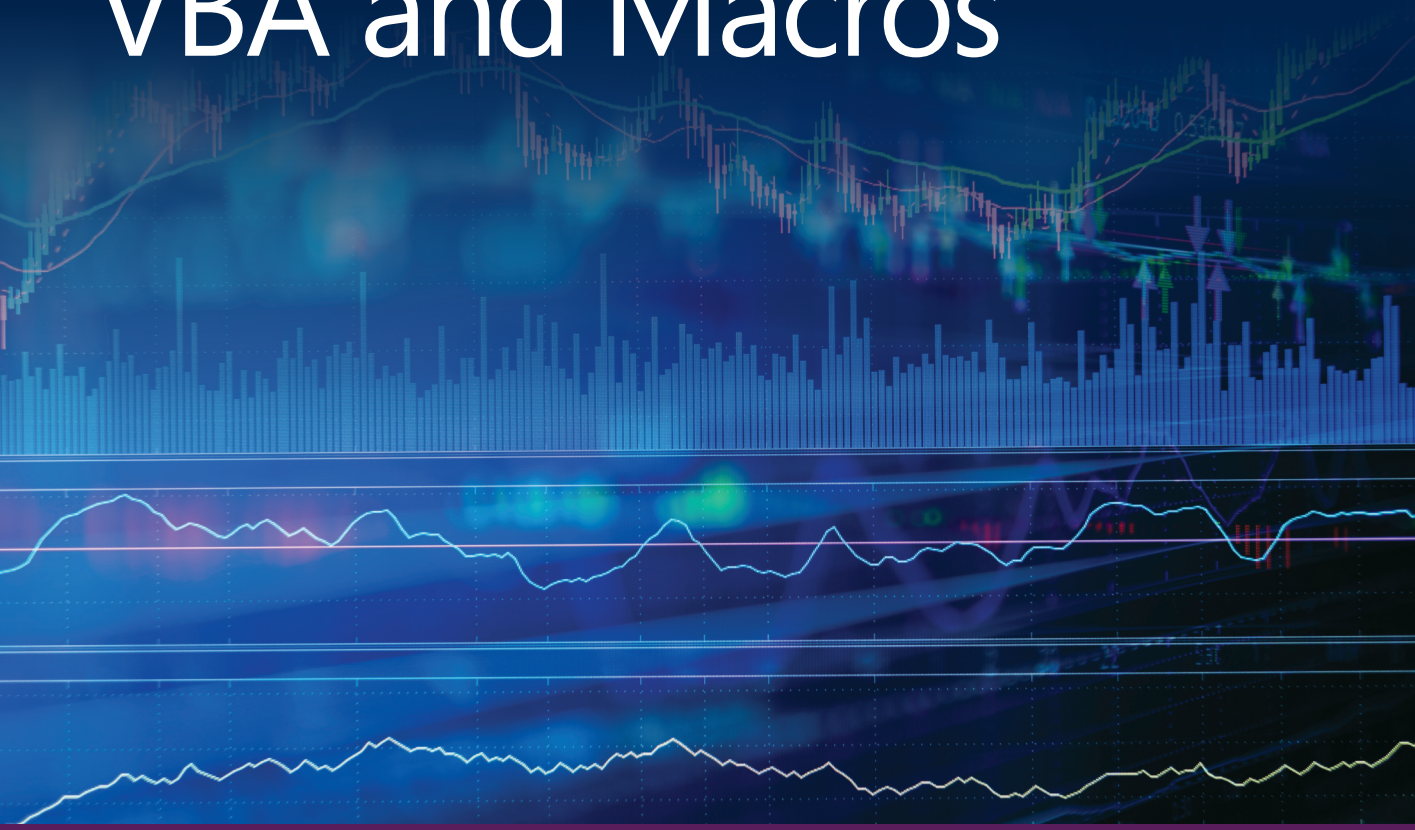
Microsoft

Microsoft Excel 2019

VBA and Macros



Bill Jelen and Tracy Syrstad

Sample files on the web

Microsoft

Microsoft Excel 2019 VBA and Macros

Bill Jelen
Tracy Syrstad

CHAPTER 10

Userforms: An introduction

In this chapter, you will:

- Use an input box to request user input
- Use a message box to display information
- Learn how to create a userform
- Add controls to the userform
- Verify a required field has an entry
- Prevent a user from closing a form
- Prompt the user to select a file

Userforms enable you to display information and allow the user to input information. Using `InputBox` and `MsgBox` controls are simple ways of doing this. You can use the userform controls in the VB Editor to create forms that are more complex.

This chapter covers simple user interfaces using input boxes and message boxes and the basics of creating userforms in the VB Editor.

