

Class: MSc

Subject : Application of IT- Basics and Advance Excel

Chapter: Unit 4 Chapter 2

Chapter Name: Event Handling

What are VBA Workbook Events?

- *VBA workbook events are defined as an action performed by a user in Microsoft Excel that can trigger the execution of a specified macro. For example, when a user opens a workbook in Excel, the event “Workbook_Open” is triggered. Similarly, when the user saves the current workbook, the event “Workbook_BeforeSave” is initiated. There are many such events that are built into Excel VBA.*
- *Users can create codes for specific workbook events, such that if the user has specified the code for a particular event that has occurred, VBA will instantly execute the code. The code that is executed when an event occurs is referred to as an event handler.*
- *VBA workbook events allow users can create macros that are automatically executed by Excel when a particular event occurs. They improve user experience, and they make it possible to add interactivity to Excel workbooks.*

Types of Events in Excel

1. Application level events

Application events occur to the Microsoft Office application itself, such as Excel. Examples of application-level events include opening a new workbook, saving the current workbook, or closing one or more of the open workbooks.

2. Workbook level events

Workbook events occur due to the user's actions on the workbook itself. Examples of such events include creating a new worksheet, opening a workbook, and printing the workbook.

3. Worksheet level events

Worksheet events are events that are triggered when a user performs an action on a worksheet. Examples of worksheet level events include double-clicking on a cell, right-clicking on a cell, changing a cell in the worksheet, changing the color of a worksheet, etc.

Types of Events in Excel

4. UserForm level events

UserForm events are events that occur to the UserForm or an object (such as a button or cell) within the UserForm. An example of a UserForm event is clicking a cell in the UserForm.

5. Chart events

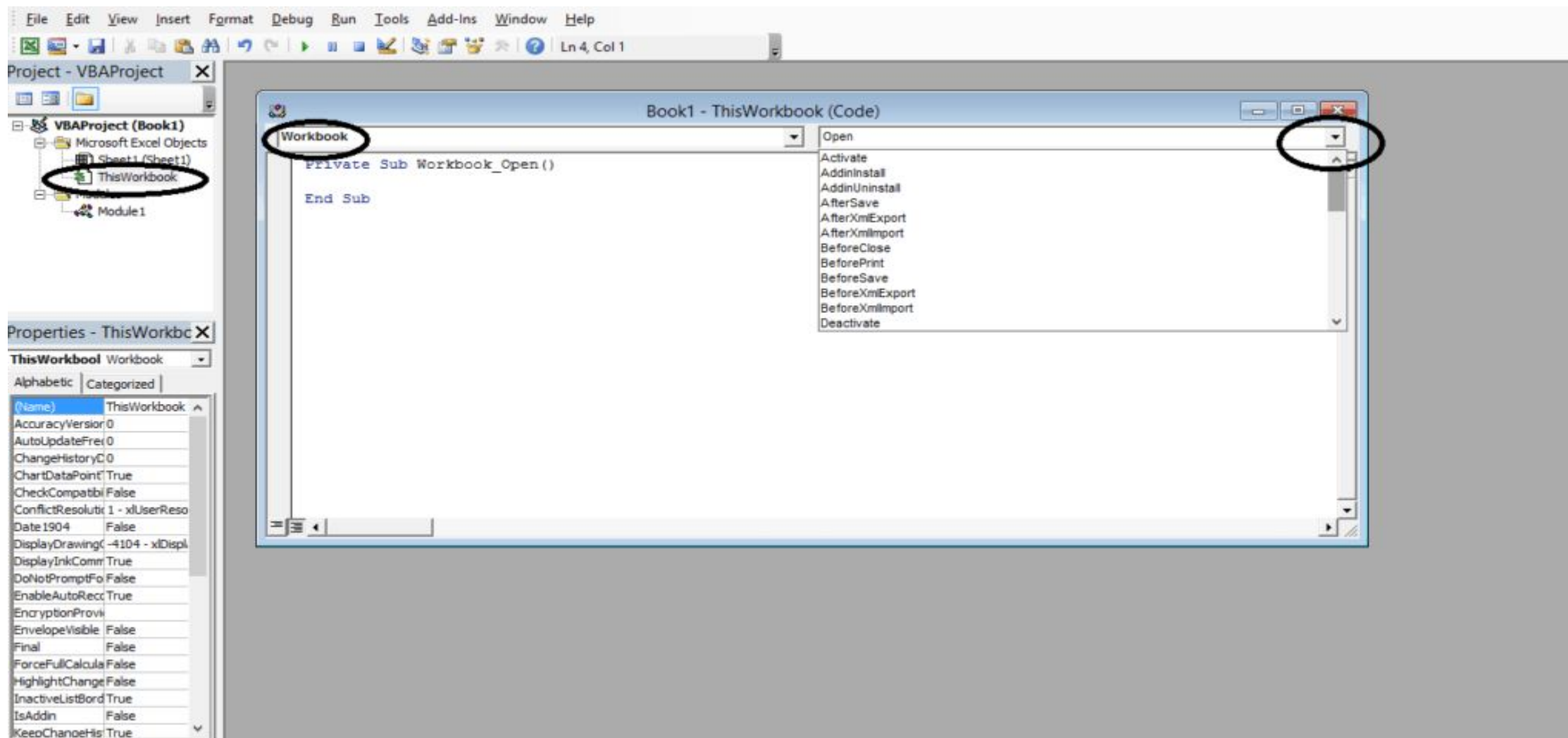
Chart events are events that occur on the chart sheet. A chart sheet is different from a worksheet, and its work is to hold charts. Examples of chart events include resizing a chart and changing the selection of a chart.

