Handbuch Sprachreferenz für den Generator

ShakeHands Kontor
ShakeHands Conto
ShakeHands Faktura
ShakeHands Budget

Inhalt

Impressum

Sprachreferenz 2010
Ausgabe 7.1

Kontakt

ShakeHands Software Ltd
Sägerei Kröschenbrunnen
CH - 3555 Trubschachen
Telefon: 0878 87 47 77
Fax: 034 495 70 25

Kapitel 1 Einführung

Über dieses Handbuch

Mit der Ihnen vorliegenden Version eines unserer Programme aus der **ShakeHands Office-Reihe** verfügen Sie über eine leistungsfähige, komfortable und aktuelle Lösung für Ihre Auftragsverwaltung und/oder Buchhaltung. Dieses Handbuch erläutert das Erstellen und Anpassen von Druckformularen mit Hilfe der integrierten Formular-Beschreibungssprache.

Wir setzen voraus, dass der Benutzer gewisse Grundkenntnisse im Umgang mit Programmiersprachen und dem Lesen und Schreiben von Quelltexten mitbringt. Aber auch für Benutzer, die sich nicht mit Beschreibungssprachen auskennen, sollte es möglich sein mit Hilfe dieses Handbuches Formulare zu erstellen und anzupassen.

Die folgenden Programme gehören derzeit zur ShakeHands-Produktlinie:

ShakeHands Conto 2010
Doppelte Finanzbuchhaltungen mit Anlagenverwaltung Saldo ist die robuste, klare und einfache Finanzbuchhaltung. ShakeHands Balance mit Fremdwährungen, Debitoren- und Kreditorensystem, sowie offener Postenverwaltung für die Anspruchsvolle Buchhaltung.

ShakeHands Faktura 2010
Auftragsverwaltung mit Kassenbuch in der Ausführung Balance mit Lagerverwaltung und Bestandesrechnung.

ShakeHands Kontor 2010
Auftragsverwaltung und Buchhaltung (Conto + Faktura kombiniert) im kaufmännischen Komplettpaket Inklusive Artikelverwaltung und Dienstleistungsstamm, Mahnungen und Verkaufsstatistik., inkl. ESR-Verfahren und neu mit Aboverwaltung und Lager.

ShakeHands Budget 2010
kostenlos Kassenbuchhaltung mit einfachen Auswertungen und Budgetverwaltung (Freeware).

ShakeHands Netzwerkversion

Die Versionen können nach Bedarf alle mit dem Real Server verwendet werden, damit arbeiten Sie gleichzeitig an Ihrer Datenbank oder via Remote von beliebigen Punkten aus.

Alle Ausführungen sind funktionsgleich sowohl für Mac OS X 10.4, 10.5 und 10.6 als auch für Windows XP, Vista und 7 erhältlich. Detaillierte tabellarische Funktionsvergleiche finden Sie auf der Webseite von ShakeHands Software Ltd unter www.shakehands.com.

Darüber hinaus ist die mehrsprachige Version (Teile der französischen und englischen Version sind schon verfügbar) in Planung.

In diesem Handbuch wird im Allgemeinen nur der Begriff **ShakeHands Conto** verwendet, da die Programme an vielen Stellen identisch sind. Sollte es an einer Stelle doch Unterschiede geben, so wird explizit darauf hingewiesen (Klammern bei Titeln). Bildschirmfotos stammen aus der Mac-Version der Anwendung, das Fensterlayout ist unter Windows aber identisch oder ähnlich.

Wir wünschen Ihnen viel Spass mit **ShakeHands Conto** und viel Erfolg.

Ihr ShakeHands-Team

Kapitel 2 Sprachreferenz

Wer sich in den Formularen und dem Generator auskennt, hat hier mit den Sprachreferenzen ein Handbuch zum Codieren. Suchen Sie Beispiele, Quelltexte oder ein Tutorial verweisen wir auf das Formulargenerator-Handbuch.

Hinweis: Ausführliche Beschreibungen, Templates und einen Einstieg in die Formularprogrammierung erhalten SIe um Handbuch Formulargenerator.

Kapitel 3 Anweisungen zur Formularsteuerung

Nachfolgend werden die möglichen Anweisungen zur Formularsteuerung beschrieben.

Bereichsdefinitionen

Die Bereiche sind Abschnitte innerhalb des Formular-Scripts. Sie sollen das Script übersichtlich gestalten und logisch gliedern. Darüber hinaus erfüllt jeder Bereich bestimmte Funktionen. Es gibt Bereiche, die im Script enthalten sein müssen und welche, die optional angegeben werden können. Der Tabellenbereich dient beispielsweise dazu, die Daten einer Tabelle auszugeben. Es können mehrere Tabellenbereiche angegeben werden. Die Ausgabe von Texten ist zum Beispiel im Bereich **BeginGraphicsText** möglich.

Hinweis: Der Scripteditor legt für ein neues Formular automatisch die notwendigen Bereiche von sich aus an.

Anweisungen

' (Kommentar)

Wird verwendet, um Kommentare einzufügen.

Syntax

' Kommentar

Parameter	Beschreibung
Kommentar	Der Kommentar, der zum Formularcode hinzugefügt werden soll.

Hinweise

Die **' (Kommentar)**-Anweisung fügt einen Kommentar in den Quelltext ein. Der Scriptinterpreter ignoriert alle Kommentare, die Sie nach dem Befehl **'** eingeben. Kommentare werden auch nicht in das fertige Formular übernommen.

Kommentare können am Zeilenende oder in einer separaten Zeile eingefügt werden. Gültige Kommentare erscheinen im Formular-Editor in grün.

Beispiel

Diese Beispiele verwenden die **' (Kommentar)**-Anweisung, um verschiedene Anweisungen zu dokumentieren.

```
'Berechnung der Summe
dim c as number = 12345 + 67890

msgBox(numToText(c)) 'Ausgabe der Summe in einer Dialogbox
```

#include scriptname

Wird verwendet, um andere Formular-Scripte einzubinden.

Syntax

#include "ScriptName"

Parameter	Beschreibung
ScriptName	Der Name des Formular-Scripts, das eingebunden werden soll.

Hinweise

Die **#include**-Anweisung bindet den Quellcode eines anderen Scripts in das aktuelle Script ein. Diese Anweisung kann hilfreich sein, wenn ein Script in mehreren anderen Formularen verwendet

wird, z.B. Titel- und Fusszeilen oder Schriften. Das Script kann dann mit der **#include**-Anweisung in jedes beliebige andere Formular-Script eingebunden werden.

Siehe auch Verwenden von Vorlagen (Templates).

Beispiel

Dieses Beispiel verwendet die **#include**-Anweisung, um den Quellcode eines Scripts in einem anderen Formular zu verwenden.

```
'das Beispiel bindet das Formular mit dem Namen „Titelzeile Formular“ ein
#include "Titelzeile Formular"
```

---

const name as type = value

Wird verwendet, um eine Konstante zu definieren.

Syntax

const Name as Datentyp = Wert

Parameter	Beschreibung
Name	Der Name der Konstante.
Datentyp	Der Datentyp (number, text, date, boolean, color) der Konstante.
Wert	Der Wert (number, text, date, boolean, color), welcher der Konstante zugewiesen werden soll.

