

Bonus Chapter 2

Ten VBA Tips and Tricks

In This Chapter

- ▶ Using helpful habits
 - ▶ Making your work more efficient
-

This chapter contains a list of ten clever tricks I've developed (or acquired from other users) over the years.

Getting VBA Help, Fast

When working in a VBA module, you can get instant help regarding a VBA object, property, or method. Just move the cursor to the word that interests you and press F1.

Speeding Up Your Macros

If you write a VBA macro that produces lots of on-screen action, you can speed things up significantly by turning off screen updating. To do so, execute this statement:

```
Application.ScreenUpdating = False
```

If your macro uses a custom dialog box, make sure to turn screen updating back on before displaying the UserForm. Otherwise, moving the dialog box on the screen leaves an ugly trail.

Avoiding Excel's Questions

Some VBA methods cause Excel to display a confirmation message, which requires the user to click a button. For example, the statement `ActiveSheet.Delete` always displays a dialog box that asks for confirmation.

To eliminate such confirmation messages, execute the following before the statement that causes the confirmation messages.

```
Application.DisplayAlerts = False
```

Use this statement to reinstate the confirmation messages:

```
Application.DisplayAlerts = True
```

Displaying One Procedure at a Time

Normally, a Code window in the Visual Basic Editor (VBE) shows all the procedures in the module, one after another. If you find this distracting, set things up so that only one procedure is visible.

- 1. Activate the VBE and choose Tools⇨Options.**
- 2. Click the Editor tab in the Options dialog box.**
- 3. Remove the check mark from the Default to Full Module View check box.**

Then you can use the drop-down lists at the top of the module window to select the procedure to view or edit.

Using With-End With

If you need to set a number of properties for an object, your code is easier to read and faster running if you use the With-End With construct. The following code doesn't use With-End With:

```
Selection.HorizontalAlignment = xlCenter  
Selection.VerticalAlignment = xlCenter  
Selection.WrapText = True  
Selection.Orientation = 0  
Selection.ShrinkToFit = False  
Selection.MergeCells = False
```

The next code performs the same action but is rewritten to use With-End With:

```
With Selection
    .HorizontalAlignment = xlCenter
    .VerticalAlignment = xlCenter
    .WrapText = True
    .Orientation = 0
    .ShrinkToFit = False
    .MergeCells = False
End With
```

Reducing the Size of a Workbook

In many cases, you can significantly reduce the size of a workbook — especially a workbook with modules you’ve heavily edited — because Excel does not do a good job of cleaning up after itself. To clean up the mess Excel leaves behind:

1. **Save your workbook.**
2. **Select a module or a UserForm in the Project Window.**
3. **Right-click and choose Remove from the shortcut menu.**
4. **When asked whether you want to export the module, click Yes.**
5. **Repeat Steps 3 and 4 for each module and UserForm, keeping track of the modules and forms that you remove.**
6. **Choose File⇨Import File to import all the modules and forms you deleted.**
7. **Save your workbook again.**

You usually find that the new workbook is much smaller than it was.

Another way to reduce the size of a workbook file is as follows:

1. **Activate your workbook.**
2. **Choose File⇨Save As Web Page, and make sure that you use the Entire Workbook option.**
3. **Close your workbook.**
4. **Use File⇨Open to open the HTML file that you saved in Step 2.**
5. **Use File⇨Save As, and resave the workbook as a standard XLS file.**

In most cases, you’ll find that the file is smaller in size.

Bypassing a Workbook_Open Procedure

Workbook_Open is a macro that Excel executes automatically when you open a workbook. In some situations, you may want to avoid running this macro. To do so, press the Shift key while opening the workbook.

Using Your Personal Macro Workbook

If you've developed some general-purpose macros, consider storing them in your Personal Macro Workbook, which opens automatically whenever Excel starts. When you record a macro, you have the option of recording it to your Personal Macro Workbook. The file, PERSONAL.XLS, is stored in your XLSTART directory. The Personal Macro Workbook is created the first time you record a macro to it.



The Personal Macro Workbook is hidden by default.

Displaying Messages in the Status Bar

If you develop a lengthy macro, use the Excel status bar to display text that describes the progress of the macro. To do so, write some code that periodically executes another macro, such as the following:

```
Sub UpdateStatusBar(PctDone)
    Application.StatusBar = "Percent Completed: " & Format(PctDone, "0%")
End Sub
```

This procedure uses one argument: a value that ranges from 0 to 1.0. The macro simply displays a message that indicates the percent completed. To return the status bar back to normal, execute the following statement:

```
Application.StatusBar = False
```

Forcing Yourself to Declare All Variables

Declaring every variable that you use in your code is an excellent practice. For example, if you use an integer variable named Count, declare it as Dim Count as Integer.

Declaring your variables as a particular data type makes your code run faster and also helps avoid typographical errors. To force yourself to declare all variables, insert the following statement at the top of your module:

```
Option Explicit
```

If you would like this statement automatically added to each new module, follow these steps:

- 1. Activate the VBE and choose Tools⇨Options.**
- 2. In the Options dialog box, click the Editor tab.**
- 3. Place a check mark next to Require Variable Declaration.**

