLDeveloper.org website, 497

RibbonX Visual Designer, 497
 Wallentin blog, Dennis, 257
 XcelFiles website, 263

open status (workbooks), checking, 287

OpenXMLDeveloper.org website, 497

option buttons in userforms, 167-169

overlapping ranges, creating new ranges from, 67

P

parameters (VBA), 35-37

- event parameters, 116
- optional parameters, 40

parsing text files, 254-255

passing arrays, 137-138

passwords

- cracking (case study), 484-485
- protected password boxes, creating, 275-277

pausing macros, 117

percentages, changing calculations to show (pivot tables), 222-224

performance, speeding up code via arrays, 135-136

permanent date and time, retrieving, 291

Personal Macro Workbook (Personal.xlsm), 15

Pieterse, Jan Karel, 466

pivot tables, 2-3, 211, 212, 219-220

- calculated data fields, 247
- calculated items, 247
- case studies, 249-250
- changing, 216-217
- configuring, 213-214
- converting to values, 217-219
- creating, 213-214
- daily dates, grouping to months, quarters, years, 221-222

- data area, adding fields to, 214-216

Data Model, 242

- adding model fields to pivot tables, 244
- adding numeric fields to value area, 244-245
- adding tables to Data Model, 242-243
- building pivot tables, 243
- creating, 245-247
- creating relationships between tables, 243
- defining pivot caches, 243

data visualization case study, 249-250

development of, 211-212

drilling down, 270-271

Excel's changes to, 541

fields

- adding to data area, 214-216
- multiple value fields, 220-221

filtering

- case study, 233-235
- conceptual filters, 229-230
- configuring filtering timelines, 239-242
- configuring slicers for filtering, 235-239
- manually filtering multiple items in pivot fields, 228-229
- record sets via ShowDetail, 248
- search filter, 233
- types of filters, 230

layouts

- changing via Design tab (VBA), 248
- report layout settings, 248-249

moving, 216-217

OLAP pivot tables, filtering by lists of items, 271-273

- percentages, changing calculations to show, 222-224
- pivot caches, defining, 212-213
- record sets, filtering via ShowDetail, 248
- reports
 - layout settings, 248-249
 - replicating for every product
- Show Report Filter Pages, 225-228
- ShowDetail, filtering record sets via, 248
- sizing tables to convert to values, 217-219
- sort order, controlling via AutoSort, 225
- subtotals, suppressing for multiple row fields, 249-250
- values area, deleting empty cells from, 225
- versions of, 211-212

Plus icon, emulating changes via SetElement, 319-323

Pope, Andy, 497

power programming (Excel)

- cells
 - creating hidden log files, 267-268
 - highlighting selected cells using conditional formatting, 263-265
 - placing charts in comments, 261-262
 - resizing comments, 260-261
 - selecting noncontiguous cells, 265-267
- charts, placing in cell comments, 261-262
- files
 - creating hidden log files, 267-268
 - deleting .CSV files, 254
 - importing .CSV files, 254

- listing in directories, 251-253
- parsing text files, 254-255
- reading text files into memory, 254-255
- workbooks
 - combining, 256-257
 - separating worksheets into workbooks, 255-256
- worksheets
 - copying data to separate worksheets, 257-258
 - copying data to separate worksheets without filters, 258-259
 - filtering data to separate worksheets, 257-258
 - separating into workbooks, 255-256
- .XML files, exporting data to, 259-260

power programming (VBA)

- cells
 - progress indicators, 274-275
 - selecting via SpecialCells, 279
- custom sort orders, 273-274
- Excel State class modules, creating, 268-270
- fund/stock quotes, 280-281
- pivot tables
 - drilling down, 270-271
 - filtering OLAP pivot tables by lists of items, 271-273
- protected password boxes, creating, 275-277
- stock/fund quotes, 280-281
- tables, resetting formatting, 279-280
- text, changing case, 277-278
- workbooks, adding code via VBA extensibility, 281-282