Hinweise

Die **const**-Anweisung kann anstelle der **dim**-Anweisung verwendet werden, gefolgt von einer Wertzuweisung, wenn der Wert der Variablen im Script nicht geändert werden soll. Die Verwendung der Anweisung **const** anstelle von **dim** bietet einen bequemen Weg, solche Werte zu verwalten.

Siehe auch dim-Anweisung

Beispiel

```
'das Beispiel definiert die Konstante pi
const pi as number = 3.1415927

'das Beispiel definiert die Farbe „rot“ als eine Konstante
const ROT as color = RGBColor(255, 0, 0)

'das Beispiel definiert den Text „Euro“ als eine Konstante
const waehrung as text = "Euro"

'das Beispiel definiert das aktuelle Datum als eine Konstante
const datum as date = GetCurrentDate
```

---

dim variable as type [= value]

Wird verwendet, um eine Variable zu definieren.

Syntax

dim Name as Datentyp [= Wert]

Parameter	Beschreibung
Name	Der Name der Variable.
Datentyp	Der Datentyp (number, text, date, boolean, color) der Variable.
Wert	Optional der Wert, welcher der Variable zugewiesen werden soll.

Hinweise

Die **dim**-Anweisung kann verwendet werden, wenn der Wert der Variablen im Script geändert werden soll. Die Wertzuweisung ist optional. Sie muss also nicht zwingend bei der Variablendefinition gemacht werden, sondern kann auch später mit der **Set**-Anweisung erfolgen.

Siehe auch Set-Anweisung, Const-Anweisung.

Beispiel

```
'das Beispiel definiert eine Variable vom Typ number ohne Wertzuweisung
dim summe as number

'das Beispiel definiert eine Variable vom Typ color mit Wertzuweisung der Farbe
„rot“
dim farbe as color = RGBColor(255, 0, 0)

'das Beispiel definiert Variable vom Typ text mit Wertzuweisung
dim waehrung as text = "Euro"

'das Beispiel definiert eine Variable vom Typ boolean ohne Wertzuweisung
dim vergleich as boolean
```

---

set variable to value

Wird verwendet, um einer Variable einen Wert zuzuweisen.

Syntax

set Variable To Wert

Parameter	Datentyp	Beschreibung
Variable	-	Der Name der Variable, der ein Wert zugewiesen werden soll.
Wert	diverse	Der Wert, welcher der Variable zugewiesen werden soll.

Hinweise

Die **set**-Anweisung weist einer Variable einen Wert zu. Die zugewiesenen Werte müssen dem Datentyp der Variable entsprechen.

Siehe auch dim-Anweisung, const-Anweisung.

Beispiel

Diese Beispiele verwenden die **set**-Anweisung, um einer Variable eine Wert zuzuweisen.

```
'das Beispiel weist einer Zahl den Wert 12345 zu.
dim zahl as number
set zahl To 12345

'das Beispiel weist der Variable „geschlecht“ den Wert „m“ zu
dim geschlecht as text
set geschlecht To "m"
```

if...elseif...else...endif

Wird verwendet, um in Abhängigkeit von einem boole'schen Ausdruck eine Reihe von Anweisungen auszuführen.

Syntax

```
if (Bedingung)
        Anweisung
        [elseif (Bedingung-n)
                elseifAnweisungen]...
        [else
                elseAnweisungen]...
endif
```

Parameter	Beschreibung
Bedingung	Ein beliebiger Boole'scher Ausdruck.
Anweisung	Optional. Eine oder mehrere Anweisungen, die ausgeführt werden, falls Bedingung **True** ist.
Bedingung-n	Optional. Weitere Bedingungen wie Bedingung.
elseifAnweisung	Optional. Eine oder mehrere Anweisungen, die ausgeführt werden, wenn die damit verbundene Bedingung-n **True** ist.
elseAnweisung	Optional. Eine oder mehrere Anweisungen, die ausgeführt werden, falls keine vorherige Bedingung oder Bedingung-n **True** ist.

Hinweise

Wenn eine **if**-Anweisung ausgeführt wird, wird die Bedingung geprüft. Wenn diese Bedingung **True** ist, werden die damit nachfolgenden Anweisungen ausgeführt. Ist die Bedingung **False** und folgende **else**-Anweisung vorhanden, so werden dessen Anweisungen ausgeführt. Ist die Anweisung **False** und gibt es keine **else**-Anweisung oder steht davor eine **elseif**-Anweisung, so wird die Bedingung der **elseif**-Anweisung geprüft. Nachdem die **if**, **elseif** oder **else** folgende Befehle ausgeführt wurden, wird die Anweisung in der dem **endif** folgenden Zeile ausgeführt.

Nach einem **elseif** kann ein **else** stehen. Eine **elseif**-Anweisung darf jedoch nie nach einer **else**-Anweisung auftreten. Eine **if**-Anweisung muss immer mit einem **endif** beendet werden.

Beispiel

```
'das Beispiel zeigt eine einfache if-Anweisung
dim error as number = -123
if (error=-123)
   beep
   msgBox ("Ein Fehler trat auf!")
endif

'das Beispiel zeigt eine if-Anweisung, die elseif- und else-Anweisungen enthält
dim zahl as number
dim digits as number
set zahl To 33
if (zahl<10)
   set digits To 1
elseif (zahl<100)
   set digits To 2
else
   set digits To 3
Endif
```

do...exitloop...loop

Wird verwendet, um in Abhängigkeit von einem booleschen Ausdruck eine Reihe von Anweisungen in einer Schleife auszuführen.

Syntax

```
do
        Anweisung...
        exitloop([Bedingung])
loop
```

Parameter	Beschreibung
Anweisung	Eine oder mehrere Anweisungen, die wiederholt ausgeführt
Bedingung	Optional. Ein beliebiger Boole'scher Ausdruck.

Hinweise

Die **do-loop**-Anweisung wiederholt alle Anweisungen, welche zwischen **do** und **loop** angegeben sind solange, bis die Bedingung in der **exitloop**-Anweisung wahr, also **TRUE** ist. Die **exitloop**-Anweisung kann auch am Anfang der Schleife stehen.

Beispiel

Diese Beispiele verwenden die **do-loop**-Anweisung, um in Abhängigkeit einer Bedingung eine Anweisung in einer Schleife auszuführen .

```
'das Beispiel zeigt eine einfache do-loop Schleife, die Anweisung wird 10 mal
'wiederholt
dim i as number     = 0
dim hoehe as number = 0

do
   DrawText("Text " + NumToText(i), 0, hoehe, 500, 80)
   set i to i + 1
   set hoehe to hoehe + 40
   exitloop(i > 10)
loop

'das Beispiel zeigt eine do-loop Schleife, die Anweisung wird 5 mal wiederholt
dim i as number     = 5
dim hoehe as number = 0

do
   if ( i = 0 )
      exitloop
   endif
   DrawText("Count Down: " + NumToText(i), 0, hoehe, 500, 80)
   set i to i - 1
   set hoehe to hoehe + 40
loop
```

exit

Wird verwendet, um einen Bereich zu verlassen.

