**Module 10: Excel Events Programming**

**Definition**

**Excel Events Programming** refers to writing VBA (Visual Basic for Applications) code that **automatically runs** when a specific **event or action** occurs in Excel.

These events can include:

* Opening or closing a workbook
* Changing cell value
* Selecting a different worksheet
* Clicking on an object (like a button)
* Double-clicking or right-clicking on a cell

In short, **event-driven programming** makes Excel **respond automatically** to user actions or system events.

Types of Excel Events

1. Workbook Events — Triggered by actions at the workbook level.  
   Example: Workbook\_Open, Workbook\_BeforeClose
2. Worksheet Events — Triggered by actions on a specific sheet.  
   Example: Worksheet\_Change, Worksheet\_SelectionChange
3. Chart Events — Triggered when you interact with charts.
4. Application Events — Triggered at the Excel application level.  
   Example: Detect when any workbook is opened.

**Example 1: Worksheet\_Change Event**

Purpose: Automatically display a message whenever a user changes any cell value.

' --- Code goes inside Sheet1 module ---

Private Sub Worksheet\_Change(ByVal Target As Range)

MsgBox "You just changed cell " & Target.Address, vbInformation, "Cell Changed!"

End Sub

Explanation:

* Target represents the cell or range that was changed.
* When a user edits any cell, a message box appears showing which cell was modified.

Step to using this code in VBA

1.Press **Alt + F11** to open the VBA editor.

2. In the **Project Explorer** (left side):

* Find your workbook name (e.g., *VBAProject (Book1.xlsm)*).
* Expand it → You’ll see folders like:
* Microsoft Excel Objects
* Sheet1 (Sheet1)
* Sheet2 (Sheet2)
* ThisWorkbook

3.**Double-click Sheet1 (Sheet1)** (or the sheet you want the event to work on).

4.**Paste your code there:**

**Example 2: Workbook\_Open Event**

**Purpose:** Display a welcome message when the Excel file opens.

' --- Code goes inside ThisWorkbook module ---

Private Sub Workbook\_Open()

MsgBox "Welcome, Amit! This workbook is now ready to use.", vbInformation, "Excel Events Demo"

End Sub

**Explanation:**

* This code runs automatically **when the workbook is opened**.
* Commonly used for initialization, setting default values, or updating data links.

**Example 3: Worksheet\_SelectionChange Event**

**Purpose:** Highlight the active cell with a different color automatically.

' --- Code goes inside Sheet1 module ---

Private Sub Worksheet\_SelectionChange(ByVal Target As Range)

Cells.Interior.ColorIndex = 0 ' Clear all background colors

Target.Interior.Color = vbYellow ' Highlight the active cell

End Sub

**Explanation:**

* Clear any previous highlighting.
* Highlights the currently selected cell in yellow.

**Example 4: Workbook\_BeforeClose Event**

**Purpose:** Ask for confirmation before closing the workbook.

' --- Code goes inside ThisWorkbook module ---

Private Sub Workbook\_BeforeClose(Cancel As Boolean)

Dim ans As VbMsgBoxResult

ans = MsgBox("Do you really want to close this workbook?", vbYesNo + vbQuestion, "Confirm Close")

If ans = vbNo Then

Cancel = True ' Cancel the closing action

End If

End Sub

**Explanation:**

* The event triggers **just before** the workbook closes.
* If the user clicks **No**, the workbook remains open.

**Application-level events**

Workbook or Worksheet events (like Workbook\_Open, Worksheet\_Change) are tied to a specific file.  
But **Application-level events** are global — they respond to events triggered in any open workbook (like when *any* workbook opens, closes, or a sheet is activated).

