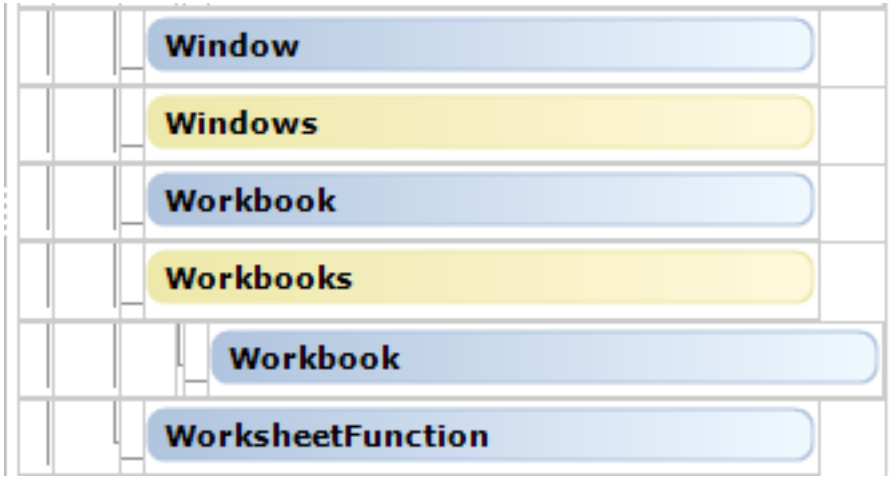
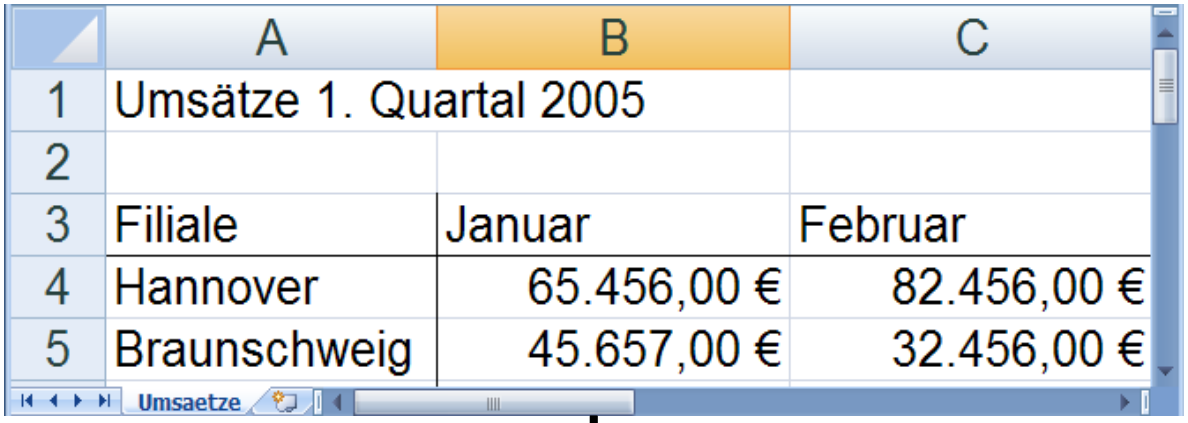
Excel – VBA
Objekte



Objekte

- ... sind Bestandteile der Anwendung Excel oder anderer Office-Anwendungen.
- Die Anwendung selbst ist wiederum ein Objekt, welches andere Objekte enthält.
- ... sind Substantive in einem Text.
- ... haben bestimmte Eigenschaften (Attribute) und Methoden (Funktionen).
- ... berichten über ihren Zustand und können diesen über Methoden verändern.
- ... reagieren auf Aktionen mit Hilfe von Ereignissen.

Beispiel: Arbeitsblatt



	A	B	C
1	Umsätze 1. Quartal 2005		
2			
3	Filiale	Januar	Februar
4	Hannover	65.456,00 €	82.456,00 €
5	Braunschweig	45.657,00 €	32.456,00 €

Umsaetze

Eigenschaften:
- Cells (Zellen)
- Columns (Spalten)
- Rows (Zeilen)
- Name

Methoden:
- Activate
- Delete
- PrintOut
- SaveAs

Ereignisse:
- Activate
- Calculate
- Change
- SelectionChange

Objekte referenzieren

```
ThisWorkbook.Worksheets("Bezirk").Range("A1:B5")
```

- In Abhängigkeit der Objekthierarchie von Excel werden die einzelnen Objekte referenziert.
- In diesem Beispiel wird die Hierarchie
  Arbeitsmappe ⇒ Arbeitsblatt ⇒ Zellbereich abgebildet.
- Jedes Eltern-Objekt wird von seinem Kind-Objekt mit Hilfe des Punktes getrennt.
- Die Referenzierung beginnt bei der Wurzel und endet bei einem bestimmten Blatt.