Note To learn more about advanced userform programming, see Chapter 22, "Advanced userform techniques."

Input boxes

The `InputBox` function is used to create a basic interface element that requests input from the user before the program can continue. You can configure the prompt, the title for the window, a default value, the window position, and user help files. The only two buttons provided are the OK and Cancel buttons. The returned value is a string.

The following code asks the user for the number of months to be averaged. Figure 10-1 shows the resulting input box.

```
AveMos = InputBox(Prompt:="Enter the number " & " of months to average", _
    Title:="Enter Months", Default:="3")
```

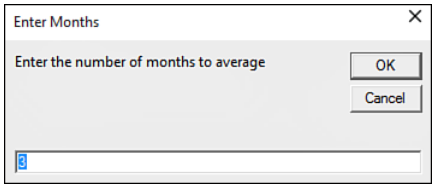


FIGURE 10-1 An input box can be simple but still effective.

Tip If you need to force the entry of a variable type other than string, use `Application.InputBox`. This method allows you to specify the return data type, including a formula, number, or cell reference.

Message boxes

The `MsgBox` function creates a message box that displays information and waits for the user to click a button before continuing. Whereas `InputBox` has only OK and Cancel buttons, the `MsgBox` function enables you to choose from several configurations of buttons, including Yes, No, OK, and Cancel. You also can configure the prompt, the window title, and help files. The following code produces a prompt to find out whether the user wants to continue. You use a `Select Case` statement to continue the program with the appropriate action:

```
myTitle = "Report Finalized"
MyMsg = "Do you want to save changes and close?"
Response = MsgBox(myMsg, vbExclamation + vbYesNoCancel, myTitle)
Select Case Response
     Case Is = vbYes
        ActiveWorkbook.Close SaveChanges:=True
     Case Is = vbNo
        ActiveWorkbook.Close SaveChanges:=False
     Case Is = vbCancel
        Exit Sub
End Select
```

Figure 10-2 shows the resulting customized message box.

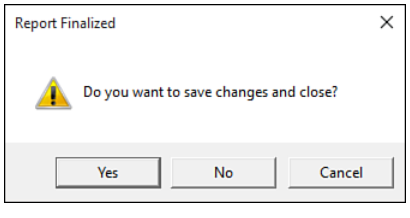


FIGURE 10-2 The `MsgBox` function is used to display information and obtain a basic response from the user.

Tip You can combine an icon option and a buttons option for the `buttons` argument by separating them with the plus (+) symbol. In the previous example, `vbExclamation + vbYesNoCancel` instructed Excel to show the exclamation symbol and the Yes, No, and Cancel buttons.

Creating a userform

Userforms combine the capabilities of `InputBox` and `MsgBox` to create a more efficient way of interacting with the user. For example, rather than have the user fill out personal information on a sheet, you can create a userform that prompts for the required data (see Figure 10-3).

FIGURE 10-3 Create a custom userform to get more information from the user.

Insert a userform in the VB Editor by selecting Insert, UserForm from the main menu. When a UserForm module is added to the Project Explorer, a blank form appears in the window where your code usually is, and the Controls toolbox appears.

To change the codename of the userform, select the form and change the (`Name`) property. The codename of a userform is used to refer to the form, as shown in the following sections. You can resize a userform by grabbing and dragging the handles on its right side, bottom edge, or lower-right corner. To add controls to the form, click the desired control in the toolbox and draw it on the form. You can move and resize controls at any time.

Note By default, the toolbox displays the most common controls. To access more controls, right-click the toolbox and select Additional Controls. However, be careful; other users might not have the same additional controls as you do. If you send users a form with a control they do not have installed, the program generates an error.

After you add a control to a form, you can change its properties from the Properties window. (Or, if you don't want to set the properties manually now, you can set them later programmatically.) If the Properties window is not visible, you can bring it up by selecting View, Properties Window. Figure 10-4 shows the Properties window for a text box.

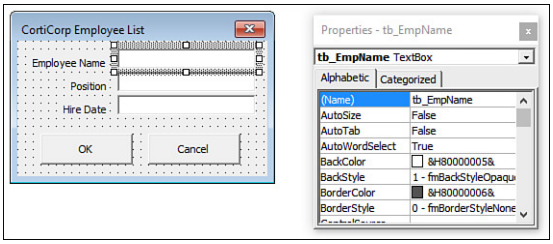


FIGURE 10-4 Use the Properties window to change the properties of a control.