So you can monitor or control Excel’s behavior *as a whole*, not just one file. 💡

**Step 1: Insert a Class Module**

1. In VBA Editor → **Insert → Class Module→Click once on Class1→Go to View→Properties window**

**→change “Class1” →clsAppEvents→Press Enter.**

**Double click on clsAppEvents**

**Step 2: Add This Code Inside clsAppEvents**

' --- Code inside Class Module: clsAppEvents ---

Public WithEvents App As Application

Private Sub App\_WorkbookOpen(ByVal Wb As Workbook)

MsgBox "Workbook opened: " & Wb.Name, vbInformation, "Application-Level Event"

End Sub

Private Sub App\_WorkbookBeforeClose(ByVal Wb As Workbook, Cancel As Boolean)

MsgBox "Closing workbook: " & Wb.Name, vbExclamation, "Application-Level Event"

End Sub

Private Sub App\_SheetChange(ByVal Sh As Object, ByVal Target As Range)

MsgBox "You changed cell " & Target.Address & " in " & Sh.Name, vbInformation, "App Event Triggered"

End Sub

**Initialize the Event Listener in ThisWorkbook**

In the same project, double-click **ThisWorkbook** and paste:

' --- Code inside ThisWorkbook ---

Public XLApp As New clsAppEvents

Private Sub Workbook\_Open()

Set XLApp.App = Application

MsgBox "Application-level events are now active!", vbInformation

End Sub

1.Save as **.xlsm** (Macro-Enabled Workbook)

2.Close and reopen the workbook

3.You’ll see:  
🔹 “Application-level events are now active!”

4.Now try:

* Open **any** other workbook → you’ll get “You just opened: …”
* Switch sheets in any workbook → you’ll get “You switched to sheet: …”

**Mini Project: Automatic Data Entry Tracker**

**🎯 Objective**

Whenever a user enters any data into the “Data Entry” sheet, Excel will automatically:

1. Record the entry details (cell, old value, new value, user name, and timestamp)
2. Save them in a separate sheet named “Change Log”
3. Prevent blank entries from being entered accidentally

**🧱 Project Setup Steps**

**Step 1: Create the Workbook**

1. Open a new Excel workbook.
2. Rename:
   * Sheet1 → **Data Entry**
   * Sheet2 → **Change Log**
3. In **Change Log**, create headers:
4. A1 → Date & Time
5. B1 → Cell Changed
6. C1 → Old Value
7. D1 → New Value

8. E1 → User Name

**Step 2: Insert the VBA Code**

**🧾 Code 1: In Sheet (Data Entry) module**

Go to **VBA Editor → Project Explorer → Microsoft Excel Objects → Sheet (Data Entry)**  
Then paste this code:

Option Explicit

Private OldValue As Variant

' Capture old value before change

Private Sub Worksheet\_SelectionChange(ByVal Target As Range)

On Error Resume Next

OldValue = Target.Value

End Sub

' Log every change automatically

Private Sub Worksheet\_Change(ByVal Target As Range)

Dim wsLog As Worksheet

Set wsLog = ThisWorkbook.Sheets("Change Log")

Dim NewValue As Variant

NewValue = Target.Value

' Prevent blank entry

If Trim(NewValue) = "" Then

MsgBox "Blank entries are not allowed!", vbExclamation, "Invalid Entry"

Application.EnableEvents = False

Application.Undo

Application.EnableEvents = True

Exit Sub

End If

' Add log details

Dim NextRow As Long

NextRow = wsLog.Cells(wsLog.Rows.Count, "A").End(xlUp).Row + 1

wsLog.Cells(NextRow, 1).Value = Now

wsLog.Cells(NextRow, 2).Value = Target.Address

wsLog.Cells(NextRow, 3).Value = OldValue

wsLog.Cells(NextRow, 4).Value = NewValue

wsLog.Cells(NextRow, 5).Value = Environ("Username")

End Sub

**Step 3: Test the Project**

1. Go to the **Data Entry** sheet.
2. Try changing any cell value.
3. Open the **Change Log** sheet — you’ll see a new row logged automatically!

If you enter a blank value, it will:

* Show a warning message
* Undo the blank entry