Objektvariablen deklarieren

```
Dim objTabellenblatt As Worksheet
Dim objZellbereich As Range

Set objTabellenblatt = ThisWorkbook.Worksheets("Bezirk")
Set objZellbereich = objTabellenblatt.Range("A1:B5")
```

- Objektvariablen werden genauso wie alle anderen Variablen deklariert.
- Als Typ wird statt des Standard-Datentyps eine Objektart angegeben. In diesem Beispiel wird der Variablen
  - ... objTabellenblatt ein Verweis auf ein Arbeitsblatt und
  - ... objZellbereich ein Verweis auf ein Zellbereich übergeben.

Objektvariablen setzen

```
Dim objTabellenblatt As Worksheet
Dim objZellbereich As Range

Set objTabellenblatt = ThisWorkbook.Worksheets("Bezirk")
Set objZellbereich = objTabellenblatt.Range("A1:B5")
```

- Setze (Set) objektvariable = Referenz.
- Die Objektvariable
  - ... bekommt ein Verweis auf ein Objekt zugewiesen.
  - ... enthält eine bestimmte Referenz auf ein definiertes Objekt von einem bestimmten Typ.
- Der Objekttyp muss der Programmiersprache VBA bekannt sein.

Objektvariable zurücksetzen

```
Dim objTabellenblatt As Worksheet
Dim objZellbereich As Range

Set objTabellenblatt = ThisWorkbook.Worksheets("Bezirk")
Set objZellbereich = objTabellenblatt.Range("A1:B5")


objZellbereich = Nothing
```

Die Verbindung zwischen der Variablen und dem Objekt wird aufgehoben. Die Objektvariable verweist auf kein Objekt.

Eigenschaften (Attribute)

- ... beschreiben das Aussehen eines Objekts.
- ... beschreiben einen Gegenstand.
- ... sind statische Werte, die ein Objekt kennzeichnen.
- ... beschreiben den aktuellen Zustand eines Objekts.
- Eigenschaftswerte können immer gelesen werden. Einige Eigenschaftswerte können mit Hilfe von VBA verändert werden.
- Mit Hilfe des Gleichheitszeichens wird der Eigenschaftswert verändert oder einer anderen Variablen übergeben.
- ... werden von dem dazugehörigen Objekt mit einem Punkt getrennt.

Eigenschaften von Objekten ändern.

```
ThisWorkbook.Worksheets("Bezirk").Range("A1:B5").Font.Size = 20
```

- In diesem Beispiel wird die Schriftgröße des angegebenen Zellbereichs verändert.
- Mit Hilfe des Gleichheitszeichen wird der Eigenschaft ein neuer Wert zugewiesen.
- Die Eigenschaft und das dazugehörige Objekt werden durch einen Punkt getrennt.

Objektvariablen nutzen

```
Dim objTabellenblatt As Worksheet
Dim objZellbereich As Range

Set objTabellenblatt = ThisWorkbook.Worksheets("Bezirk")
Set objZellbereich = objTabellenblatt.Range("A1:B5")

objZellbereich.Font.Size = 12

objZellbereich = N
```

Jede Objektbezeichnung kann durch eine Objektvariable ersetzt werden. Die Nutzung von Objektvariablen erhöht die Lesbarkeit des Programms.

With... End With

```
Dim objTabellenblatt As Worksheet
Dim objZellbereich As Range

Set objTabellenblatt = ThisWorkbook.Worksheets("Bezirk")
Set objZellbereich = objTabellenblatt.Range("A1:B5")

With objZellbereich.Font
    .Size = 12
    .Name = "Courier"
    .Color = vbRed
End With

objZellbereich = Nothing
```

... fasst Anweisungen für ein bestimmtes Objekt zusammen. Hier wird die Schrift für einen definierten Zellbereich eingestellt.

For-Each-Schleife nutzen

```
Sub allTabellen()
    Dim objArbeitsmappe As Workbook
    Dim objTabellenblatt As Worksheet

    Set objArbeitsmappe = ThisWorkbook

    For Each objTabellenblatt In objArbeitsmappe.Worksheets

        Debug.Print objTabellenblatt.Name

    Next objTabellenblatt

End Sub
```

Alle Elemente einer Auflistung können mit Hilfe der Schleife durchlaufen werden.

Methoden (Member, Elementfunktion)

- ... beschreiben das Verhalten eines Objekts.
- ... verändern oder lesen Attribute des dazugehörigen Objekts.
- ... sind Prozeduren, die an ein Objekt gebunden sind.
- ... ermöglichen die Kommunikation mit anderen Objekten.
- Als Trennzeichen zwischen den Objekten und deren Methoden wird ein Punkt genutzt.
- ... werden wie benutzerdefinierte Prozeduren aufgerufen.