WorkBook Level Events

Follow the steps below to view the list of workbook events:

- Open the VBA window from the Developer tab.*
- Click on “ThisWorkbook” on the left-hand side below the Microsoft Excel Objects to open the code window.*
- On the Code window, select Workbook from the drop-down option on the left. It will show the Workbook_Open code in the code window.*
- Click the right-hand drop-down to see the list of workbook events.*

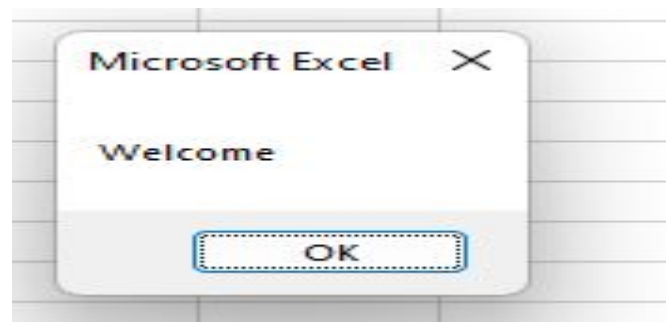
WorkBook Level Events



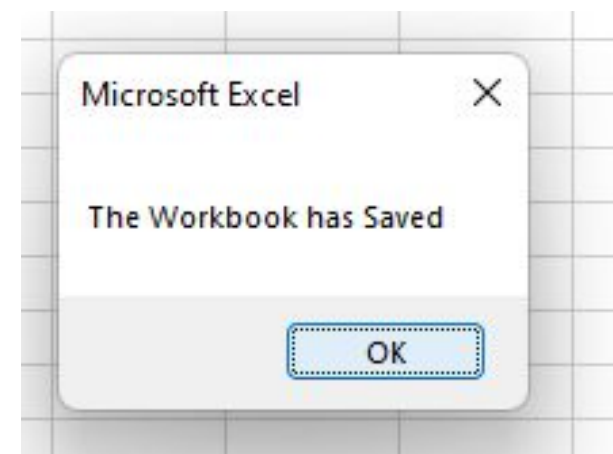
Examples

The following code will show a message “Welcome” whenever you open the workbook.

```
Private Sub Workbook_Open()  
MsgBox "Welcome"  
End Sub
```



```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As Boolean, Cancel As Boolean)  
MsgBox "The Workbook has Saved"  
End Sub
```