Power Query

loading large text files to Data Model, 401-402

Web data (Internet), retrieving, 375-378

progress indicators (cells), creating, 274-275

Project Explorer (VB Editor), 20

properties (VBA), 36-37

Columns property, specifying ranges, 66

CurrentRegion property, selecting data ranges, 68

custom properties, creating via UDT, 153-156

data visualization, 334-335

Excel8CompatibilityMode property, 543-544

MultiSelect property and userform list boxes, 166-167

NumberFormat property, 353-354

Offset property, referencing ranges, 63-65

Resize property, sizing ranges, 65-66

returning objects, 43

Rows property, specifying ranges, 66

Version property, 543

Properties window (VB Editor), 20

protected password boxes, creating, 275-277

publishing data to web pages, 385-386

content management via Excel, 387-389

custom web pages, 386-387

FTP from Excel, 389-390

Q

quarters, grouping daily rates to via pivot tables, 221-222

queries (web) and data retrieval, 375-377, 379-380

building multiple queries, 377-378

examples of, 380

finding results from retrieved data, 378-379

QueryClose events, userform windows, 174-175

querying while stepping through code (debugging), 46

hovering the cursor, 47-48

Immediate window, 46-47

watches, 48

Quick Access Toolbar, creating macro buttons on, 17

Quick Analysis tool

Excel's changes to, 541

recording macros, 30-31

quotes (stock/fund), power programming techniques, 280-281

R

R1C1 formulas

A1 versus, 93-97

absolute references and, 98

accessing, 94-95

array formulas and, 101-102

autofilling data, 96

case study, 96-97

cleaning up recorded code, 54

columns

number/letter associations, 101

references, 99

- mixed references and, 98-99
- relative references and, 97-98
- replacing multiple A1 formulas with one R1C1 formula, 99-101
- row references, 99

random (static), 306**Range object (Word), 416-419****ranges, 59**

- case study, selecting specific cells in ranges, 68-70
- cells
 - checking for empty cells, 67
 - finding the first nonzero-length cell in a range, 296-297
 - ranges versus cells when cleaning up code, 52
 - selecting specific cells, 68-70
- color scales, adding to ranges, 339-340
- criteria ranges (Advanced Filter), 186, 192-194
 - case study, 194-197
 - complex criteria, 194-195
 - formula-based conditions, 196-201
 - joining multiple ranges via Logical AND, 194
 - joining multiple ranges via Logical OR, 193-194
- data bars
 - adding to ranges, 335-339
 - multiple colors of data bars in ranges, 345-347
- dynamic date ranges, selecting via AutoFilter, 182-183
- highlighting unique values in ranges, 352-353
- icon sets
 - adding to ranges, 341-343
 - creating for subsets of ranges, 341-343

- joining multiple ranges, 66-67
- named ranges, 60, 112-113
- noncontiguous ranges, returning via Areas collection, 70-71
- overlapping ranges, creating new ranges from, 67

Range object, 59-60**referencing**

- Offset property, 63-65
- in other worksheets, 61
- relative to other ranges, 61-62
- shortcuts, 60
- tables, 71
- removing duplicates from, 295-296
- selecting
 - data ranges, 68
 - ranges via cells, 62-63
- sizing via Resize property, 65-66
- specifying
 - columns and, 66
 - rows and, 66
 - syntax for, 60

reading/writing to add-ins (Office) via JavaScript, 536**Record Macro dialog, filling out, 14-16****Record Macro icon (Code group), 10****record sets (ADO), 426****recording**

- cleaning up recorded code, 51
 - avoiding hard-coding rows, 53-54
 - case study, 55-57
 - cells versus ranges, 52
 - copying/pasting in statements, 54
 - With.End With blocks for multiple actions, 54-55
 - finding the last row, 52-53
 - multiple actions in recorded code, 54-55