Beispiel: Zellen aktivieren

```
Sub aktivTabellenblatt()
    Dim tabellenblatt As Worksheet
    Dim zellbereich As Range

    Set tabellenblatt = ThisWorkbook.Worksheets("Bezirk")
    tabellenblatt.Activate

    Set zellbereich = tabellenblatt.Range("A1:B5")
    zellbereich.Select
    zellbereich.Activate
End Sub
```

Es wird immer nur eine Zelle aktiviert. In einem Zellbereich wird die linke obere Zelle aktiviert. Das Objekt ActiveCell beschreibt die aktive Zelle.

Application

- ... beschreibt die Anwendung selber.
- ... hat Eigenschaften und Methoden,
  - ... die Informationen zur Anwendung liefern.
  - ... die das Aussehen der Anwendung beeinflussen.
- Active... beschreibt Objekte, die innerhalb der Anwendung aktiv sind.

Eigenschaften und Methoden

- Application.Path gibt den Speicherort der Anwendung zurück.
- Application.PathSeparator gibt das Trennzeichen zwischen Dateien und Ordnern wieder.
- Application.Cursor verändert den Mauszeiger.
- Application.Quit schließt Excel.
- Application.DisplayAlerts = False unterdrückt Warnmeldungen der Anwendung.

Aktive Objekte

- ActiveWindow definiert das aktive Fenster.
- ActiveWorkbook beschreibt die aktive Arbeitsmappe.
- ActiveSheet enthält ein Verweis auf das aktive Arbeitsblatt.
- ActiveCell ist ein Verweis auf die aktive Zelle in einem Arbeitsblatt.

Arbeitsmappe (Workbook)

- ... ist ein Container für alle Arbeitsblätter in Excel.
- ... besteht standardmäßig aus drei Arbeitsblättern.
- ... fasst den Inhalt einer Excel-Datei zusammen.
- ... wird mit der Dateiendung "xls" oder ".xlsm" gespeichert.

Arbeitsblätter und Diagrammblätter

- Die Auflistung Sheets enthält alle Arbeitsblätter und Diagrammblätter.
- Die Auflistung Worksheets
  - ... enthält alle Arbeitsblätter in Tabellenform.
  - Die Informationen werden in Zeilen und Spalten geschrieben.
- Die Auflistung Charts
  - ... enthält alle Diagrammblätter der Excel-Datei.
  - Diagramme bereiten die Daten visuell auf.

Wo befindet sich die aktive Zelle?

ActiveCell.Parent.Name

- Die Eigenschaft / das Objekt Parent
  - ... bezeichnet immer das Arbeitsblatt, in dem sich die aktive Zelle befindet.
  - ... enthält ein Verweis auf die Eltern des definierten Objekts.
- Die Eigenschaft Name gibt immer die Bezeichnung eines Objekts zurück. In diesem Beispiel wird der Name des aktiven Arbeitsblatts zurückgegeben.
- Die Eigenschaft ActiveCell.Parent.Name und ActiveSheet.Name liefern das gleiche Ergebnis.

Arbeitsblatt aktivieren

```
Sub aktivTabellenblatt()
    Dim tabellenblatt As Worksheet
    Dim zellbereich As Range

    Set tabellenblatt = ThisWorkbook.Worksheets("Bezirk")
    tabellenblatt.Activate
End Sub
```

Das Arbeitsblatt ist aktiviert.
Das aktivierte Arbeitsblatt liegt in der Arbeitsmappe im Vordergrund.

Absolute Positionierung

- [Objekt].Range("[Zelle]") und [Objekt].Cell([Zeile], [Spalte]) beschreibt die Position einer Zelle.
- [Objekt].Range("[Zelle] : [Zelle]") definiert einen zusammenhängenden Zellbereich. Es wird die Zelle in der oberen linken Ecke und die Zelle in der unteren rechten Ecke angegeben.
- [Objekt].Range("[Zelle], [Zelle]") wählt verschiedene nicht zusammenhängende Zellen aus.
- [Objekt].Range("[Zelle]:[Zelle] [Zelle]:[Zelle]") nutzt die Schnittmenge von verschiedenen zusammenhängenden Zellbereichen.
- [Objekt].Range("[Zelle], [Zelle]:[Zelle]") nutzt alle angegebenen Zellbereiche.

Zellen berechnen

Range(Cells(1, 1), Cells(2, 5))

- ... bietet die Möglichkeit mit Hilfe von berechneten Indizes Zellen zu adressieren.
- In diesem Beispiel wird ein Zellbereich von A1 bis E2 genutzt.

