

Grundkurs für Excel – VBA – Part III

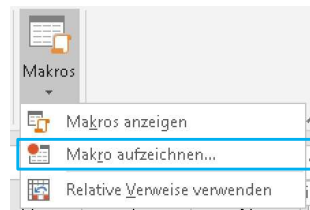
Nico Ludwig

Themen

- Makros aufzeichnen
- Formulare
- Wichtige Steuerelemente
- Steuerelementeigenschaften und -ereignisse

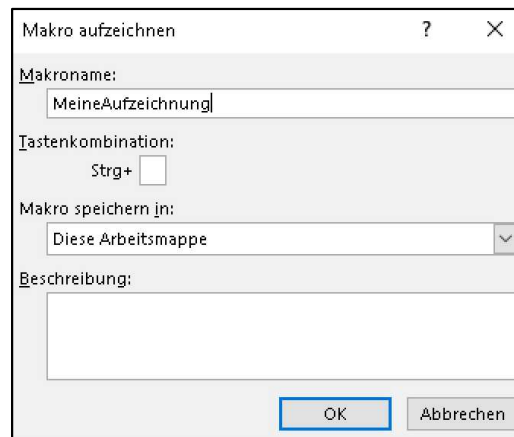
Makros aufzeichnen – Teil I

- Bisher haben wir VBA immer von Hand programmiert, um Probleme zu lösen.
- Darüber hinaus bietet uns Excel auch die Möglichkeit Makros interaktiv aufzuzeichnen.
 - Das funktioniert ähnlich wie eine Tastaturmakroaufzeichnung an einer Gamer-Tastatur.
- Im Modus "Makro aufzeichnen", wird VBA-Code für jede Aktion generiert, die wir interaktiv ausführen, bis die Aufzeichnung gestoppt wird.
- Die Aufzeichnung eines neuen Makros starten wir im Ribbon "Ansicht":



Makros aufzeichnen – Teil II

- Im erscheinenden Dialog geben wir den Namen der Prozedur an, die den erzeugten Code enthalten soll:



- Die neue Prozedur wird also *MeineAufzeichnung()* heißen.
- Nach Klick auf OK startet die Makroaufzeichnung sofort.
 - Excel zeigt in der Statuszeile, dass eine Aufzeichnung läuft, in dem ein Stopp-Symbol angezeigt wird:



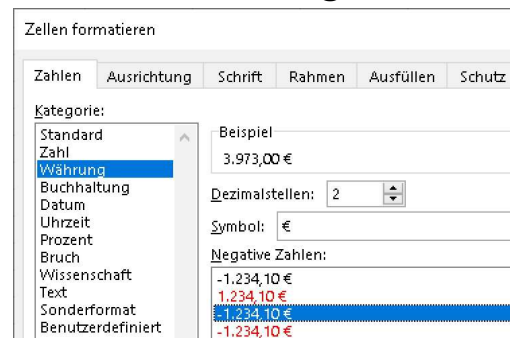
- Dieses Symbol kann dann angeklickt werden, um die Makroaufzeichnung zu stoppen.

Makros aufzeichnen – Teil III

- Wir führen jetzt zwei interaktive Aktionen aus, die automatisch mit aufgezeichnet werden.
- (1) Wir geben den Wert 3973 in C3 ein:

	A	B	C
1			
2			
3			3973

- (2) Wir formatieren C3 als Währungswert:



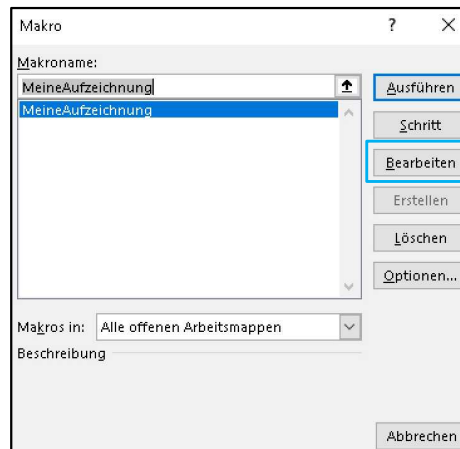
	A	B	C
1			
2			
3			3.973,00 €

- Wir stoppen die Makroaufzeichnung, indem wir in der Statusleiste den Stopp-Button klicken:



Makros aufzeichnen – Teil IV

- Nun können wir uns das aufgezeichnete Makro, bzw. Prozedur *MeineAufzeichnung()* anschauen:




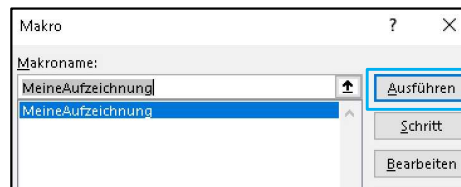
- Der Code sieht etwas merkwürdig aus, aber im Kern ist er durchaus verständlich:

- Neu für uns: *ActiveCell* repräsentiert die gerade ausgewählte Zelle.
- Das Property *FormulaR1C1* erlaubt uns einem Bereich eine Formel zuzuordnen, hier ist die Formale ja nur der feste Wert 3973.
 - Dieses Property verlangt die Angabe von Formeln in der relativen Z1S1 (Zeile1Spalte1, engl. R1C1 für Row1Column1) Schreibweise. Das haben wir nicht besprochen, aber damit kann man Excel Zellbezüge relativ zur aktuellen Zellen angeben und die Schreibweise wird nicht oft benutzt.
- *Selection* ist ein Objekt, dass die aktuell selektierten Zellen repräsentiert.
- Das Property *NumberFormat* erlaubt die Angabe einer Wertformatierung für Zahlen.

```
(Allgemein)
Sub MeineAufzeichnung()
    '
    '   MeineAufzeichnung Makro
    '
    ActiveCell.FormulaR1C1 = "3973"
    Range("C3").Select
    Selection.NumberFormat = "$#,##0.00_);(,$#,##0.00)"
End Sub
```

Makros aufzeichnen – Teil V

- Aber es ist ja noch zu beweisen, die Aufzeichnung auch ausgeführt werden kann!
- Wir wählen zunächst C3 aus und löschen den Wert und die Formatierung mit Klick auf 
 - Dann sollte die Tabelle soweit wieder völlig leer sein.
- Dann führen wir das aufgezeichnete Makro aus:



- Was zum erwarteten Ergebnis führt:

	A	B	C
1			
2			
3			3.973,00 €

- Die Makroaufzeichnung kann guten Basiscode für eigenen Makros liefern.

Einführung: Formular-basierte Makros – Teil I

- Bisher haben wir nur Makros besprochen, die mit Excel-Tabellen arbeiten.
- Daneben können wir mit Excel auch Formular-basierte Makros erstellen.

Einführung: Formular-basierte Makros – Teil II

- Ein Formular ist hierbei einfach eine Benutzeroberfläche, die einem Papierformular nachgebildet ist.
- Auf einem Papierformular finden wir Beschriftungen, Einrahmungen, Textfelder, "Ankreuzboxen" usw.
- In Excel können wir tatsächlich Formulare mit solchen Elementen erstellen.

Dienstnummer:	<input type="text"/>	Vorname:	<input type="text"/>	Nachname:	<input type="text"/>
Datum:	<input type="text"/>	Vorgangsnr.:	<input type="text"/>	Meldung ergangen:	Ja <input type="checkbox"/> Nein <input type="checkbox"/> mt <input type="checkbox"/>
Vorgangsbeschreibung: <div style="border: 1px solid black; height: 150px; width: 100%;"></div>					
Unterschrift					

X

Dienstnummer: Vorname: Nachname:

Datum: Vorgangsnr.: Meldung eingegangen: ☐ Ja ☐ Nein ☐ mt

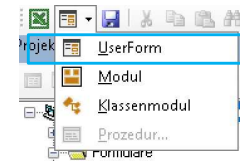
Vorgangsbeschreibung:

Einführung: Formular-basierte Makros – Teil III

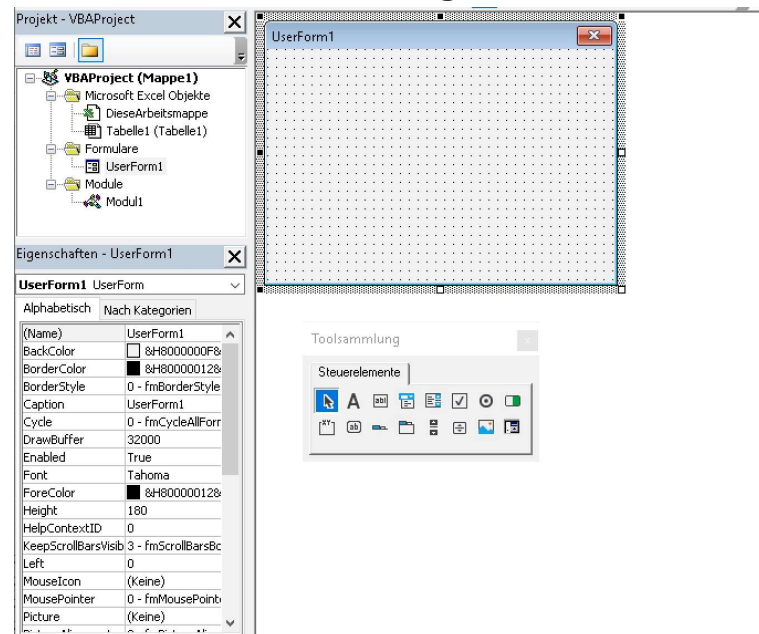
- Welchen Sinn hat es denn Formulare in Excel "nachzubauen"?
- Naja, was ist denn überhaupt der Sinn eines Formulars? Z.B. bei einer Steuererklärung.
- Formulare helfen uns alle benötigten Daten für einen bestimmten Sachverhalt zu erfassen.
 - Die deutliche Beschriftung von Feldern sagt uns, wo wir welche Daten hinschreiben sollen.
 - Die Gestaltung der Felder sagt uns, welcher Art die Daten sind, z.B. eine Ankreuzbox für bestimmte Optionen.
- Mit Formularen machen wir einfach viel weniger Fehler bei der Datenerfassung!
- Im Englischen wird ein Formular als "Form" bezeichnet.

Ein neues Formular erstellen

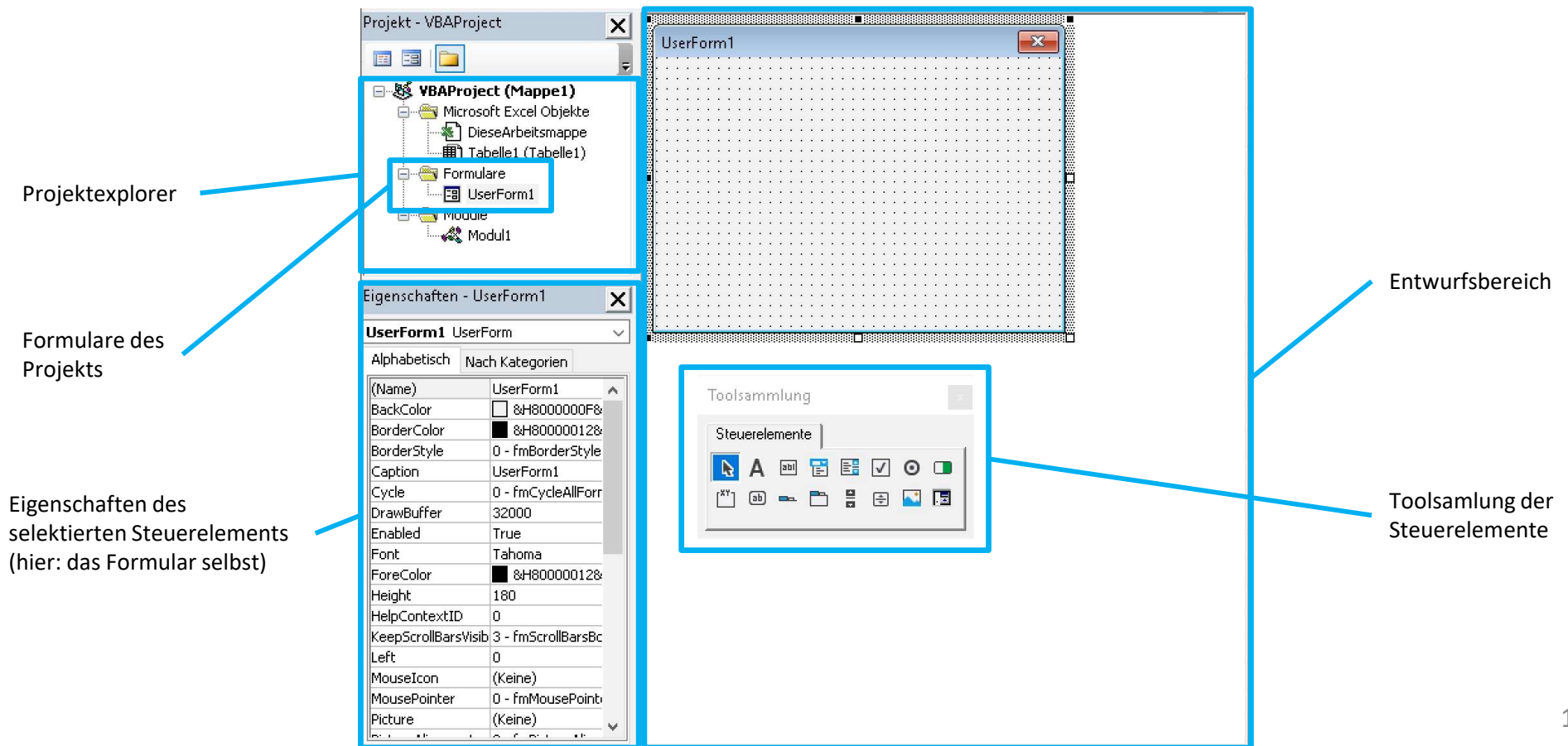
- Wir erstellen in Excel neue Formulare im VBA-Editor:



- Darauf hin öffnet sich der VBA-Formular-Designer:

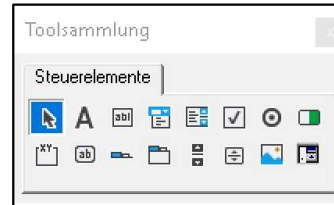


Bestandteile des Formular-Designers



Die wichtigsten Steuerelemente – Teil I


- Zunächst schauen wir uns die Elemente an, die wir auf einem Excel-Formular platzieren können.
 - Die finden wir eben auf dem kleinen Fenster "Toolsammlung":



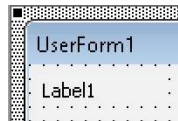
- Ein Steuerelement ist das kleinste Element, mit dem wir auf einer Benutzeroberfläche interagieren können.
 - Steuerelemente (engl. Controls) erlauben dem Benutzer einfach ein Programm über die Benutzeroberfläche steuern.
- Beispiele für Steuerelemente: Schaltflächen, Beschriftungen, Textfelder, Listenfelder, Kontrollkästchen usw.
- Das heißt wir können mit Excels Formularen eigene Benutzeroberflächen, bzw. Dialoge zusammenbauen!


Die wichtigsten Steuerelemente – Teil II

- Wir werden uns in diesem Teil des Kurses mit folgenden Steuerelementen beschäftigen:

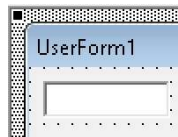
- Die Beschriftung (engl. Label), Symbol: 

- Standardverhalten: das Label ist "passiv" und zeigt einen beschreibenden Text an.
- So sieht ein Label aus:



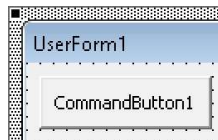
- Das Textfeld (engl. Textbox), Symbol: 

- Eine Textbox nimmt vom Benutzer eingegebenen Text auf.
- So sieht eine Textbox aus:



- Die Schaltfläche (engl. CommandButton (kurz "Button")), Symbol: 

- Standardverhalten: der Button löst eine Aktion aus.
- So sieht ein Button aus:

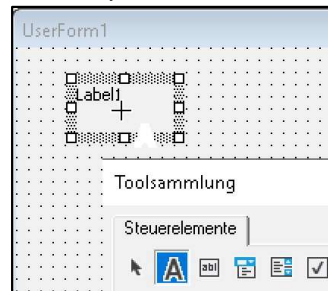


Die wichtigsten Steuerelemente – Teil III

- Die Aufgaben der Steuerelemente klären in gewisser Art auch ihre Anwendung in Makros:
 - Labels erklären welche Eingaben erwartet werden. – Sie beschriften andere Steuerelemente.
 - Textboxen nehmen Benutzereingaben entgegen und zeigen Ausgaben eines Makros.
 - Buttons lösen Aktionen aus.
- Das entspricht den Konzepten in der Programmierung, die wir schon kennen:
 - Labels ersetzen gewissermaßen die Prompts (z.B. *MsgBox()*, teilweise auch *InputBox()*).
 - Eigentlich kann das Formular, da es ja ein Dialog ist, selbst eine Art Prompt darstellen.
 - Textboxen ersetzen Zellen als Mittel zur Dateneingabe- und Ausgabe.
 - Buttons lösen die Ausführung einer Prozedur aus.
- Wir entwickeln jetzt ein einfaches Formular, dass diese Idee veranschaulicht.