- R1C1 formulas, 54
- ranges versus cells, 52
- selecting things, 51
- macros, 14-16
 - AutoSum, 30-31
 - case study, 21-24
 - navigating data sets, 31
 - Quick Analysis, 30-31
 - relative references, 26-30, 31
 - using different methods while recording, 31
- RefEdit controls (userforms), 444**
- referencing**
 - charts (specific), 315-316
 - ranges
 - Offset property, 63-65
 - in other worksheets, 61
 - relative to other ranges, 61-62
 - shortcuts, 60
 - tables, 71
 - Word objects
 - early binding, 406-408
 - early binding using New keyword, 409
 - late binding, 408-409
- relative references**
 - R1C1 formulas and, 97-98
 - recording macros, 26-31
- reminders (verbal), scheduling for macro operation, 383-384**
- removing**
 - add-ins, 514-515
 - duplicates from ranges, 295-296
- renaming Excel files, 497**
- reports**
 - accounting reports case study, 3-4
 - layout settings, 248-249
 - replicating for every product (pivot tables), 225-228
- reserved names, 110-111**
- resetting table formatting, 279-280**
- Resize property, sizing ranges, 65-66**
- resources (online)**
 - Cell Masters website, 254
 - code files, 5
 - ExcelMatters.com website, 254
 - exp.com, 271
 - JKP Application Development Services, 466
 - Juanpg.com website, 268-270
 - OpenXMLDeveloper.org website, 497
 - RibbonX Visual Designer, 497
 - Wallentin blog, Dennis, 257
 - XcelFiles website, 263
- restarting macros, 117**
- retrieving**
 - data from arrays, 134-135
 - date and time
 - permanent date and time, 291
 - retrieving from last save, 291
 - Internet data, 375-380
 - building multiple queries, 377-378
 - examples of, 380
 - finding results from retrieved data, 378-379
 - numbers from mixed text, 298-299
 - user ID, 289-290
- reversing cell contents, 304**
- Ribbon**
 - creating macro buttons on, 16
 - customizing for running macros, 487-488
 - accessing Excel file structure, 491
 - adding controls to Ribbon, 490-491
 - creating groups, 489-490

- creating tabs, 489-490
- customui folder and file, 488-489
- images on buttons, 497-500
- .RELS files, 496-497
- renaming Excel files, 497
- troubleshooting (error handling), 500-503

Excel's changes to, 539

RibbonX Visual Designer, 497

rows

- hard-coding rows, avoiding (cleaning up recorded code), 53-54
- highlighting for the largest value, 353
- last row, finding (cleaning up recorded code), 52-53
- RIC1 formulas, row references, 99
- ranges, specifying via rows, 66

Ruiz, Juan Pablo Gonzalez, 268-270

running

- to cursor (debugging code), 46
- macros, 16
 - case study, 25-26
 - creating macro buttons on Quick Access Toolbar, 17
 - creating macro buttons on Ribbon, 16
 - on another day produces undesired results, 25-26

running timers, creating via API declarations, 471

Runtime Error 9: Subscript Out of Range, 482-483

Runtime Error 1004: Method Range of Object Global Failed, 483-484

S

Save As method, converting workbooks to add-ins, 511

saving

- date and time from last save, retrieving, 291
- files in .xlsm file format, 11

scaling sparklines, 357-361

scheduling macros

- canceling
 - all pending scheduled macros, 383
 - previously scheduled macros, 382
- to run
 - every 2 minutes, 384-385
 - x minutes in the future, 383
- verbal reminders, 383-384

scraping websites (data retrieval), 375-380

- examples of, 380
- multiple queries, building, 377-378
- results, finding from retrieved data, 378-379

Scrollbar controls (userforms), 446

SDI (Single Document Interface), Excel's changes to, 540

search box (AutoFilter), selecting via, 180-181

searching for strings within text, 303-304

security

- add-ins, 513-514
- error handling (troubleshooting)
 - code security and debugging, 484
 - locking code, 485-486
 - password cracking (case study), 484-485
- locking code, 485-486