Relative Positionierung

```
Sub aktivZell()
    Dim spalte As Long
    Dim zeile As Long
    Dim zellbereich As Range

    spalte = ActiveCell.Row
    zeile = ActiveCell.Column
    Set zellbereich = ActiveCell.Offset(zeile + 1, spalte + 1)

    Debug.Print zellbereich.Address
End Sub
```

Nutzung der Methode Offset

[Objekt].Offset([zeile], [spalte])

- In Abhängigkeit des angegebenen Objekts wird der Zeiger neu positioniert.
- Der Methode verschiebt den Zeiger um eine bestimmte Anzahl von Zeilen und / oder Spalten.
  - Ein Versatz von 0 verschiebt die aktuelle Zeile oder Spalte nicht.
  - Ein positiver Versatz rückt den Zeiger spaltenweise nach rechts oder zeilenweise nach unten.
  - Ein negativer Versatz rückt den Zeiger spaltenweise nach links oder zeilenweise nach oben.

Zellen auswählen

```
Sub aktivTabellenblatt()
    Dim tabellenblatt As Worksheet
    Dim zellbereich As Range

    Set tabellenblatt = ThisWorkbook.Worksheets("Bezirk")
    tabellenblatt.Activate

    Set zellbereich = tabellenblatt.Range("A1:B5")
    zellbereich.Select
    zellbereich.Activate
End Sub
```

Es kann eine Zelle oder eine Zelle ausgewählt werden.

Benutzten Zellbereich

- ActiveSheet.UsedRange selektiert alle Zellen, die einen Wert enthalten. Nicht beachtet werden eingebundene Objekte wie Zeichnungen etc. Die Eigenschaft ist teilweise unberechenbar, weil teilweise gelöschte Inhalte und Formate mit berücksichtigt werden.
- Range("C2").CurrentRegion.Select selektiert alle um die Zelle C2 liegenden Zellen, die einen Wert enthalten.
-

Zellen aktivieren

```
Sub aktivTabellenblatt()
    Dim tabellenblatt As Worksheet
    Dim zellbereich As Range

    Set tabellenblatt = ThisWorkbook.Worksheets("Bezirk")
    tabellenblatt.Activate

    Set zellbereich = tabellenb
    zellbereich.Select
    zellbereich.Activate
End Sub
```

Es wird immer nur eine Zelle aktiviert. In diesem Beispiel kann die Zelle A1 bearbeitet. In einem Zellbereich ist standardmäßig die linke, obere Zelle aktiv. Das Objekt ActiveCell beschreibt die aktive Zelle.

Position einer Zelle oder Zellbereich ermitteln

- ActiveWindow.RangeSelection.Address gibt die Absolutadresse des ausgewählten Zellbereichs an. Der ausgewählte Bereich ist durch einen Rahmen gekennzeichnet.
- ActiveCell.Address gibt die Absolutadresse der aktiven Zelle zurück. Die Zelle ist ausgewählt und bekommt die Einfügemarke. In einem Zellbereich ist standardmäßig die linke, obere Zelle aktiv.
- ActiveCell.Row gibt die Zeilennummer der aktiven Zelle zurück.
- ActiveCell.Column gibt die Spaltennummer der aktiven Zelle zurück.

Bestimmte Zellen ermitteln

- Cells(Rows.Count, 2).End(xlUp).Activate aktiviert die letzte Zelle mit Inhalt in der zweiten Spalte. Rows.Count gibt die Anzahl der Zeilen, egal ob leer oder mit Inhalt an.
- Cells(Rows.Count, 2).End(xlUp).Offset(1, 0).Activate aktiviert die erste leere Zelle in der zweiten Spalte.
- Cells(1, Columns.Count).End(xlToLeft).Offset(0, 1).Activate aktiviert die erste leere Zelle in der ersten Zeile.
- ActiveSheet.Cells.SpecialCells(xlCellTypeLastCell).Activate wird die letzte Zelle mit Inhalt in dem aktiven Arbeitsblatt markiert.

Inhalt einer Zelle

```
Sub ZellWert()
    Dim spalte As Long
    Dim zeile As Long
    Dim zellbereich As Range

    spalte = ActiveCell.Row
    zeile = ActiveCell.Column

     Set zellbereich = ActiveCell.Offset(spalte + 1, zeile + 1)

    zellbereich.Value = 4
End Sub
```

Value oder der Name des Range gibt den Inhalt der Zelle zurück.
Für das Auslesen der Werte muss die Zelle nicht aktiv sein.

Inhalt kopieren und einfügen

```
Sub ZellKopieren()
Dim quelle As Range
Dim ziel As Range

...

If (quelle.Value <> "") Then
    Set ziel = quelle.Offset(1, 1)
    quelle.Copy
    ActiveSheet.Paste Destination:=ziel
End If

End Sub
```

Der Inhalt der Zelle quelle wird in die Zwischenablage kopiert

... und anschließend aus der Zwischenablage in die Zelle ziel eingefügt.