Syntax

exit

Hinweise

Die **exit**-Anweisung kann verwendet werden, wenn der Bereich (siehe Bereichsdefinition), in dem die Anweisung aufgerufen wird, vorzeitig verlassen werden soll. Alle Anweisungen, die nach **exit** angegeben sind, werden nicht ausgeführt.

Beispiel

```
'das Beispiel verlässt den Bereich „BeginDocument“, wenn die Summe kleiner 0 ist
#BeginDocument
dim summe as number
if (summe < 0)
   exit
endif

'das Beispiel verlässt den Bereich „BeginDocument“, wenn die Variable geschlecht
gleich „m“ ist. Die letzte Zeile wird dann nicht mehr ausgeführt!
```

```
#BeginDocument
dim geschlecht as text = "m"
if (geschlecht = "m")
   exit
endif
set geschlecht To "w"
```

msgbox(msg)

Wird verwendet, um eine Dialogbox mit dem angegebenen Text anzuzeigen.

Syntax **msgbox(Inhalt)**

Parameter	Datentyp	Beschreibung
Inhalt	text	Ein beliebiger Text, der in Der Dialogbox ausgegeben werden soll.

Hinweise Die **msgbox**-Anweisung zeigt eine Dialogbox mit dem angegebenen Text an

Beispiel

```
'das Beispiel gibt eine einfache Fehlermeldung aus
msgBox("Es ist ein Fehler aufgetreten!")

'das Beispiel gibt einen Hinweistext aus, wenn das Attribut „geschlecht“ nicht den
Wert „m“ oder „w“ hat
dim geschlecht as text
if (geschlecht <> "m" AND geschlecht <> "w")
   msgBox("Das Geschlecht wurde nicht korrekt angegeben! Mögliche Werte sind 'm'
oder 'w'")
endif
```

beep

Wird verwendet, um den System-Warnton wiederzugeben.

Syntax **beep**

Hinweise Die **beep**-Anweisung spielt den Warnton, der in der Ton-Systemeinstellung eingestellt wurde.

Beispiel

```
'das Beispiel spielt einen Warnton, wenn das Ergebnis False ist
dim a as boolean = True
dim b as boolean = False
if (a AND b = False)
   beep
endif
```

#BeginDocument

Wird verwendet, um den Beginn für den Dokumentbereich festzulegen. In diesem Bereich werden die Dokumenteigenschaften festgelegt.

Syntax **#BeginDocument**

Hinweise Die **#BeginDocument**-Anweisung definiert den Beginn des Dokumentbereichs. Der Bereich wird zuerst ausgeführt und dient dazu, den Rand des Dokuments festzulegen, sowie Konstanten und Variablen zu definieren. In diesem Abschnitt muss die Definition der Dokumenteigenschaften mit der Anweisung **SetDocument** erfolgen. Fehlt diese Anweisung, so wird kein Text und keine Grafik ausgegeben. Der Bereich muss im Script enthalten sein, darf aber nur einmal angegeben werden.

Siehe auch SetDocument.

Beispiel

```
'das Beispiel legt die Dokumenteigenschaften im Dokumentbereich und fest
#BeginDocument
SetDocument(200, 150, 100, 100, 16)
```

#BeginPage

Wird verwendet, um den Beginn für den Seitenbereich festzulegen. In diesem Bereich können seitenspezifische Angaben gemacht werden.

Syntax **#BeginPage**

Hinweise Die **#BeginPage**-Anweisung definiert den Beginn des Seitenbereichs und wird immer vor den grafischen Ausgabebereichen ausgeführt. In diesem Bereich können keine grafischen Ausgaben gemacht werden. Der Bereich wird auf jeder Seite ausgeführt und dient dazu, Berechnungen durchzuführen, Variablen zu definieren und Werte zuzuweisen. Dadurch kann beispielsweise eine individuelle Seitengestaltung erfolgen. Der Bereich kann im Script enthalten sein, darf aber nur einmal definiert werden.

Beispiel

```
'das Beispiel legt den Beginn des Seitenbereichs fest und weist der Variable seite
die aktuelle Seitennummer zu
#BeginPage
Set seite To PAGE_NUMBER
```

#BeginBackground

Wird verwendet, um den Beginn für den Hintergrundbereich festzulegen. In diesem Bereich kann der Seitenhintergrund definiert werden.

Syntax **#BeginBackground**

Hinweise Die **#BeginBackground**-Anweisung definiert den Beginn des Hintergrundbereichs. Hier kann der Hintergrund für alle oder spezielle Seiten definiert werden. Der Background Bereich ist der erste Bereich der gezeichnet wird. Der Bereich kann im Script enthalten sein, darf aber nur einmal definiert werden.

Beispiel

```
'das Beispiel legt das Hintergrundbild für den Hintergrundbereich fest
#BeginBackground
DrawPicture("Bild1.jpg", 0, 0, -1, PAGE_HEIGHT)

'das Beispiel fügt ein Hintergrundbild nur auf der 1. Seite ein
#BeginBackground
if (PAGE_NUMBER = 1)
   DrawPicture("Bild1.jpg", 0, 0, -1, PAGE_HEIGHT)
endif
```

#BeginHeader

Wird verwendet, um den Beginn für den Kopfbereich festzulegen. In diesem Bereich kann die grafische Ausgabe für die Kopfzeile festgelegt werden.

Syntax **#BeginHeader**

Hinweise Die **#BeginHeader**-Anweisung definiert den Beginn des Kopfbereichs. Der Kopfbereich ist der oberste Bereich auf der Seite für eine grafische Ausgabe. Mit **SetArea** muss eine feste Höhe für den Bereich definiert werden. Der Bereich kann im Script enthalten sein, darf aber nur einmal definiert werden.

Siehe auch SetArea.

Beispiel

```
'das Beispiel legt den Kopfbereich mit 20 mm Höhe fest
#BeginHeader
SetArea(200)
```

#BeginFooter

Wird verwendet, um den Beginn für den Fussbereich festzulegen. In diesem Bereich kann die grafische Ausgabe für die Fusszeile festgelegt werden.

Syntax **#BeginFooter**

Hinweise Die **#BeginFooter**-Anweisung definiert den Beginn des Fussbereichs. Der Fussbereich ist der letzte Bereich auf der Seite für eine grafische Ausgabe. Mit **SetArea** muss eine feste Höhe für den Bereich definiert werden. Der Bereich kann im Script enthalten sein, darf aber nur einmal definiert werden.

Siehe auch SetArea.

Beispiel

```
'das Beispiel legt den Fussbereich mit 20 mm Höhe fest
#BeginFooter
SetArea(200)
```

#BeginGraphicsText

Wird verwendet, um den Beginn für den grafischen Textbereich festzulegen. In diesem Bereich können beliebige Texte und Grafiken ausgegeben werden.