- Macro Security icon (Code group), 10-12
- macros
 - Disable All Macros Except Digitally Signed Macros option, 14
 - Disable All Macros with Notification option, 13-14
 - Disable All Macros Without Notification option, 13
 - Enable All Macros option, 14
 - Macro Security icon (Code group), 10-12
 - using macros outside of trusted locations, 13-14
- password cracking (case study), 484-485
- protected password boxes, creating, 275-277
- Trust Center, accessing, 10
- trusted locations
 - adding a trusted location, 12-13
 - using macros outside of trusted locations, 13-14
- Select.Case.End Select loops and flow control, 88**
 - complex expressions in Case statements, 89
 - nesting If statements, 89-91
- Select.Case statements**
 - JavaScript interactivity in Office (MS) add-ins, 530-531
 - worksheets, 307
- selecting**
 - cells, 263
 - creating hidden log files, 267-268
 - highlighting selected cells using conditional formatting, 263-264
 - highlighting selected cells without using conditional formatting, 264-265
 - noncontiguous cells, 265-267
 - via SpecialCells, 279
 - dynamic date ranges via AutoFilter, 182-183
 - multiple items via AutoFilter, 180
 - ranges via cells, 62-63
 - via search box (AutoFilter), 180-181
 - visible cells via AutoFilter, 183-184
- Selection object (Word), 415-416**
- semicolons (;), JavaScript interactivity in Office (MS) add-ins, 527**
- SetElement method, emulating Plus icon changes, 319-323**
- shapes**
 - assigning macros to, 18-19
 - attaching macros to, 505-506
- sharing UDF, 286**
- shortcuts (keyboard), running macros from, 504**
- Show Report Filter Pages (pivot tables), 225-228**
- ShowDetail, filtering record sets in pivot tables, 248**
- signatures (digital), Disable All Macros Except Digitally Signed Macros option (macro security), 14**
- sizing**
 - cell comments, 260-261
 - pivot tables to convert to values, 217-219
 - ranges via Resize property, 65-66
 - userforms “on the fly,” 452
- slicers**
 - configuring for pivot table filtering, 235-239
 - Excel's changes to, 541

SmartArt, Excel's changes to, 542**Smith, Chris "Smitty," 267****sorting**

- alpha characters, 302-303
- concatenating data (sorting and), 300-302
- custom sort orders, creating, 273-274
- numeric characters, 302-303
- pivot table sort orders, controlling via AutoSort, 225

sounds, playing via API declarations, 472**spaces, JavaScript interactivity in Office (MS) add-ins, 527****sparklines, 355**

- columns, 355
 - creating, 356-357
 - formatting, 361-369
 - RGB colors, 364-365
 - scaling, 357-361
 - theme colors, 361-364
- dashboards, creating, 369-373

lines, 355

- creating, 356-357
- formatting, 361-369
- RGB colors, 364-365
- scaling, 357-361
- theme colors, 361-364
- observations about, 369-370
- usage observations, 369-370
- win/loss charts, 355
 - binary event tracking, 368-369
 - creating, 356-357
 - formatting, 361-369
 - RGB colors, 364-365
 - scaling, 357-361
 - theme colors, 361-364

SpecialCells method

- selecting cells via, 279
- selecting specific cells in ranges, 68-70

speeding up code via arrays, 135-136**spin buttons in userforms, 170-171****SQL server and Access, 437-438****standard modules, creating in collections, 146-147****statements, copying/pasting within (cleaning up recorded code), 54****static random, 306****Step clause, For.Next loops, 76-77****stepping through code (debugging), 43-45****stock/fund quotes (power programming techniques), 280-281****storing**

- macros in hidden workbooks, 515-516
- userforms in hidden workbooks, 515-516

strings

- delimited strings, extracting a single element from, 300
- JavaScript strings in Office (MS) add-ins, 528
- names, 107-108
- searching for a string within text, 303-304

substituting multiple characters via UDF, 297-298**Sullivan, Jerry, 271****summarizing MDB records, 433-434****summing cells based on interior color, 293-294**

T

tables

Access tables

- adding tables “on the fly” via ADO, 436
- verifying existence via ADO, 434-435

formatting, resetting, 279-280

names, 109

- pivot tables. *See* individual entry
- referencing, 71

tabs, customizing Ribbon for running macros, 489-490

TabStrip controls (userforms), 442-443

tags (HTML) and Office (MS) add-ins, 524

task pane (Office add-ins), writing to, 535

testing macros, case study, 24-25

text

- case, changing, 277-278
- cells, formatting text in, 351
- mixed text, retrieving numbers from, 298-299
- searching for a string within text, 303-304
- tip text, adding to userform controls, 457

text boxes

- macros, assigning to text boxes, 18-19
- userforms, 163-165

text files, 391

- importing, 391
 - delimited files, 395-397
 - files with less than 1,048,576 rows, 391-397
 - files with more than 1,048,576 rows, 398-402