Inhalt ausschneiden und einfügen

```
Sub ZellKopieren()
Dim quelle As Range
Dim ziel As Range

Set quelle = ActiveCell
Set ziel = ActiveCell.Offset(1, 1)

If quelle.Value <> "" Then
   quelle.Cut Destination:=ziel
End If

End Sub
```

Der Inhalt der Zelle quelle wird in die Zelle ziel kopiert. Anschließend bekommt quelle ein Verweis auf die Zelle ziel übergeben.

Parameter Destination

- ... kann für alle drei Methoden genutzt werden.
- Wo wird der Inhalt der Zwischenablage abgelegt?
- Wohin werden die Daten kopiert?

Reihen, Spalten und Zellen hinzufügen

- ActiveCell.EntireRow.Insert
  - Es wird oberhalb der aktuellen Zeile eine neue Zeile eingefügt.
- ActiveCell.EntireColumn.Insert
  - Es wird links von der aktuellen Spalte eine neue Spalte eingefügt.
- ActiveCell.Insert Shift:=xlShiftDown
  - Es wird eine neue Zelle eingefügt.
  - Die übrigen Zellen werden nach unten verschoben.
  - Andere Möglichkeit: xlShiftToRight.

Löschen

- ActiveCell.Clear
  - ... löscht den Inhalt und die Formatierungen der Zelle.
- ActiveCell.ClearContents
  - ... löscht den Inhalt der aktiven Zelle. Die Formatierungen bleiben erhalten.
- ActiveCell.ClearFormats
  - ... löscht die Formatierung der aktiven Zelle. Die Inhalte bleiben erhalten.
- ActiveCell.EntireRow.Delete
  - ... löscht die Zeile, in der sich die aktive Zelle befindet.
- ActiveCell.EntireColumn.Delete
  - ... löscht die Spalte, in der sich die aktive Zelle befindet.

Text mit Hilfe der Schleife suchen

```
For Each zelle In suchRange

    If LCase(zelle.Value) Like "*" & suchmuster & "*" Then
        tmpText = zelle.Value
        pos = InStr(LCase(tmpText), suchmuster)
        zelle.Value = Left(tmpText, pos - 1)
        zelle.Value = zelle.Value & "Ökonomie"

        If Len(tmpText) > (pos + laenge) Then
            zelle.Value = zelle.Value & Right(tmpText, _
                                Len(tmpText) - pos - laenge + 1)
        End If

    End If
Next zelle
```

Zahlen mit Hilfe der Schleife suchen

```
For Each zelle In suchRange

    If IsNumeric(zelle.Value) = True Then

        If zelle.Value > 70000 Then
            zelle.Interior.Color = RGB(220, 220, 220)
        End If

    End If

Next zelle
```

Kann der Inhalt der Zelle als Zahl interpretiert werden?

Formatierungen suchen

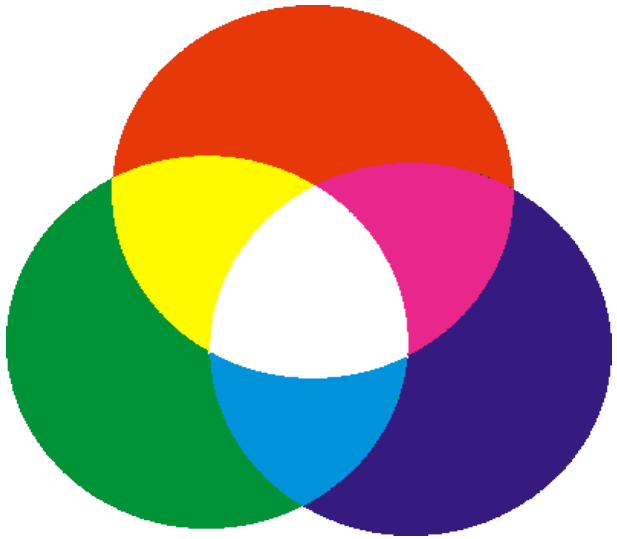
```
For Each zelle In suchRange

  If zelle.Interior.Color = RGB(220, 220, 220) Then

    With zelle
      .Borders.LineStyle = xlDot
      .Font.Bold = True
    End With

  End If

Next zelle
```

Farben setzen

- Mit Hilfe der Funktion RGB(red, blue, green) können Farben für Schriften, Rahmen und Zellhintergründe gesetzt werden.
- Andere Möglichkeiten:
  - Für die Standardfarben schwarz (vbBlack), weiß (vbWhite), rot (vbRed), grün (vbGreen), gelb (vbYellow), blau (vbBlue), blaugrün (vbCyan) und magenta (vbMagenta) sind Farbkonstanten vordefiniert.
  - Der Eigenschaft ColorIndex wird ein Wert von 1 bis 56 übergeben. Der Wert beschreibt exakt eine Farbe der Farbpalette der Arbeitsmappe.
- Mit Hilfe von .Color = xlNone kann eine Farbe zurückgesetzt werden.