Ein Formular entwerfen

- Das Beispielformular, bzw. das Makro dahinter, arbeitet ganz einfach: es berechnet die Summe zweier Zahlen.
- Bevor wir loslegen, müssen wir noch verstehen, wie man Formulare/Dialoge im Formular designer erstellt.



- Der Designer folgt dem Prinzip eines einfachen Zeichenprogramms.
- Wir wählen Steuerelemente, die wir auf den Dialog setzen wollen in der Toolsammlung und zeichnen sie auf den Dialog.
 - Alternativ geht das auch per Drag'n'Drop aus der Toolsammlung.
- Nachdem Steuerelemente auf dem Dialog platziert sind können wir ihre Position und Größe mit der Maus verändern.
- Die Anordnung der Steuerelemente per Maus nennt man manchmal "visuelles Design".
 - Aus "Marketinggründen" kam auch der Name "Visual Basic for Applications" zustande.

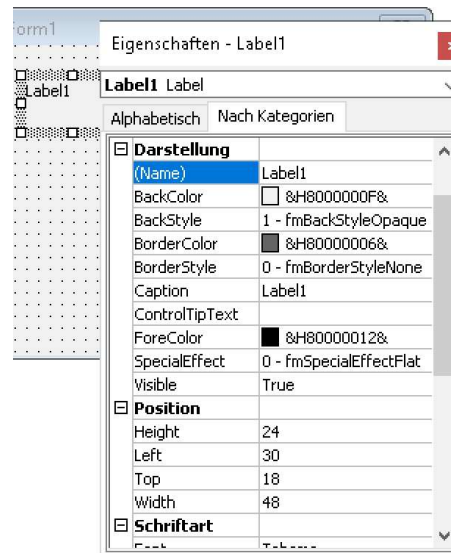
Eigenschaften eines Steuerelements

- Das Konzept der Eigenschaften haben wir schon bei den Zelleigenschaften besprochen.
- Wir wissen auch, dass Eigenschaften die Eigenschaften eines Objekts repräsentieren.
- Objekte sind hierbei Zellen und Bereiche, aber auch Formulare und Steuerelemente!
- Daher haben z.B. auch Steuerelemente Eigenschaften!
 - Konsequenterweise können wir Steuerelementeigenschaften visuell beeinflussen.
- Die Steuerelementeigenschaften können im Eigenschaftsfenster bearbeitet werden.

Das Eigenschaftsfenster

- Im Eigenschaftsfenster sehen wir immer die Eigenschaften des selektierten Steuerelements.
 - Z.B. die Eigenschaften eines Labels:

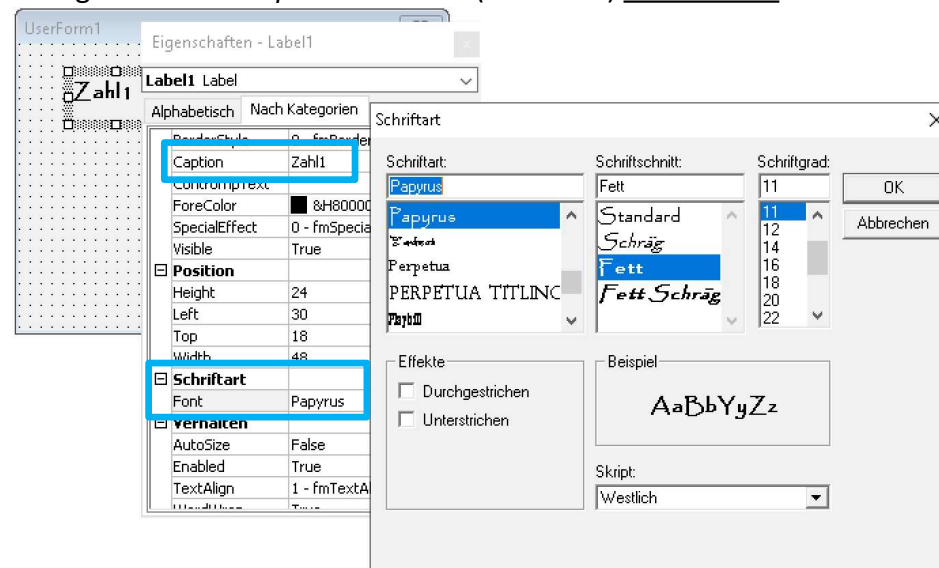
- Manche Eigenschaften sind nachvollziehbar: *Position, Schriftart*
- Andere Eigenschaften sind eher "unklar": *Caption, ControlTipText*.