fixed-width files, 392-395

loading large files to Data Model via Power Query, 401-402

running files a row at a time, 398-400

parsing, 254-255

reading into memory, 254-255

writing, 402

timelines (pivot table filtering), configuring, 239-242

timers (running), creating via API declarations, 471

tip text, adding to userform controls, 457

titles (chart), specifying, 316

ToggleButton controls (userforms), 444

top/bottom rules, 334, 348-349

transparent userforms, 460-461

troubleshooting (error handling), 473

blanks/errors formatting, 351-352

case study, 480

clients, training, 481-482

code security

debugging and, 484

locking code, 485-486

password cracking (case study), 484-485

development stage errors versus errors months later, 482

encountering errors on purpose, 481

On Error GoTo syntax, 477-478

Excel warnings, suppressing, 481

generic error handlers, 478-479

Help (VBA), 38

ignoring errors, 479

Ribbon, customizing for running macros, 500-503

Runtime Error 9: Subscript Out of Range, 482-483

Runtime Error 1004: Method Range of Object Global Failed, 483-484

userforms

adding controls to existing forms, 162-163

debug errors, 475-477

VBA, 473-475

versioning errors, 486

Trust Center, accessing, 10

trusted locations

adding a trusted location, 12-13

macros, using outside of trusted locations, 13-14

Tufte, Prof. Edward, 355

U

UDF (User-Defined Functions), 283, 286

case study, 284-285

cells

finding the first nonzero-length cell in a range, 296-297

returning column letters from cell addresses, 306

reversing cell contents, 304

setting workbook names and file paths in cells, 287

setting workbook names in cells, 286

summing cells based on interior color, 293-294

column letters, returning from cell addresses, 306

concatenating data (sorting and), 300-302

creating, 283-285

date and time

retrieving from last save, 291

retrieving permanent date and time, 291

duplicate max values, returning the addresses of, 304-305

email addresses, validating, 292-293

hyperlink addresses, returning, 305-306

multiple characters, substituting, 297-298

numbers

converting week numbers to dates, 299

retrieving from mixed text, 298-299

ranges

finding the first nonzero-length cell in a range, 296-297

removing duplicates from, 295-296

Select.Case statements in worksheets, 307

sharing, 286

sorting

alpha characters, 302-303

concatenating data (sorting and), 300-302

numeric characters, 302-303

static random, 306

strings

extracting a single element from a delimited string, 300

searching for a string within text, 303-304

unique values, counting, 294-295

user ID, retrieving, 289-290

workbooks

checking open status, 287

counting number of workbooks in a directory, 288-289

setting names and file paths in cells, 287

setting names in cells, 286

- worksheets
 - checking existence of, 287-288
 - Select.Case statements, 307
- UDT (User-Defined Types), creating custom properties, 153-156**
- Union method, joining multiple ranges, 66-67**
- unique cells, formatting, 349-350**
- unique lists (Advanced Filter), 186**
 - extracting values via user interface, 186
 - changing list ranges to single column format, 186-187
 - copying customer headings, 187
 - extracting values via VBA code, 187-191
 - unique combinations of two or more fields, 191-192
- unique values**
 - counting, 294-295
 - highlighting in ranges, 352-353
- Until clauses and Do loops, 81-82**
- updating existing records in MDB, 431-432**
- Urtis, Tom, 260-261, 265, 270, 274**
- Use Relative References icon (Code group), 10**
- user ID, retrieving, 289-290**
- userforms, 157**
 - calling, 159-160
 - case study, 162-163, 459
 - closing, disabling X button via API declarations, 470-471
 - collections and controls, 447-449
 - combining, 171-173
 - combo boxes, 165-167
 - command buttons, 163-165
 - controls, 440
 - adding, 453
 - adding at runtime, 450-456
 - adding controls to existing forms, 162-163
 - adding “on the fly,” 452
 - adding tip text, 457
 - case study, 162-163
 - Checkbox controls, 440-441
 - collections and, 447-449
 - coloring active controls, 457-459
 - designing via toolbar, 439-440
 - programming, 162
 - RefEdit controls, 444
 - Scrollbar controls, 446
 - TabStrip controls, 442-443
 - ToggleButton controls, 444
 - creating, 158-159
 - debug errors, troubleshooting, 475-477
 - events, 160-161
 - field entry, verifying, 174
 - filenames, 175-176
 - frame control events, 167-169
 - graphics, 169
 - help, adding, 456
 - adding control tip text, 457
 - coloring active controls, 457-459
 - creating tab order, 457
 - showing accelerator keys, 456
 - hiding, 160
 - hyperlinks, 449-450
 - illegal window closing, 174-175
 - images, adding “on the fly,” 453-454
 - input boxes, 157-158
 - labels, 163-165
 - list boxes, 165, 459