Syntax **#BeginGraphicsText**

Hinweise Die **#BeginGraphicsText**-Anweisung definiert den Beginn des grafischen Textbereichs. Der restliche Bereich zwischen **BeginHeader** und **BeginFooter** steht für den Bereich **BeginGraphicsText** zur Verfügung. Der Text bzw. die Grafiken sind in diesem Bereich frei positionierbar. Mit **SetArea** muss eine Bereichshöhe definiert werden. Die definierte Höhe stellt die maximale Höhe des Bereichs dar und kann an den Inhalt angepasst (nur verkleinert) werden. Die Anpassung der Höhe erfolgt mit der Anweisung **SetArea** durch den Parameter **autoSize**.
Bei einem Seitenumbruch wird immer der gesamte Bereich, der nicht mehr auf die Seite passt, auf der nächsten Seite ausgegeben. Der Bereich kann im Script vor oder nach einem Tabellenbereich beliebig oft definiert werden.

Siehe auch BeginHeader, BeginFooter, SetArea.

Beispiel

```
'das Beispiel legt den Beginn des grafischen Textbereichs fest, die maximale Bereichshöhe beträgt 100 mm und wird an den Inhalt angepasst
#BeginGraphicsText
SetArea(1000, AREA_AUTOSIZE)
```

#BeginTableHeader

Wird verwendet, um den Beginn für den Tabellenkopfbereich festzulegen. In diesem Bereich können beliebige Texte und Grafiken am Anfang der Tabelle ausgegeben werden.

Syntax **#BeginTableHeader**

Hinweise Der Bereich einer vollständigen Tabelle muss immer mindestens **BeginTableHeader** und **BeginTableFooter** enthalten. Die Bereiche **BeginTableRow** und **BeginTableBreak** können zwischen **BeginTableHeader** und **BeginTableFooter** eingefügt werden.

Die **#BeginTableHeader**-Anweisung definiert den Beginn des Tabellenkopfbereichs. Dieser Bereich kann im Script enthalten sein. Er muss aber immer bei Ausgabe einer Tabelle mit angegeben werden. In diesem Bereich können grafische Ausgaben erfolgen (Texte, Grafiken). Mit **SetArea** muss eine feste Höhe für den Bereich definiert werden.
Um auf den Inhalt (Felder) einer bestimmten Tabelle zugreifen zu können, muss die Tabelle mit **SetCurrentTable** in diesem Bereich ausgewählt werden. Nach **BeginTableHeader** kann der Bereich **BeginTableRow** definiert werden. Der Bereich darf nur einmal pro Tabellenbereich angegeben werden. Bei Seitenumbrüchen innerhalb einer Tabelle wird der Tabellenkopf auf jeder Seite neu ausgegeben.

Siehe auch SetArea, SetCurrentTable, #BeginTableRow, #BeginTableBreak, BeginTableFooter.

Beispiel

```
'das Beispiel legt den Tabellenkopfbereich mit 20 mm Höhe fest und wählt die Tabelle mit dem Namen „Tabelle1“ aus
#BeginTableHeader
SetArea(200)
SetCurrentTable("Tabelle1")
```

#BeginTableRow

Wird verwendet, um den Beginn für den Tabellenzeilenbereich festzulegen. In diesem Bereich können die Inhalte der in **BeginTableHeader** angegebenen Tabelle ausgegeben werden.

Syntax **#BeginTableRow**

Hinweise

Der Bereich einer vollständigen Tabelle muss mindestens immer **BeginTableHeader** und **BeginTableFooter** enthalten. Die Bereiche **BeginTableRow** und **BeginTableBreak** können zwischen **BeginTableHeader** und **BeginTableFooter** eingefügt werden.

Die **#BeginTableRow**-Anweisung definiert den Beginn des Tabellenzeilenbereichs. In diesem Bereich können grafische Ausgaben erfolgen (Texte, Grafiken). Mit **SetArea** muss eine Bereichshöhe definiert werden. Die definierte Höhe stellt die maximale Höhe des Bereichs dar und kann an den Inhalt angepasst (nur verkleinert) werden. Die Anpassung der Höhe erfolgt mit der Anweisung **SetArea** durch den Parameter **autoSize**.
Der Bereich wird entsprechend der Zeilenanzahl der ausgewählten Tabelle wiederholt. Der Seitenumbruch erfolgt automatisch. Wenn eine Zeile (ein Durchlauf von **BeginTableRow**) nicht mehr vollständig auf eine Seite passt, wird die Zeile (beziehungsweise die restliche Tabelle) auf der nächsten Seite ausgegeben. Der Bereich kann im Tabellenbereich des Scripts enthalten sein, darf aber nur einmal pro Tabellenbereich angegeben werden.

Siehe auch SetArea, SetCurrentTable, BeginTableHeader, #BeginTableBreak, BeginTableFooter.

Beispiel

```
'das Beispiel legt den den Tabellenzeilenbereich fest, die maximale Bereichshöhe
'beträgt 100 mm und wird an den Inhalt angepasst
#BeginTableRowSetArea
(1000, AREA_AUTOSIZE)
```

#BeginTableBreak

Wird verwendet, um den Beginn für den Tabellenumbruchbereich festzulegen. In diesem Bereich können Texte und Grafiken beim Seitenumbruch einer Tabelle am Ende der Seite ausgegeben werden.

Syntax **#BeginTableBreak**

Hinweise

Der Bereich einer vollständigen Tabelle muss immer mindestens **BeginTableHeader** und **BeginTableFooter** enthalten. Die Bereiche **BeginTableRow** und **BeginTableBreak** können zwischen **BeginTableHeader** und **BeginTableFooter** eingefügt werden.

Die **#BeginTableBreak**-Anweisung definiert den Beginn des Tabellenumbruchbereichs. Passt eine Tabelle nicht auf eine Seite, wird der Inhalt des TableBreak-Bereichs am Ende der Seite ausgegeben. Hier können z.B. Zwischensummen definiert werden. Mit **SetArea** muss eine feste Höhe für den Bereich definiert werden. Der Bereich kann im Tabellenbereich des Scripts enthalten sein, darf aber nur einmal pro Tabellenbereich angegeben werden.

Siehe auch SetArea, SetCurrentTable, BeginTableHeader, #BeginTableRow, BeginTableFooter.

Beispiel

Dieses Beispiel verwendet die **#BeginTableBreak**-Anweisung, um den Beginn für den Tabellenumbruchbereich festzulegen.

```
'das Beispiel legt den Tabellenumbruchbereich mit 10 mm Höhe fest
#BeginTableBreak
SetArea(100)
```

#BeginTableFooter

Wird verwendet, um den Beginn für den Tabellenfussbereich festzulegen. In diesem Bereich können beliebige Texte und Grafiken am Ende der Tabelle ausgegeben werden.

Syntax

#BeginTableFooter

Hinweise

Der Bereich einer vollständigen Tabelle muss immer mindestens **BeginTableHeader** und **BeginTableFooter** enthalten. Die Bereiche **BeginTableRow** und **BeginTableBreak** können zwischen **BeginTableHeader** und **BeginTableFooter** eingefügt werden.