- Die wichtigste Eigenschaft für Labels ist *Caption*. Mit *Caption* legen wir den Beschriftungstext fest.

Visuelles Arbeiten mit Eigenschaften

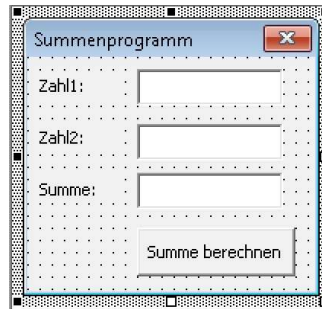
- Änderungen von Eigenschaften im Eigenschaftsfenster ändern i.d.R. sofort die Darstellung im Designer.
 - Z.B. sind Änderungen an den Eigenschaften *Caption* und *Font* (Schriftart) unmittelbar auf dem Label wirksam:




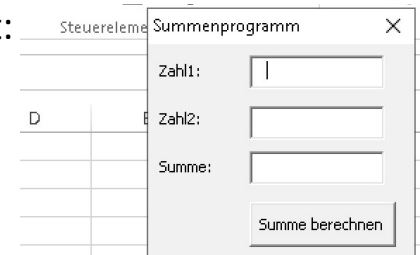
- Das unmittelbare visuelle "Feedback" bearbeiteter Eigenschaften ist ein Eckpfeiler des visuellen Designs.
 - Die Programmentwicklung mittels visuellem Design wird manchmal "Rapid Application Development" (RAD) genannt.
 - "Rapid Application Development" = "Schnelle Programmentwicklung" ist auch ein Marketingbegriff für VBA.
 - (RAD ist auch ein wesentliches Marketingargument für die professionelle Variante von VBA, nämlich "Visual Basic" (ohne "A").)¹⁹

Weiter mit dem Entwurf

- Jetzt haben wir genug Wissen, um unser Summenprogramm zu entwickeln.
- Wir zeichnen das Formular mit dem Designer:

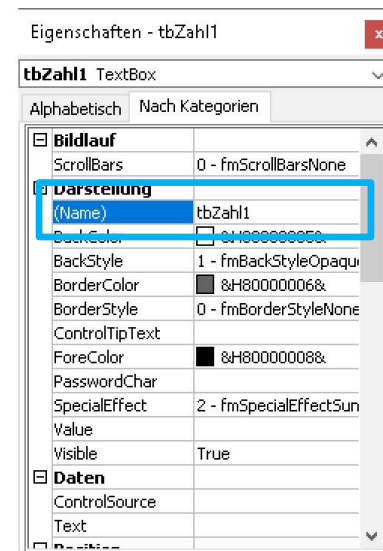
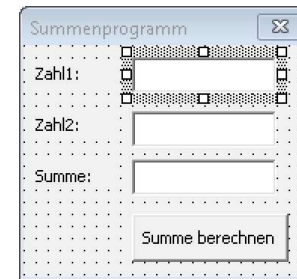


- Nun selektieren wir das Formular im Designer und klicken auf die Schaltfläche  .
 - Nun wird das Makro ausgeführt und das Formular erscheint:
- Um das Makro zu beenden schließen wir einfach das Formular.