- message boxes, 158
- modeless userforms, 449
- multicolumn list boxes (case study), 459
- MultiSelect property and list boxes, 166-167
- option buttons, 167-169
- programming, 160
- QueryClose events, 174-175
- scrollbar, using as a slider to select values, 446
- sizing “on the fly,” 452
- spin buttons, 170-171
- storing in hidden workbooks, 515-516
- text boxes, 163-165
- toolbar and control design, 439-440
- transparent userforms, 460-461
- troubleshooting, 162-163, 475-477
- X button for closing userforms, disabling via API declarations, 470-471

V

validating email addresses, 292-293

value-based cell formatting, 350-351

variables

- JavaScript variables in Office (MS) add-ins, 527-528

- object variables

- For Each. loops, 83-85

- VBA loops (For Each. loops), 83-85

variant variables in arrays, 133

VB Editor, 19

- cleaning up code, 51
 - avoiding hard-coding rows, 53-54
 - case study, 55-57
 - cells versus ranges, 52

- copying/pasting in statements, 54

- With.End With blocks for multiple actions, 54-55

- finding the last row, 52-53

- multiple actions in recorded code, 54-55

- R1C1 formulas, 54

- ranges versus cells, 52

- selecting things, 51

- converting workbooks to add-ins, 511-512

- debugging code

- backing up/moving forward in code, 45-46

- breakpoints, 45, 49

- querying while stepping through code, 46-48

- running to cursor, 46

- stepping through code, 43-45

- defined constants, 40-43

- Object Browser

- navigating, 50

- opening, 50

- opening, 10

- parameters (optional), 40

- Project Explorer, 20

- properties, 43

- Properties window, 21

- returning objects, 43

- settings, 20

- UDF, creating, 284-285

- userforms, creating, 158-159

VBA (Visual Basic for Applications), 1, 7, 9

- add-ins, 509

- case study, 515-516

- characteristics of, 509-510

- client installations, 512-514

- closing, 514

- converting workbooks to, 510-512
- hidden workbooks as alternative to add-ins, 515-516
- Office (MS) add-ins. *See* individual entry
- removing, 514-515
- security, 513-514
- BASIC versus, 8, 33-34
- charts
 - applying color, 317-318
 - changing object fills, 325-327
 - emulating Plus icon changes via SetElement, 319-323
 - filtering, 318
 - formatting, 312-315
 - formatting line settings, 327
 - good/bad of the VBA creation process, 309-310
 - micromanaging formatting changes, 324-325
 - specific chart references, 315-316
 - specifying titles, 316
 - styles of, 312-315
 - types of, 313-315
- collections, 35-37, 139, 145
 - creating, 146
 - creating in class modules, 148-149
 - creating in standard modules, 146-147
 - returning noncontiguous ranges, 70-71
 - userform controls and, 447-449
- Data Model, 242
 - adding model fields to pivot tables, 244
 - adding numeric fields to value area, 244-245
 - adding pivot tables to Data Model, 242-243
 - building pivot tables, 243
 - creating, 245-247
 - creating relationships between pivot tables, 243
 - defining pivot caches, 243
- data visualization
 - methods, 334-335
 - properties, 334-335
- error handling (troubleshooting), 473-475
- example of, 33
- future of, 4
- Help, 38-40
 - defined constants, 40-43
 - optional parameters, 40
- Macs and, 4
- methods, 34-37
 - Application.OnTime method and data analysis, 381-385
 - data visualization, 334-335
 - Format method, 324-325
 - Intersect method, 67
 - learning new methods, 542
 - parameters, 35-37
 - Save As method, 511
 - SetElement method, 319-323
 - SpecialCells method, 68-70
 - Union method, 66-67
- objects, 34-37
 - collections, 35-37
 - custom objects, creating, 143-145
 - custom objects, using, 145
 - hierarchy of, 59
 - inserting class modules into objects, 139-140
 - learning new objects, 542
 - properties, 36-37