RGB-Farben

- ... werden für die Darstellung von Farben am Bildschirm genutzt.
- Das RGB-Farbsystem addiert (mischt) Licht in verschiedenen Farben.
- Jede Farbe (Rot, Grün, Blau) wird in 256 Helligkeitsstufen unterteilt. Zum Beispiel:
  - RGB(255, 255, 255) stellt die Farbe weiß dar.
  - RGB(0, 0, 0) stellt die Farbe schwarz dar.
  - RGB(255, 255, 0) stellt die Farbe gelb dar.
- Um so mehr sich eine Farbe Weiß annährt, um so heller wird sie.

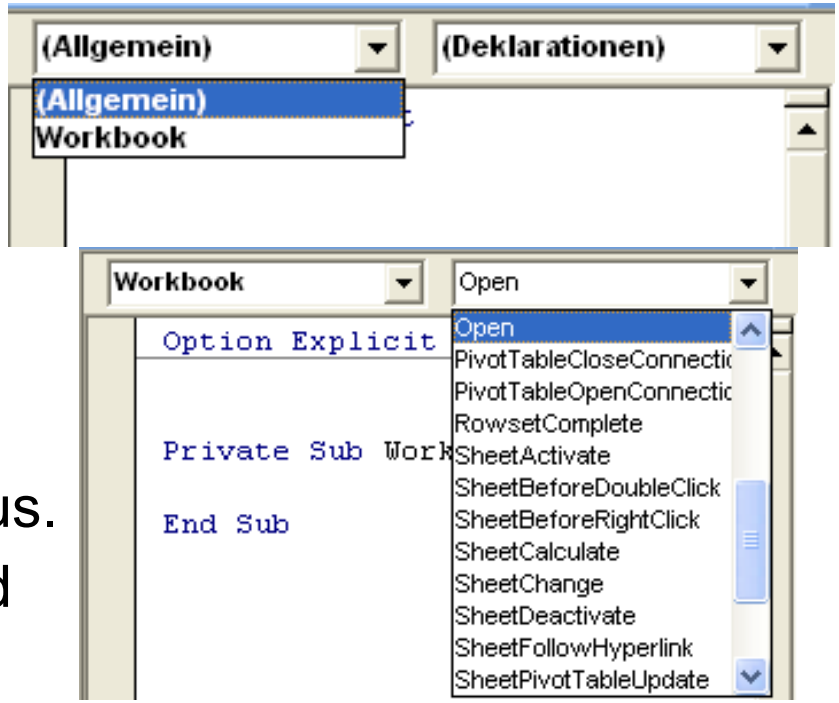


Ereignis (Event)

- ... tritt immer in Verbindung mit einem Objekt auf.
- ... wird durch die Maus oder die Tastatur ausgelöst.
- ... ist eine Reaktion auf eine Benutzeraktion.
- ... ist eine von einem Objekt gesendete Nachricht.
- Beispiele für Ereignisse:
  - Die Größe des Fensters wird verändert.
  - Die Arbeitsmappe wird geöffnet.
  - Ein Arbeitsblatt wird aktiviert.
  - Der Inhalt einer Zelle wird verändert.

Ereignisse auswählen

- Klicken Sie doppelt auf ein Objekt im Projekt-Explorer. Zum Beispiel: *Diese Arbeitsmappe*. Das dazugehörige Modul wird im Codefenster angezeigt.
- Wählen Sie aus dem Kombinationsfeld Objekt am oberen Rand das passende Objekt aus. Hier: Workbook.
- Anschließend wählen Sie ein Ereignis aus dem Kombinationsfeld Prozeduren aus.
- Das Gerüst des Ereignisses wird automatisch eingefügt.



Ereignisprozeduren (Event-Handler)

Private Sub Workbook_Open()

- ... fangen ein Ereignis ab.
- ... reagieren mit Hilfe von Anweisungen auf das ausgelöste Ereignis.
- ... werden häufig nicht implementiert. Ein Ereignis wird zum Beispiel für
  - ... die Fehlerkorrektur bei der Eingabe der Daten abgefangen.
  - ... die automatisierte Formatierung von Zellen abgefangen.
  - ... die automatisierte Aktivierung eines Arbeitsblattes abgefangen.

Informationen im Namen des Event-Handlers

Private Sub Workbook_Open()

- Der Name eines Event-Handlers besteht aus
  - ... dem Objekt, welches das Ereignis ausgelöst hat.
  - ... dem Ereignis, welches von der Maus oder Tastatur ausgelöst wurde.
  - Objekt und Ereignis werden durch ein Unterstrich getrennt.