Die **#BeginTableFooter**-Anweisung definiert den Beginn des Tabellenfussbereichs. Der Tabellenfussbereich ist ein Pflichtbereich. Dieser Bereich kann im Script enthalten sein, muss aber immer bei Ausgabe einer Tabelle mit angegeben werden. In diesem Bereich können grafische Ausgaben erfolgen (Texte, Grafiken). Der Tabellenfussbereich wird nur am Ende der Tabelle ausgegeben (auch bei längeren Tabellen mit Seitenumbrüchen). Hier können z.B. Endsummen definiert werden. Mit **SetArea** muss eine Bereichshöhe definiert werden. Die definierte Höhe stellt die maximale Höhe des Bereichs dar und kann an den Inhalt angepasst (nur verkleinert) werden. Die Anpassung der Höhe erfolgt mit der Anweisung **SetArea** durch den Parameter **autoSize**.

Siehe auch, SetArea, SetCurrentTable, BeginTableHeader, #BeginTableRow, #BeginTableRow.

Beispiel

Dieses Beispiel verwendet die **#BeginTableFooter**-Anweisung, um den Beginn für den Tabellenfussbereich festzulegen.

```
'das Beispiel legt den Tabellenfussbereich mit 20 mm Höhe fest
#BeginTableFooter
SetArea(200)
```

Anweisungen zur Definition von Eigenschaften

Mit diesen Anweisungen kann die Definition der einzelnen Bereiche und der Text- und Grafikformatierung vorgenommen werden.

SetDocument (left, top, right, bottom[, colorDepth])

Wird verwendet, um den Dokumentbereich zu definieren.

Syntax

SetDocument(Links, Oben, Rechts, Unten)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Rand (in zehntel Millimeter).
Oben	number	Der obere Rand (in zehntel Millimeter).
Rechts	number	Der rechte Rand (in zehntel Millimeter).

Parameter	Datentyp	Beschreibung
Unten	number	Der untere Rand (in zehntel Millimeter).

Hinweise

Die **SetDocument**-Anweisung definiert die Position des Dokumentbereichs. Sie kann nur einmal und ausschliesslich im Bereich **BeginDocument** verwendet werden.

Siehe auch BeginDocument.

Beispiel

```
'Das Beispiel definiert für das Dokument einen Rand von 10 mm.
SetDocument(100, 100, 100, 100)

'das Beispiel definiert für das Dokument 20 mm für den linken und rechten Abstand
und 30 mm für den oberen und unteren Abstand
SetDocument(200, 300, 200, 300)
```

SetArea(height[, autoSize])

Wird verwendet, um die maximale Höhe für einen Bereich zu definieren.

Syntax

SetArea(Hoehe[, Bedingung])

Parameter	Datentyp	Beschreibung
Hoehe	number	Die maximale Höhe des Bereichs (in zehntel Millimeter).
Bedingung	boolean	Optionaler Wert. **True**, wenn die Höhe automatische angepasst werden soll und **False**, wenn die Höhe nicht automatisch angepasst werden soll.

Hinweise

Die **SetArea**-Anweisung definiert die maximale Höhe des Bereichs, in der die Anweisung verwendet wird. Die **SetArea**-Anweisung kann nur für die Bereiche **BeginHeader**, **BeginGraphicsText**, **BeginTableHeader**, **BeginTableRow**, **BeginTableBreak**, **BeginTableFooter** und **BeginFooter** angewendet werden. Die Anweisung darf auch nur einmal im entsprechenden Bereich angegeben werden.

Optional kann eine automatische Höhenanpassung eingestellt werden. Hier wird die Höhe automatisch verringert, wenn der Inhalt kleiner ist als die angegebene maximale Höhe. Die automatische Höhenanpassung ist nur für die Bereiche **BeginGraphicsText** und **BeginTableRow** möglich.

Siehe auch BeginHeader, BeginGraphicsText, BeginTableHeader, BeginTableRow, BeginTableBreak, BeginTableFooter, BeginFooter.

Beispiel

```
'das Beispiel definiert für den Headerbereich eine Höhe von 25 mm
#BeginHeader
SetArea(250)

'das Beispiel definiert für den grafischen Textbereich eine Höhe von 80 mm und
eine automatische Anpassung der Höhe; die Konstante AREA_AUTOSIZE besitzt den Wert
„True“
#BeginGraphicsText
SetArea(800, AREA_AUTOSIZE)

'das Beispiel definiert für den Tabellenzeilenbereich eine Höhe von 50 mm ohne
automatischer Anpassung der Höhe
#BeginTableRow
SetArea(500, False)
```

SetTextFont(name)

Wird verwendet, um die Schriftart zu definieren.

Syntax

SetTextFont(Name)

Parameter	Datentyp	Beschreibung
Name	text	Die Schriftart, die verwendet werden soll.

Hinweise

Die **SetTextFont**-Anweisung definiert die Schriftart. In dem Auswahlfeld **Fonts** sind alle Schriftarten aufgelistet, die für die **SetTextFont**-Anweisung verwendet werden können. Standardmässig wird die Schriftart Arial verwendet. Die aktuelle Schriftart ist bis zur nächsten **SetTextFont**-Anweisung gültig.

Beispiel

```
'das Beispiel verwendet die Schriftart „Helvetica“
SetTextFont("Helvetica")
DrawText("Helvetica", 0, 30, 200, 50)

'das Beispiel verwendet die Schriftarten „Arial“ und „Times New Roman“
SetTextFont("Arial")
DrawText("Arial", 0, 30, 200, 50)
SetTextFont("Times New Roman")
DrawText("Times New Roman", 0, 100, 400, 50)
```

SetTextSize(size)

Wird verwendet, um die Schriftgrösse zu definieren.

Syntax

SetTextSize(Groesse)

Parameter	Datentyp	Beschreibung
Groesse	number	Die Schriftgrösse (in Punkten), die verwendet werden soll.

Hinweise

Die **SetTextSize**-Anweisung definiert die Schriftgrösse in Punkten. Vorgabe-Schriftgrösse beträgt **10** (10 Punkte). Die aktuelle Schriftgrösse ist bis zur nächsten **SetTextSize**-Anweisung gültig.

Beispiel

```
'das Beispiel verwendet die Schriftgrösse 12
SetTextSize(12)
DrawText("Schriftgrösse: 12 Punkte", 0, 30, 600, 50)

'das Beispiel verwendet die Schriftgrössen 12 und 16
SetTextSize(12)
DrawText("Schriftgrösse: 12 Punkte", 0, 30, 600, 50)
SetTextSize(16)
DrawText("Schriftgrösse: 16 Punkte", 0, 100, 600, 80)
```

SetTextColor(color)

Wird verwendet, um die Schriftfarbe zu definieren.

Syntax

SetTextColor(Farbe)

Parameter	Datentyp	Beschreibung
Farbe	color	Die Schriftfarbe, die verwendet werden soll.

Hinweise

Die **SetDocument**-Anweisung definiert die Schriftfarbe. Vorgabe ist die Farbe schwarz. Die aktuelle Schriftfarbe ist bis zur nächsten **SetTextColor**-Anweisung gültig. Die Farbtiefen setzen sich aus den Komponenten Rot, Grün, Blau zusammen mit den Skalen 0 bis 255 ändern Sie die Farbtiefen.

Siehe auch SetDocument-Anweisung, BeginDocument.