- Range object, 59-60
 - returning objects via properties, 43
 - watches and, 49-50
- parameters, 35-37
 - event parameters, 116
 - optional parameters, 40
- pivot tables, 211-212, 219-220
 - adding fields to data area, 214-216
 - calculated data fields, 247
 - calculated items, 247
 - case study, 249-250
 - changing, 216-217
 - changing calculations to show percentages, 222-224
 - conceptual filters, 229-230
 - configuring, 213-214
 - configuring filtering timelines, 239-242
 - configuring slicers for filtering, 235-239
 - controlling sort order via AutoSort, 225
 - converting to values, 217-219
 - creating, 213-214
 - Data Model and, 242-247
 - data visualization case study, 249-250
 - defining pivot caches, 212-213
 - deleting empty cells from values area, 225
 - development of, 211-212
 - drilling down, 270-271
 - filtering case study, 233-235
 - filtering record sets via ShowDetail, 248
 - grouping daily dates to months, quarters, years, 221-222
 - layout changes via Design tab (VBA), 248
 - manually filtering multiple items
 - in pivot fields, 228-229
 - moving, 216-217
 - multiple value fields, 220-221
 - replicating reports for every product, 225-228
 - report layout settings, 248-249
 - search filter, 233
 - Show Report Filter Pages, 225-228
 - sizing tables to convert to values, 217-219
 - suppressing subtotals for multiple row fields, 249-250
 - types of filters, 230
 - versions of, 211-212
- power programming
 - adding code to workbooks via VBA extensibility, 281-282
 - cell progress indicators, 274-275
 - changing case (text), 277-278
 - creating Excel State class modules, 267
 - custom sort orders, 273-274
 - drilling down pivot tables, 270-271
 - filtering OLAP pivot tables by lists of items, 271-273
 - protected password boxes, 275-277
 - resetting table formatting, 279-280
 - selecting cells via SpecialCells, 279
 - stock/fund quotes, 280-281
- properties, 36-37
 - Columns property, 66
 - CurrentRegion property, 68
 - custom properties, creating via UDT, 153-156
 - data visualization, 334-335
 - Excel8CompatibilityMode property, 543-544

- MultiSelect property and userform list boxes, 166-167
- NumberFormat property, 353-354
- Offset property, 63-65
- Resize property, 65-66
- returning objects, 43
- Rows property, 66
- Version property, 543
- syntax of, 34-37
- workbooks, adding code via VBA extensibility, 281-282
- VBA loops (For Each. loops), 82-83**
- VBEVB Editor**
 - new methods, learning, 542
 - new objects, learning, 542
- verbal reminders, scheduling for macro operation, 383-384**
- verifying**
 - Access field existence via ADO, 435-436
 - Access table existence via ADO, 434-435
 - field entry in userforms, 174
- Version property, Excel compatibility issues, 543**
- versioning errors, 486**
- visible cells, selecting via AutoFilter, 183-184**
- Visual Basic icon (Code group), 10**
- visualizing data, 333**
 - above/below average rules, 334, 348
 - cells
 - blanks/errors formatting, 351-352
 - date formatting, 351
 - duplicate cell formatting, 349-350
 - formulas to determine cells to format, 352-353
 - text formatting, 351
 - unique cell formatting, 349-350
 - value-based formatting, 350-351
 - color scales, 339-340
 - conditional formatting, 334
 - data bars, 334
 - adding to ranges, 335-339
 - multiple colors of data bars in ranges, 345-347
 - duplicate value rules, 334
 - highlight cell rules, 334
 - icon sets, 334
 - adding to ranges, 341-343
 - creating for subsets of ranges, 344-345
 - methods, 334-335
 - NumberFormat property, 353-354
 - pivot tables case study, 249-250
 - properties, 334-335
 - ranges
 - data bars, 335, 345-347
 - highlighting unique values in, 352-353
 - icon sets, 341, 344-345
 - rows, highlighting for the largest value, 353
 - top/bottom rules, 334, 348-349
- Vlookup, named ranges for (case study), 112-113**

W

Wallentin, Dennis, 257

warnings (Excel), suppressing, 481

watches

- breakpoints in, 49
- objects and, 49-50
- querying while stepping through code, 48