Aufbau des Event-Handlers

```
Private Sub Workbook_Open()
End Sub
```

- Ein Event-Handler
  - ... beginnt mit Sub und endet mit End Sub.
  - ... ist eine Prozedur, die keinen Wert zurück gibt.
- Als Zugriffsmodifizierer wird immer Private angegeben. Die Prozedur ist nur innerhalb des Moduls bekannt, in der die Prozedur definiert ist. Die Prozedur ist an ein Objekt gebunden.
- In Abhängigkeit des Ereignisses können der Prozedur Werte übergeben werden.

Ereignisse für die Arbeitsmappe (Workbook)

- Activate. Die Arbeitsmappe wird aktiviert. Die Arbeitsmappe liegt im Vordergrund.
- Deactivate. Eine andere Arbeitsmappe wird aktiviert. Die Arbeitsmappe wird in den Hintergrund verschoben.
- BeforeClose. Bevor die Arbeitsmappe geschlossen wird.
- BeforeSave. Bevor die Arbeitsmappe gesichert wird.
- NewSheet. Ein neues Arbeitsblatt wird eingefügt.
- Open. Die Arbeitsmappe wird geöffnet.
- ... und viele andere.

Die Arbeitsmappe wird geöffnet

```
Private Sub Workbook_Open()
    Dim tabelle As Worksheet
    Dim zelle As Range

    Set tabelle = ActiveWorkbook.Worksheets("Deckblatt")
    Set zelle = tabelle.Range("B1")
    zelle.Value = Application.UserName

    Set zelle = tabelle.Range("B2")
    If zelle.Value = "" Then
        zelle.Value = Date
    End If
End Sub
```

Welcher Benutzer ist angemeldet?

Die Arbeitsmappe wird geschlossen

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)

    Dim button As VbMsgBoxResult

    If ActiveWorkbook.Saved = False Then
       button = MsgBox("Möchten Sie die Änderungen speichern?", _
                                  vbYesNo, "Speicherung")

       If button = vbYes Then
          Call setAenderungTime
          ActiveWorkbook.Save
       End If
    End If
End Sub
```

Ereignisse für das Arbeitsblätter (Worksheet)

- Activate. Das Arbeitsblatt wird aktiviert. Das Arbeitsblatt liegt im Vordergrund.
- Deactivate. Eine anderes Arbeitsblatt wird aktiviert. Das Arbeitsblatt wird in den Hintergrund verschoben.
- Calculate. Das Arbeitsblatt wird neu berechnet.
- Change. Die Zellen des Arbeitsblatts haben sich verändert.
- SelectionChange. Die Auswahl auf dem Arbeitsblatt hat sich verändert.

Die Zelle wird verlassen...

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Dim tmpPLZ As String

    If Target.Column = 1 Then
        Target.Font.Color = vbBlack
        tmpPLZ = Target.Value

        If (Len(tmpPLZ) > 5)  or (Len(tmpPLZ) < 5) Then
            MsgBox ("Die Postleitzahl hat n Zeichen zu wenig")
        ElseIf IsNumeric(tmpPLZ) = False Then
            MsgBox ("Die Postleitzahl besteht nur aus Zahlen von 0 bis 9")
        End If
    End If
End Sub
```

Welche Zelle wurde verlassen?

Das Arbeitsblatt wird aktiviert...

```
Private Sub Worksheet_Activate()
    Dim ziel As Range

    With ActiveCell
        .BorderAround Color:=RGB(20, 20, 20), Weight:=xlThin
        .Interior.Color = RGB(220, 220, 220)
        .Font.Size = 28
        .Font.Name = "Arial"
        .RowHeight = 30
        .Value = "Januar"
        Set ziel = Worksheets("Wochentag").Range("A1:L1")
        .AutoFill Destination:=ziel
        ziel.Columns.AutoFit
    End With
End Sub
```

Die Zellen werden automatisch gefüllt und angepasst.

Diagramme

- ... visualisieren Werte in Abhängigkeit von bestimmten Kategorien.
- ... stellen grafisch Beziehungen zwischen verschiedenen Daten dar.
- ... stellen Abhängigkeiten zwischen Daten in kompakter und übersichtlicher Form dar.
- Zusammenhänge zwischen zwei oder mehr abhängigen Werten werden veranschaulicht.
- Durch Verzerrungen der Größenverhältnisse, Farben etc. kann die visuelle Darstellung manipuliert und eine bestimmte Aussage erreicht werden.
- ... werden mit Hilfe von Microsoft Graph erstellt.