Beispiel

```
'das Beispiel verwendet die Schriftfarbe „schwarz“
dim fontFarbe as color
set fontFarbe to CMYColor(1, 1, 1)
SetTextColor(fontFarbe)
DrawText("Schwarz", 0, 30, 200, 50)

'das Beispiel verwendet die Schriftfarbe „rot“
dim fontFarbe as color
set fontFarbe to RGBColor(255, 0, 0)
SetTextColor(fontFarbe)
DrawText("Rot", 0, 30, 200, 50)

'das Beispiel verwendet die Schriftfarbe „gruen“
set fontFarbe to RGBColor(0, 255, 0)
SetTextColor(fontFarbe)
DrawText("Gruen", 0, 100, 200, 80)

'das Beispiel verwendet die Schriftfarbe „blau“
set fontFarbe to RGBColor(0, 0, 255)
SetTextColor(fontFarbe)
DrawText("Blau", 0, 100, 200, 80)
```

SetTextBold(bold)

Wird verwendet, um einen Text fett zu schreiben.

Syntax

SetTextBold(Bedingung)

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Wenn Bedingung **True** ist, wird der Text fett geschrieben, ansonsten nicht.

Hinweise

Die **SetTextBold**-Anweisung definiert die Textstil-Eigenschaft **Fett** für den entsprechenden Text. Vorgabe ist **SetTextBold** gleich False, d.h. der Text wird nicht fett geschrieben. Die aktuelle Formatierung ist bis zur nächsten **SetTextBold**-Anweisung gültig.

Beispiel

```
'das Beispiel verwendet die Textstil-Eigenschaft „Fett“
SetTextBold(True)
DrawText("Fett", 0, 30, 200, 50)

'der Text wird wieder normal geschrieben
SetTextBold(False)
DrawText("Normal", 0, 100, 200, 50)
```

SetTextItalic(italic)

Wird verwendet, um einen Text kursiv zu schreiben.

Syntax

SetTextItalic(Bedingung)

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Wenn Bedingung **True** ist, wird der Text kursiv geschrieben, ansonsten nicht.

Hinweise

Die **SetTextItalic**-Anweisung definiert die Textstil-Eigenschaft **Kursiv** für den entsprechenden Text. Standardmässig ist **SetTextItalic** False, d.h. der Text wird nicht kursiv geschrieben. Die aktuelle Formatierung ist bis zur nächsten **SetTextItalic**-Anweisung gültig.

Beispiel

```
'das Beispiel verwendet die Textstil-Eigenschaft „Kursiv“
SetTextItalic(True)
DrawText("Kursiv", 0, 30, 200, 50)

'der Text wird wieder normal geschrieben
SetTextItalic(False)
DrawText("Normal", 0, 100, 200, 50)
```

SetTextUnderline(underline)

Wird verwendet, um einen Text zu unterstreichen.

Syntax **SetTextUnderline(Bedingung)**

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Wenn Bedingung **True** ist, wird der Text unterstrichen, ansonsten nicht.

Hinweise

Die **SetTextUnderline**-Anweisung definiert die Textstil-Eigenschaft **Unterstreichen** für den entsprechenden Text. Vorgabe **SetTextUnderline** ist False, d.h. der Text wird nicht unterstrichen. Die aktuelle Formatierung ist bis zur nächsten **SetTextUnderline**-Anweisung gültig.

Beispiel

```
'das Beispiel verwendet die Textstil-Eigenschaft „Unterstrichen“
SetTextUnderline(True)
DrawText("Unterstreichen", 0, 30, 200, 50)

'der Text wird wieder normal geschrieben
SetTextUnderline(False)
DrawText("Normal", 0, 100, 200, 50)
```

SetTextAlignment(align)

Wird verwendet, um einen Text auszurichten nach links, zentriert oder rechts.

Syntax **SetTextAlignment(Ausrichtung)**

Parameter	Datentyp	Beschreibung
Ausrichtung	number	Wert, der die Textausrichtung definiert.

Hinweise

Die **SetTextAlignment**-Anweisung definiert die Textausrichtung. Standardmässig wird der Text links ausgerichtet. Die aktuelle Textausrichtung ist bis zur nächsten **SetTextAlignment**-Anweisung gültig. Die Textausrichtung kann auch direkt innerhalb der **DrawText**- und **DrawTextMonospace**-Anweisung definiert werden.

Die Ausrichtung erfolgt:

Wert	Ausrichtung
1	links
2	zentriert
3	rechts

Für eine einfache Auswahl der Ausrichtung, kann auch beim Auswahlfeld **Konstanten** der entsprechende Wert (**ALIGN_CENTER**, **ALIGN_LEFT**, **ALIGN_RIGHT**) gewählt werden. Siehe auch Definition der Formulareigenschaften.

Siehe auch DrawText, DrawTextMonospace, Auswahlliste für Konstanten.

Beispiel

```
'in dem Beispiel wird der Text links ausgerichtet
SetTextAlignment(1)
DrawText("links", 0, 30, PAGE_WIDTH, 50)

'der Text wird zentriert ausgerichtet
SetTextAlignment(ALIGN_CENTER)
DrawText("zentriert", 0, 100, PAGE_WIDTH, 50)
```

SetTextLineSpacing(factor)

Wird verwendet, um die Zeilenhöhe bei mehrzeiligen Texten zu definieren.

Syntax **SetTextLineSpacing(Faktor)**

Parameter	Datentyp	Beschreibung
Faktor	number	Wert, der die Zeilenhöhe bestimmt.

Hinweise

Die **SetTextLineSpacing**-Anweisung definiert die Zeilenhöhe bei der Ausgabe von mehrzeiligen Texten. Die Zeilenhöhe ist das **Faktor**-fache der Standardzeilenhöhe. Das bedeutet, mit einem Faktor kleiner als 1 verringert sich die Zeilenhöhe und mit einem Faktor grösser als 1 vergrössert sich der Zeilenabstand eines mehrzeiligen Textes. Ist der Faktor gleich 1, so bleibt die Zeilenhöhe unverändert. Vorgabe ist der Faktor der Zeilenhöhe gleich **1**.

Siehe auch DrawText**.**

Beispiel

```
'in dem Beispiel ist die Zeilenhöhe halb so gross wie die Standardzeilenhöhe
SetTextLineSpacing(0.5)
DrawText("Das ist ein mehrzeiliger Text mit einem 0,5-fachen Zeilenabstand.", 0,
30, 500, 200)

'in dem Beispiel ist die Zeilenhöhe doppelt so gross wie die Standardzeilenhöhe
SetTextLineSpacing(2)
DrawText("Das ist ein mehrzeiliger Text mit einem 2-fachen Zeilenabstand.", 0, 30,
500, 500)
```

SetLineWidth(width)

Wird verwendet, um die Linienstärke zu definieren.

Syntax **SetLineWidth(Staerke)**

Parameter	Datentyp	Beschreibung
Staerke	number	Die Linienstärke (in zehntel Millimeter).