Steuerelemente benennen

- Unser Formular macht noch gar nichts, ein Klick auf "Summe berechnen" bewirkt nichts.
- Bevor wir dem Formular Leben einhauchen, müssen wir nochmal an die Steuerelemente ran.
- Wir müssen den Steuerelementen noch sprechende Namen geben:
 - Dazu setzen wir die Spezialeigenschaft Name.
 - *Name* bestimmt den Bezeichner des Objekts, also auch der Variablen, das das Steuerelement im VBA-Code repräsentiert.
 - In VBA ist es üblich Bezeichnen von Steuerelementen Präfixe zu geben, um im Namen den Objekttyp zu erkennen.
 - Übliche Präfixe: Textbox: *tb*, Button: *btn*, Label: *lbl*, usw.
 - Wir benennen die Textboxen mit *tbZahl1*, *tbZahl2* und *tbSumme*.
 - Der Button erhält den Namen *btnSumme*.
 - Beachte: der Steuerelementname hat nichts mit seiner Beschriftung, also der Eigenschaft *Caption*, zu tun!



Das Verhalten des Formulars festlegen

- Das Formular soll folgendes Verhalten zeigen:
 - Es nimmt vom Benutzer zwei Dezimalzahlen aus den Textboxen *tbZahl1* und *tbZahl2* auf.
 - Bei Klick auf die Schaltfläche *btnSumme* wird deren Summe berechnet.
 - Die berechnete Summe wird in die Textbox *tbSumme* geschrieben.
- Die Textboxen brauchen keine weitere Vorbereitung: Texteingabe und -ausgabe beherrschen sie bereits.
 - Also Textboxen besitzen bereits grundlegende interaktive Funktionen.
- Anders sieht es bei der Schaltfläche *btnSumme* aus: deren Funktionalität müssen wir programmieren.
 - Schaltflächen zeigen immerhin einen grafischen Effekt beim Klicken, indem sie das "Eindrücken" der Schaltfläche simulieren.
 - Wir wollen aber, dass die Schaltfläche eine zusätzliche Aktion ausführt.
- VBAs Schaltflächen heißen ja *CommandButtons*: wir müssen für *btnSumme* ein Kommando angeben.

Ein Kommando für die Schaltfläche

- Wenn wir in VBA von einem Kommando reden, ist eigentlich eine Prozedur gemeint.
- Das heißt wir müssen eine Prozedur mit der Schaltfläche verbinden.
 - Um das zu erreichen führen wir im Designer einen Doppelklick auf die Schaltfläche *btnSumme*.
 - Es erscheint der VBA Editor mit der neuen Prozedur *btnSumme_Click()*:



- Wir müssen nun in der Prozedur *btnSumme_Click()* die Berechnung durchzuführen.
- Bevor wir das machen besprechen wir noch das Konzept der Ereignisse.

Ereignisse – Teil I

- Auf technischer Ebene weisen wir eine Prozedur dem Klickereignis der Schaltfläche zu.
- Ein Ereignis wird vom Windows Betriebssystem für jede Interaktion mit einem Steuerelement ausgelöst.
- Für die Behandlung des Klickereignisses wurde eine Prozedur mit einem Namensschema erstellt:

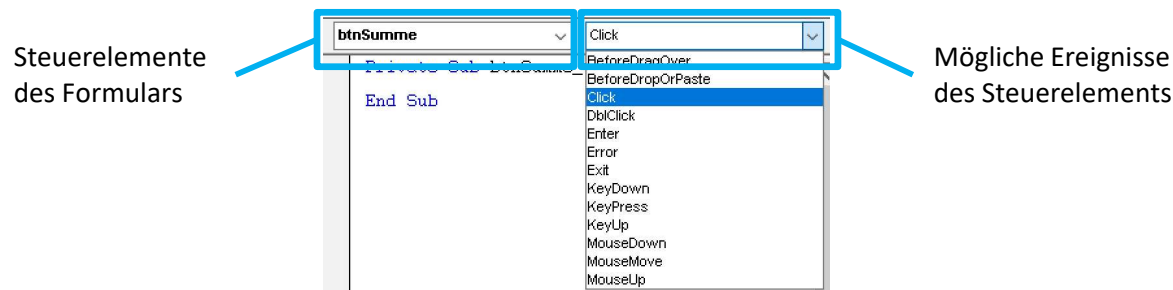


Das Schlüsselwort **Private** gibt an, dass diese Prozedur quasi nur innerhalb des Formulars "lokal" zugreifbar ist, sie ist "privat".