MsgBox

- The *MsgBox* function displays a message box and waits for the user to click a button and then an action is performed based on the button clicked by the user.
 - Syntax: **MsgBox(prompt[,buttons],[title],[helpfile,context])**
1. **Prompt** – A Required Parameter. A String that is displayed as a message in the dialog box. The maximum length of prompt is approximately 1024 characters.
 2. **Buttons** – An Optional Parameter. A Numeric expression that specifies the type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If left blank, the default value for buttons is 0.
 3. **Title** – An Optional Parameter. A String expression displayed in the title bar of the dialog box. If the title is left blank, the application name is placed in the title bar.
 4. **Helpfile** – An Optional Parameter. A String expression that identifies the Help file to use for providing context-sensitive help for the dialog box.
 5. **Context** – An Optional Parameter. A Numeric expression that identifies the Help context number assigned by the Help author to the appropriate Help topic. If context is provided, helpfile must also be provided.

Buttons in MsgBox

The **Buttons** parameter can take any of the following values –

- 0 vbOKOnly – Displays OK button only.
- 1 vbOKCancel – Displays OK and Cancel buttons.
- 2 vbAbortRetryIgnore – Displays Abort, Retry, and Ignore buttons.
- 3 vbYesNoCancel – Displays Yes, No, and Cancel buttons.
- 4 vbYesNo – Displays Yes and No buttons.
- 5 vbRetryCancel – Displays Retry and Cancel buttons.
- 16 vbCritical – Displays Critical Message icon.
- 32 vbQuestion – Displays Warning Query icon.
- 48 vbExclamation – Displays Warning Message icon.
- 64 vbInformation – Displays Information Message icon.
- 0 vbDefaultButton1 – First button is default.
- 256 vbDefaultButton2 – Second button is default.
- 512 vbDefaultButton3 – Third button is default.
- 768 vbDefaultButton4 – Fourth button is default.
- 0 vbApplicationModal Application modal – The current application will not work until the user responds to the message box.
- 4096 vbSystemModal System modal – All applications will not work until the user responds to the message box.

Return Values

The MsgBox function can return one of the following values which can be used to identify the button the user has clicked in the message box.