Hinweise

Die **SetLineWidth**-Anweisung definiert die Linienstärke. Vorgabe ist die Linienstärke **0**. Das bedeutet, dass keine Linien gezeichnet werden. Um eine Linie zu zeichnen, muss zuvor die Linienstärke mit grösser **0** definiert werden. Die aktuelle Linienstärke ist bis zur nächsten **SetLineWidth**-Anweisung gültig.

Siehe auch DrawLine, DrawVerticalLine, DrawHorizontalLine, DrawRect, DrawRoundRec, DrawOval.

Beispiel

```
'in dem Beispiel wird die Linienstärke 1 definiert
SetLineWidth(1)
DrawHorizontalLine(10)

'in dem Beispiel wird die Linienstärke 4 definiert
SetLineWidth(4)
DrawHorizontalLine(20)
```

SetLineColor(color)

Wird verwendet, um die Linienfarbe zu definieren.

Syntax

SetLineColor(Farbe)

Parameter	Datentyp	Beschreibung
Farbe	color	Die Linienfarbe.

Hinweise

Die **SetLineColor**-Anweisung definiert die Linienfarbe. Standardmässig ist die Linienfarbe mit **schwarz** definiert. Die aktuelle Linienfarbe ist bis zur nächsten **SetLineColor**-Anweisung gültig.

Siehe auch DrawLine, DrawVerticalLine, DrawHorizontalLine, DrawRect, DrawRoundRec, DrawOval.

Beispiel

```
'in dem Beispiel wird die Linienfarbe „rot“ definiert
dim linienFarbe as color
SetLineWidth(1)
set linienFarbe to RGBColor(255, 0, 0)
SetLineColor(linienFarbe)
DrawHorizontalLine(10)
```

SetFillColor(color)

Wird verwendet, um die Füllfarbe zu definieren.

Syntax

SetFillColor(Farbe)

Parameter	Datentyp	Beschreibung
Farbe	color	Die Füllfarbe.

Hinweise

Die **SetFillColor**-Anweisung definiert die Füllfarbe, die bei den Anweisungen FillRect, FillRoundRect oder FillOval verwendet werden soll. Als Vorgabe ist die Füllfarbe mit **schwarz** definiert. Die aktuelle Füllfarbe ist bis zur nächsten **SetFillColor**-Anweisung gültig.

Siehe auch FillRect, FillRoundRect oder FillOval

Beispiel

```
'in dem Beispiel wird die Linienfarbe „blau“ definiert
dim fuellFarbe as color
set fuellFarbe to RGBColor(0, 0, 255)
```

```
SetFillColor(fuellFarbe)
FillRect(10,10,100,100)
```

SetCurrentTable(name)

Wird verwendet, um eine Tabelle zu wählen.

Syntax

SetCurrentTable(Name)

Parameter	Datentyp	Beschreibung
Name	text	Der Name der Tabelle, die verwendet werden soll.

Hinweise

Mit der **SetCurrentTable**-Anweisung kann eine Tabelle ausgewählt werden. Die auswählbaren Tabellen werden in einer Liste rechts neben dem Auswahlfeld **Felder(Global)** angezeigt. Die Anweisung kann nur innerhalb des Tabellenbereichs **BeginTableHeader** verwendet werden.

Siehe auch BeginTableHeader.

Beispiel

Dieses Beispiel verwendet die **SetCurrentTable**-Anweisung, um eine Tabelle auszuwählen.

```
'in dem Beispiel wird die Tabelle „Journal“ ausgewählt
#BeginTableHeader
SetCurrentTable("Journal")
```

Anweisungen zum Zeichnen von Text und Grafik

DrawText(text, left, top, width, height[, alignment])

Wird verwendet, um einen mehrzeiligen Text auszugeben.

Syntax

DrawText(Inhalt, Links, Oben, Breite, Hoehe[, Ausrichtung])

Parameter	Datentyp	Beschreibung
Inhalt	text	Ein beliebiger Text.
Links	number	Der Abstand vom linken Rand (in zehntel Millimeter).
Oben	number	Der Abstand vom oberen Rand (in zehntel Millimeter).
Breite	number	Die Breite des Zeichenbereiches für den Text (in zehntel Millimeter).
Hoehe	number	Die Höhe des Zeichenbereiches für den Text (in zehntel Millimeter).
Ausrichtung	number	Optionaler Wert für die Textausrichtung.

Hinweise

Die **DrawText**-Anweisung gibt einen mehrzeiligen Text aus. Ist der Text länger als die mit **DrawText** angegebene Breite, so wird der Text automatisch umgebrochen. Der Text wird unvollständig ausgegeben, wenn die Texthöhe des Feldinhaltes grösser ist als die bei **DrawText** angegebene Höhe. Optional kann die Textausrichtung (1 für links, 2 für zentriert und 3 für rechts) angegeben werden. Als Vorgabe wird der Text links ausgerichtet. Mit der Anweisung **SetTextLineSpacing** kann die Zeilenhöhe des mehrzeiligen Textes geändert werden.

Die **DrawText**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch DrawTextMonospace, SetTextLineSpacing.

Beispiel

```
'das Beispiel gibt den Text „Guten Morgen“ links aliniert aus
dim inhalt as text = "Guten Morgen"
DrawText(inhalt, 0, 0, 1000, 80)

'das Beispiel gibt „Guten Abend“ aus, der Text wird rechts ausgerichtet
dim inhalt as text = "Guten Abend"
DrawText(inhalt, 0, 0, 1000, 80, 3)
'das Beispiel gibt „Hallo“ aus, der Text wird zentriert ausgerichtet
dim inhalt as text = "Hallo"
DrawText(inhalt, 0, 0, 1000, 80, 2)
```

DrawTextMonospace(text, left, top, width, height, charWidth[, alignment])

Wird verwendet, um einen einzeiligen Text mit einem definierten Zeichenabstand auszugeben.

Syntax

DrawTextMonospace(Inhalt, Links, Oben, Breite, Hoehe, Zeichenweite[, Ausrichtung])

Parameter	Datentyp	Beschreibung
Inhalt	text	Ein beliebiger Text.
Links	number	Der Abstand vom linken Rand (in zehntel Millimeter).
Oben	number	Der Abstand vom oberen Rand (in zehntel Millimeter).
Breite	number	Die Breite des Zeichenbereiches für den Text (in zehntel Millimeter).
Hoehe	number	Die Höhe des Zeichenbereiches für den Text (in zehntel Millimeter).
Zeichenweite	number	Abstand der Zeichen (in zehntel Millimeter).
Ausrichtung	number	Optionaler Wert für die Textausrichtung (links oder rechts).

Hinweise

Die **DrawTextMonospace**-Anweisung gibt einen einzeiligen Text mit einem definierten Zeichenabstand aus. Der Zeichenabstand wird mit dem Parameter **Zeichenweite** festgelegt. Ist der Text länger als die mit **DrawTextMonospace** angegebene Breite, so wird der Text abgeschnitten. Es wird kein Text ausgegeben, wenn die Texthöhe des Feldes grösser ist als die bei **DrawTextMonospace** angegebene Höhe. Optional kann die Textausrichtung 1 für links und 3 für rechts angegeben werden. Als Vorgabe wird der Text links ausgerichtet.