Watches window, retrieving Word constant values, 411-412**Web data (Internet)**

- analyzing via Application.OnTime method, 381
 - canceling all pending scheduled macros, 383
 - canceling previously scheduled macros, 382
- Ready mode for scheduled procedures, 381-382
- scheduling macros to run every 2 minutes, 384-385
- scheduling macros to run x minutes in the future, 383
- scheduling verbal reminders, 383-384
- scheduling window of time for updates, 382
- content management via Excel, 387-389
- publishing to web pages, 385-386
 - content management via Excel, 387-389
 - custom web pages, 386-387
 - FTP from Excel, 389-390
- retrieving, 375-380
 - building multiple queries, 377-378
 - examples of, 380
 - finding results from retrieved data, 378-379

web pages

- content management via Excel, 387-389
- custom web pages, 386-387
- publishing data to, 385-386
 - content management via Excel, 387-389
 - custom web pages, 386-387
 - FTP from Excel, 389-390

web queries and data retrieval, 375-380

- examples of, 380
- multiple queries, building, 377-378
- results, finding from retrieved data, 378-379

web resources

- Cell Masters website, 254
- code files, 5
- ExcelMatters.com website, 254
- exp.com, 271
- JKP Application Development Services, 466
- Juanpg.com website, 268-270
- OpenXMLDeveloper.org website, 497
- RibbonX Visual Designer, 497
- Wallentin blog, Dennis, 257
- XcelFiles website, 263
 - building multiple queries, 377-378
 - examples of, 380
 - finding results from retrieved data, 378-379

websites, scraping (data retrieval), 375-380**week numbers, converting to dates, 299****While clauses and Do loops, 81-82****While.Wend loops, 82****windows (userform), illegally closing, 174-175****win/loss charts (sparklines), 355**

- creating, 356-357
- formatting, 361
 - binary event tracking, 368-369
 - RGB colors, 364-365
 - theme colors, 361-364
- scaling, 357-361

With.End With blocks, cleaning up recorded code, 54-55

Word, automating, 405-406

- bookmarks, 419-420
- Document object, 413-415
- form fields, 420-422
- objects, 413
 - creating new instances via
 - CreateObject function, 409
 - late binding using constant values, 411-413
 - referencing existing instances via
 - GetObject function, 410-411
 - referencing via early binding, 406-408
 - referencing via early binding using
 - New keyword, 409
 - referencing via late binding, 408-409
- Range object, 416-419
- Selection object, 415-416

workbooks

- adding code via VBA extensibility, 281-282
- add-ins
 - case study, 515-516
 - converting to add-ins, 510-511
 - converting to add-ins via Save As method, 511
 - converting to add-ins via VB Editor, 511-512
 - hidden workbooks as alternative to add-ins, 515-516
- combining, 256-257
- directories, counting number of workbooks in, 288-289
- events
 - workbook events, 117
 - workbook-level sheet events, 119

- hidden workbooks
 - as alternative to add-ins, 515-516
 - storing macros in, 515-516
 - storing userforms in, 515-516
 - names, setting in cells, 286-287
 - open status, checking, 287
 - Personal Macro Workbook (Personal.xlsm), 15
 - trusted locations
 - adding a trusted location, 12-13
 - using macros outside of trusted locations, 13-14
 - workbook-level sheet events, 119
 - worksheets
 - checking existence of, 287-288
 - separating into workbooks, 255-256
 - .xls files, 11
 - .xlsb files, 11
 - .xlsm files
 - macros, 11
 - saving files in, 11
 - .xlsx files, 10-11
- worksheets**
- checking existence of, 287-288
 - copying data to separate worksheets, 257-258
 - copying data to separate worksheets without filters, 258-259
 - events, 120
 - filtering data to separate worksheets, 257-258
 - referencing ranges in other worksheets, 61
 - Select.Case statements, 307
 - separating into workbooks, 255-256

writing text files, 402

X

X button (userforms), disabling via API declarations, 470-471

XcelFiles website, 263

xlFilterCopy (Advanced Filter), 203

copying

columns, 203-204

subsets of columns, 204-206

reordering columns, 204-206

.xls files, 11

.xlsb files, 11

.xlsm files

concerns with, 11

macros, 11

public perception of, 11

saving files in, 11

.xlsx files, 10-11

XML (Extensible Markup Language)

exporting data to files, 259-260

groups (Developer tab), 10

macros4

Office (MS) add-ins, 525-526

Y-Z

years, grouping pivot tables to, 221-222