1 - vbOK - OK was clicked

2 - vbCancel - Cancel was clicked

3 - vbAbort - Abort was clicked

4 - vbRetry - Retry was clicked

5 - vbIgnore - Ignore was clicked

6 - vbYes - Yes was clicked

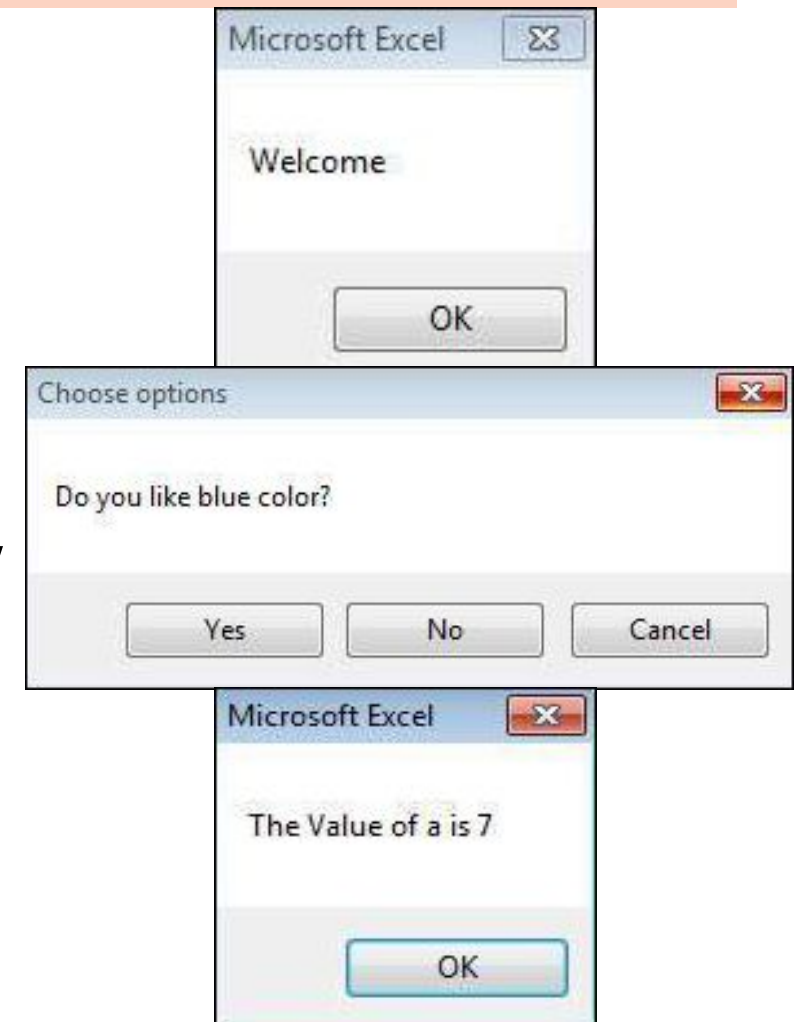
7 - vbNo - No was clicked

Example

```
Function MessageBox_Demo()  
    'Message Box with just prompt message  
    MsgBox("Welcome")  
  
    'Message Box with title, yes no and cancel Buttons  
    int a = MsgBox("Do you like blue color?",3,"Choose options")  
    ' Assume that you press No Button  
    msgbox ("The Value of a is " & a)  
End Function
```

Output

- *Step 1 – The above Function can be executed either by clicking the "Run" button on VBA Window or by calling the function from Excel Worksheet as shown in the following screenshot.*
- *Step 2 – A Simple Message box is displayed with a message "Welcome" and an "OK" Button*
- *Step 3 – After Clicking OK, yet another dialog box is displayed with a message along with "yes, no, and cancel" buttons.*
- *Step 4 – After clicking the 'No' button, the value of that button (7) is stored as an integer and displayed as a message box to the user as shown in the following screenshot. Using this value, it can be understood which button the user has clicked.*



InputBox

- The `InputBox` function prompts the users to enter values. After entering the values, if the user clicks the OK button or presses ENTER on the keyboard, the `InputBox` function will return the text in the text box. If the user clicks the Cancel button, the function will return an empty string ("").
- Syntax: `InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])`
- **Prompt** – A required parameter. A String that is displayed as a message in the dialog box. The maximum length of prompt is approximately 1024 characters.
- **Title** – An optional parameter. A String expression displayed in the title bar of the dialog box. If the title is left blank, the application name is placed in the title bar.
- **Default** – An optional parameter. A default text in the text box that the user would like to be displayed.
- **XPos** – An optional parameter. The position of X axis represents the prompt distance from the left side of the screen horizontally. If left blank, the input box is horizontally centered.
- **YPos** – An optional parameter. The position of Y axis represents the prompt distance from the left side of the screen vertically. If left blank, the input box is vertically centered.

Example

Let us calculate the area of a rectangle by getting values from the user at run time with the help of two input boxes (one for length and one for width).

```
Function findArea()  
    Dim Length As Double  
    Dim Width As Double  
  
    Length = InputBox("Enter Length ", "Enter a Number")  
    Width = InputBox("Enter Width", "Enter a Number")  
    findArea = Length * Width  
End Function
```

Output

- Step 1 – To execute the same, call using the function name and press Enter as shown in the following screenshot.
- Step 2 – Upon execution, the First input box (length) is displayed. Enter a value into the input box.
- Step 3 – After entering the first value, the second input box (width) is displayed.
- Step 4 – Upon entering the second number, click the OK button. The area is displayed as shown in the following screenshot.

The first screenshot shows a dialog box titled "Enter Value" with a close button (X). It contains a label "Enter Length" and an input field with the value "7". There are "OK" and "Cancel" buttons.

The second screenshot shows a similar dialog box titled "Enter Value" with a close button (X). It contains a label "Enter Width" and an input field with the value "4". There are "OK" and "Cancel" buttons.

The third screenshot shows a spreadsheet with columns B, C, D, and E. In cell D2, the text "Calculate Area" is displayed above the value "28", which is circled in red.

B	C	D	E
		Calculate Area	
		28	