Calling and hiding a userform

A userform can be called from any module. The syntax `FormName.Show` causes a form for the user to pop up:

```
frm_AddEmp.Show
```

The `Load` method can also be used to call a userform to place it in memory. It allows a form to be loaded while remaining hidden:

```
Load frm_AddEmp
```

To hide a userform, use the `Hide` method. When you do, the form is still active but is hidden from the user. However, the controls on the form can still be accessed programmatically:

```
frm_AddEmp.Hide
```

The `Unload` method unloads a form from memory and removes it from the user's view, which means the form cannot be accessed by the user or programmatically:

```
Unload Me
```

TIP `Me` is a keyword that can be used to refer to the userform. It can be used in the code of any control to refer to itself.

Programming userforms

The code for a control goes in the form's module. Unlike with the other modules, double-clicking the form's module opens the form in Design view. To view the code, you can right-click either the module or the userform in Design view and select View Code.

Userform events

Just like a worksheet, a userform has events that are triggered by actions. After the userform has been added to a project, the events are available in the Properties drop-down menu at the top right of the code window (see Figure 10-5); to access them, select UserForm from the Object drop-down menu on the left.

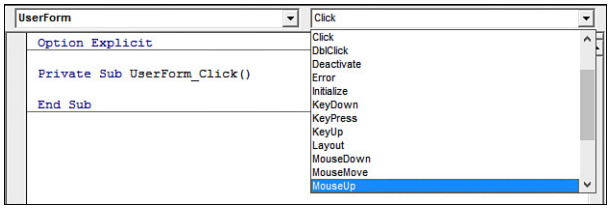


FIGURE 10-5 Various events for a userform can be selected from the drop-down menu at the top of the code window.

The available events for userforms are described in Table 10-1.

TABLE 10-1 Userform events

Event	Description
`Activate`	Occurs when a userform is either loaded or shown. This event is triggered after the `Initialize` event.
`AddControl`	Occurs when a control is added to a userform at runtime. Does not run at design time or upon userform initialization.
`BeforeDragOver`	Occurs while the user does a drag and drop onto the userform.
`BeforeDropOrPaste`	Occurs right before the user is about to drop or paste data into the userform.
`Click`	Occurs when the user clicks the userform with the mouse.
`DblClick`	Occurs when the user double-clicks the userform with the mouse. If a click event is also in use, the double-click event will not work.
`Deactivate`	Occurs when a userform is deactivated.
`Error`	Occurs when the userform runs into an error and cannot return the error information.
`Initialize`	Occurs when the userform is first loaded, before the `Activate` event.
`KeyDown`	Occurs when the user presses a key on the keyboard.
`KeyPress`	Occurs when the user presses an ANSI key. An ANSI key is a typable character such as the letter A. An example of a nontypable character is the Tab key.
`KeyUp`	Occurs when the user releases a key on the keyboard.
`Layout`	Occurs when the control changes size.
`MouseDown`	Occurs when the user presses the mouse button within the borders of the userform.
`MouseMove`	Occurs when the user moves the mouse within the borders of the userform.
`MouseUp`	Occurs when the user releases the mouse button within the borders of the userform.
`QueryClose`	Occurs before a userform closes. It allows you to recognize the method used to close a form and have code respond accordingly.
`RemoveControl`	Occurs when a control is deleted from within the userform.

`Resize`	Occurs when the userform is resized.
`Scroll`	Occurs when a visible scrollbar box is repositioned.
`Terminate`	Occurs after the userform has been unloaded. This is triggered after `QueryClose`.
`Zoom`	Occurs when the zoom value is changed.

Programming controls

To program a control, highlight the control and select View, Code. The footer, header, and default action for the control are entered in the programming field automatically. To see the other actions that are available for a control, select the control from the Object drop-down menu and view the actions in the Properties drop-down menu, as shown in Figure 10-6.

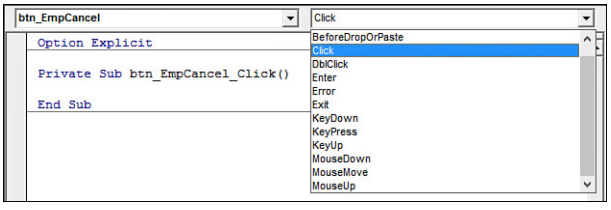


FIGURE 10-6 You can select various actions for a control from the VB Editor drop-down menus.

The controls are objects, like `ActiveWorkbook`. They have properties and methods that depend on the type of control. Most of the programming for the controls is done in the form's module. However, if another module needs to refer to a control, the parent, which is the form, needs to be included with the object. Here's an example of a button event that closes the form:

```
Private Sub btn_EmpCancel_Click()
Unload Me
End Sub
```

The preceding code can be broken down into three sections.

- `btn_EmpCancel`—Name given to the control
- `Click`—Action of the control
- `Unload Me`—Code behind the control, which, in this case, is unloading the form

Tip Change the (`Name`) property in the control's Properties window to rename a control from the default assigned by the editor.