- Das Namensschema lautet einfach <Steuerelementbezeichner_Ereignisname>.
- Noch ein paar Begriffe aus der Programmierung:
 - Im Englischen und auch im Deutschen werden Ereignisse oft Events genannt.
 - Prozeduren, die Ereignisse behandeln werden auch im Deutschen oft als Handler (engl. für "Handhaber"/Behandler) bezeichnet.
 - Zusammengefasst heißen die Prozeduren dann auch Eventhandler.

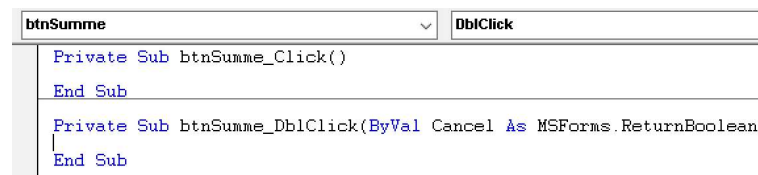
Ereignisse – Teil II

- Außer Klickereignisse können Schaltflächen noch andere Ereignisse "empfangen", z.B. den Doppelklick.
 - Wir können uns die verfügbaren Ereignisse eines Steuerelements im VBA-Editor anschauen und Prozeduren dafür hinterlegen.
 - Dazu müssen wir oben im Editor das entsprechende Steuerelement und dessen Ereignis auswählen:



Beachte: Wenn ein Steuerelementbezeichner nachträglich umbenannt werden, müssen auch die Eventhandler umbenannt werden! – Ansonsten werden sie einfach nicht aufgerufen, wenn ein Event eintritt.

- Wenn ein Ereignis ausgewählt, für das es noch keine Prozedur gibt, wird sie sofort angelegt und selektiert.
 - Also z.B. für den Doppelklick (*DbClick*-Event):



- Ansonsten selektiert der Editor in die schon vorhandene Prozedur.
- Wie wir am Doppelklick-Handler sehen, können Eventhandler auch Parameter haben.

Der Eventhandler – Teil I

- Hier wird nun eine kurze Variante des Summen-Eventhandlers besprochen.
 - Aus Gründen der Übersichtlichkeit programmieren wir die ganze EVA in einer Prozedur:

```
Private Sub btnSumme_Click()  
    Dim zahl1, zahl2, summe As Double ' (1)  
    If IsNumeric(tbZahl1.Value) And IsNumeric(tbZahl2.Value) Then ' (2)  
        zahl1 = CDb1(tbZahl1.Value) ' (3)  
        zahl2 = CDb1(tbZahl2.Value) ' (4)  
        summe = zahl1 + zahl2  
        tbSumme.Value = summe ' (5)  
    Else  
        MsgBox("Sie müssen zwei Zahlen eingeben!") ' (6)  
    End If  
End Sub
```

- In (2) prüfen wird erst die Inhalte der Textboxen *zahl1* und *zahl2*.
 - Das machen wir mit dem bekannten Prädikat *IsNumeric()* und verbinden die Bedingungen mit *And*.
- Neu in (2) ist, dass wir die Inhalte der Textboxen mit deren Eigenschaft *Value* abrufen.

Der Eventhandler – Teil II

```
Private Sub btnSumme_Click()  
    Dim zahl1, zahl2, summe As Double  
    If IsNumeric(tbZahl1.Value) And IsNumeric(tbZahl2.Value) Then ' (1)  
        zahl1 = CDbI(tbZahl1.Value) ' (2)  
        zahl2 = CDbI(tbZahl2.Value) ' (3)  
        summe = zahl1 + zahl2  
        tbSumme.Value = summe ' (4)  
    Else  
        MsgBox("Sie müssen zwei Zahlen eingeben!")  
    End If  
End Sub
```

- In (2) – (3) werden die Inhalte der Textboxen in **Doubles** gewandelt.
 - Das ist nötig, denn in Textboxen wird nur Text abgelegt (daher auch ihr Name).
 - Auch eine (Dezimal-)Zahl steht als Text in der Textbox, wir müssen sie mit **CDbI()** extrahieren.
- In (5) wird der Eigenschaft *tbSumme.Value* einfach die berechnete Summe zugewiesen.

Formular – nicht besprochene Themen

- Hauptformular und Unterformulare
- Andere Steuerelemente wie Kombinationsfeld, Listenfeld, Radiobox