Diagrammblätter erstellen

```
Sub newDiagramm()
    Dim datenquelle As Worksheet

    Set datenquelle = Worksheets("KlimaHannover")
    ThisWorkbook.Charts.Add After:=datenquelle



    With ActiveChart
        .ChartType = xlLineMarkers
        .SetSourceData datenquelle.Range("A3:M4")
        .Name = "Niederschläge"
    End With
End Sub
```

Mit Hilfe von Add wird der Auflistung Charts ein neues Diagrammblatt hinzugefügt.

Eigenschaften des Diagramms

- Name enthält die Bezeichnung für das Diagrammblatt.
- SetSourceData legt die Datenquelle für das Diagramm fest. Standardmäßig werden die Daten nach Spalten (xlColumns) dargestellt. Als zweiter Parameter kann der Methode aber auch der Wert xlRows übergeben werden, um die Daten in Abhängigkeit der Zeilen darzustellen.

Diagrammarten

- ChartType legt den Typ des Diagramms fest.
- Liniendiagramme (xlLine...) bieten die Möglichkeit zeitabhängige Entwicklungen darzustellen.
- Balkendiagramme (xlBar...) oder Säulendiagramme (xlColumn...) werden genutzt, um absolute Zahlen in Beziehung zu setzen. Es können abhängige Werte miteinander verglichen werden.
- Kreis- oder Tortendiagramme (xlPie...) stellen prozentuale Anteile an einem Gesamtwert dar.

Diagrammfläche, Zeichnungsfläche und Titel

```
With ActiveChart"

    ' Farbe der Diagramfläche
    .ChartArea.Interior.Color = vbWhite

    ' Farbe der Zeichnungsfläche
    .PlotArea.Interior.Color = RGB(220, 220, 220)

    ' Wird ein Titel angezeigt? Wenn ja, Text für den Titel
    .HasTitle = True
    .ChartTitle.Text = "Niederschläge (mm) pro Monat in Hannover"
End With
```

Legende

```
With ActiveChart"

    .HasLegend = True

    With .Legend
        .Position = xlLegendPositionBottom

        .Interior.Color = RGB(220, 220, 220)
        .Border.Color = vbBlack
        .Border.Weight = xlThick
    End With
End With
```

Die Legende kann mit Hilfe von .Top und .Left positioniert werden. Hier wird die Legende mit Hilfe von vordefinierten Konstanten positioniert.

Die Innenfläche sowie die Rahmen können definiert werden.

Achsen

```
With ActiveChart"

  With .Axes(xlCategory)
      .HasTitle = True
      .AxisTitle.Text = "Monat"
   End With

   With .Axes(xlValue)
      .HasTitle = True
      .AxisTitle.Text = "Niederschlag mm"
      .TickLabels.NumberFormatLocal = "00"
      .MinimumScale = 0
      .MaximumScale = 20
   End With
End With
```

Die horizontale Achse wird formatiert.

Die vertikale Achse wird formatiert.

Datenreihe

```
With ActiveChart"

    With .SeriesCollection(1)

        .Border.Color = vbRed

        .MarkerStyle = xlMarkerStyleDot
        .MarkerForegroundColor = vbBlue
        .MarkerBackgroundColor = vbBlue

    End With

End With
```

Die Datenreihen werden von 1 bis n durchnummeriert.

Der Rahmenfarbe oder bei einem Liniendiagramm die Linienfarbe werden definiert.

Das Aussehen der Markierungspunkte definieren.

Datenpunkte

```
For count = 1 To .SeriesCollection(1).Points.Count

    With .SeriesCollection(1).Points(count)
      .Border.Color = vbBlue
      .MarkerStyle = xlMarkerStyleSquare
      .MarkerForegroundColor = vbBlue
      .MarkerBackgroundColor = vbBlue

      .ApplyDataLabels xlShowValue
    End With

  Next count
```

Auflistung aller Punkte einer Datenreihe. Index von 1 bis n für die Datenpunkte.

Werden die Werte angezeigt?

Eingebettetes Diagramm

```
Sub newDiagrammInSheet()
    Dim datenquelle As Worksheet
    Dim objChart As ChartObject
    Dim picChart As Chart

    Set datenquelle = Worksheets("KlimaHannover")
    Set objChart = datenquelle.ChartObjects.Add(10, 100, 400, 250)
    Set picChart = objChart.Chart

    With picChart
        .ChartType = xlLineMarkers
        .SetSourceData datenquelle.Range("A3:M4")
    End With
End Sub
```

ChartObjects und Chart

- ChartObjects
  - ... ist eine Auflistung aller Objektrahmen auf einem Arbeitsblatt.
  - ... befinden sich in der Objekthierarchie unterhalb des Objekts Worksheet.
  - [Worksheet].ChartObjects.Add([Left], [Top], [Breite], [Hoehe]) erzeugt einen neuen Objektrahmen.
- Die Eigenschaft Chart
  - ... verweist auf das Diagramm in dem angegebenen Rahmen.
  - ... beschreibt das eingebettete Diagramm