Die **DrawTextMonospace**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch DrawText.

Beispiel

```
'das Beispiel gibt den Text „Grosser Zeichenabstand“ mit einem Zeichenabstand von
'5 mm aus
dim inhalt as text = "Grosser Zeichenabstand"
DrawTextMonospace(inhalt, 0, 0, 1000, 80, 50)

'das Beispiel gibt „Kleiner Zeichenabstand“ aus, der Text wird rechts ausgerichtet
'und hat einen Zeichenabstand von 2 mm
dim inhalt as text = "Kleiner Zeichenabstand"
DrawTextMonospace(inhalt, 0, 0, 1000, 80, 20, 3)
```

DrawLine(x1, y1, x2, y2)

Wird verwendet, um eine Linie zu zeichnen.

Syntax

DrawLine(X1, Y1, X2, Y2)

Parameter	Datentyp	Beschreibung
X1	number	Die vertikale Anfangsposition (in zehntel Millimeter.)
Y1	number	Die horizontale Anfangsposition (in zehntel Millimeter).
X2	number	Die vertikale Endposition (in zehntel Millimeter).
Y2	number	Die horizontale Endposition (in zehntel Millimeter).

Hinweise

Die **DrawLine**-Anweisung zeichnet in der aktuellen Linienfarbe eine Linie von X1, Y1 nach X2, Y2. Vor der **DrawLine**-Anweisung muss die Linienstärke mittels **SetLineWidth** definiert werden. Die Linien werden unvollständig gezeichnet, wenn die vertikale Linienlänge grösser ist als die mit **SetArea** definierte Höhe.

Die **DrawLine**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch SetLineColor, SetLineWidth, DrawVerticalLine, DrawHorizontalLine, SetArea.

Beispiel

```
'das Beispiel zeichnet eine horizontale Linie von 50 mm Länge und mit einer
'Linienstärke von 0,1 mm
SetLineWidth(1)
DrawLine(0, 0, 500, 0)

'das Beispiel zeichnet eine vertikale Linie von 100 mm Länge und mit einer
'Linienstärke von 1 mm
SetLineWidth(10)
DrawLine(0, 0, 0, 1000)

'das Beispiel zeichnet eine blaue diagonale Linie mit einer Linienstärke von 0,1
'mm
SetLineColor(RGBColor(0, 0, 255))
SetLineWidth(1)
DrawLine(0, 0, 1000, 1000)
```

DrawVerticalLine(x)

Wird verwendet, um eine vertikale Linie zu zeichnen.

Syntax

DrawVerticalLine(X)

Parameter	Datentyp	Beschreibung
X	number	Die horizontale Position, bei der die vertikale Linie gezeichnet werden soll (in zehntel Millimeter).

Hinweise

Die **DrawVerticalLine**-Anweisung zeichnet in der aktuellen Linienfarbe bei der Position X eine vertikale Linie. Vor der **DrawVerticalLine**-Anweisung muss die Linienstärke mittels **SetLineWidth** definiert werden. Die Linienlänge entspricht der in **SetArea** angegebenen Höhe.

Die **DrawVerticalLine**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch SetLineColor, SetLineWidth, DrawLine, DrawHorizontalLine, SetArea.

Beispiel

```
'das Beispiel zeichnet eine vertikale Linie von 100 mm Länge und mit einer Linien
'stärke von 0,1 mm bei der Position x=0 mm (linker Rand)
SetArea(100)
```

```
SetLineWidth(1)
DrawVerticalLine(0)
'das Beispiel zeichnet eine vertikale Linie von 10 mm Länge und mit einer Linien
'stärke von 1 mm am rechten Rand
SetArea(10)
SetLineWidth(10)
DrawVerticalLine(PAGE_WIDTH)
```

DrawHorizontalLine(y)

Wird verwendet, um eine horizontale Linie zu zeichnen.

Syntax

DrawHorizontalLine(Y)

Parameter	Datentyp	Beschreibung
Y	number	Die vertikale Position, bei der die horizontale Linie gezeichnet werden soll (in zehntel Millimeter).

Hinweise

Die **DrawHorizontalLine**-Anweisung zeichnet in der aktuellen Linienfarbe bei der Position Y über die gesamte Seite eine horizontale Linie. Vor der **DrawHorizontalLine**-Anweisung muss die Linienstärke mittels **SetLineWidth** definiert werden. Die vertikale Position muss innerhalb des mit **SetArea** definierten Bereichs liegen.

Die **DrawHorizontalLine**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch SetLineColor, SetLineWidth, DrawLine, DrawVerticalLine, SetArea.

Beispiel

```
'das Beispiel zeichnet eine horizontale Linie mit einer Linienstärke von 0,1 mm
'bei der Position y=0 mm (am Bereichsanfang)
SetArea(100)
SetLineWidth(1)
DrawHorizontalLine(0)

'das Beispiel zeichnet eine horizontale Linie mit einer Linienstärke von 1 mm
'bei der Position y=10 mm
SetArea(200)
SetLineWidth(10)
DrawHorizontalLine(100)
```

DrawPicture(name, left, top[, destWidth, destHeight])

Wird verwendet, um ein Bild einzufügen.

Syntax

DrawPicture(Name, Links, Oben[, Breite, Hoehe])

Parameter	Datentyp	Beschreibung
Name	text	Der Dateiname des Bildes.
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Optionaler Wert für die Breite des Bildes (in zehntel Millimeter).
Hoehe	number	Optionaler Wert für die Höhe des Bildes (in zehntel Millimeter).

Hinweise

Mit der **DrawPicture**-Anweisung kann ein Bild in das Formular eingefügt werden. Die Bilder sollten im Tiff-Format oder jpg-Format mit 300 dpi gespeichert werden, die Konvertierung in eine davon abweichende Druckerauflösung erfolgt automatisch. Alle im Formular benötigten Grafiken müssen in der **Bilderverwaltung** unter **Verwalten/Bilder** oder im **Bilder-Ordner** der Anwendung vorhanden sein.

Hinweis: Den Ordner **Bilder** öffnen Sie beim Mac wie folgt:

1. ShakeHands Programm auswählen (seit ShakeHands 2008 unter **Programme/ShakeHands**)
2. Paketinhalt anzeigen (Rechtsklick auf das Programm und aus dem Kontextmenü **Paketinhalt zeigen** auswählen)
3. Ordner **Contents** öffnen. In diesem Ordner befindet sich der Ordner **Bilder**.

Die **DrawPicture**-Anweisung sucht zuerst in der Bilderverwaltung nach dem Bild/den Bildern und erst dann, wenn das Bild nicht in der Bilderverwaltung gefunden werden konnte, im Ordner Bilder.

Optional kann die Breite und Höhe des Bildes angegeben werden. Das Bild wird entsprechend der Breite oder der Höhe proportional skaliert. Soll das Bild in der Höhe skaliert werden, so muss für die Breite der Wert **-1** definiert werden und umgekehrt. Werden sowohl Breite als auch Höhe angegeben, so ist die Angabe der Breite für die Skalierung des Bildes massgebend.

Bilder können in den Bereichen **BeginBackground**, **BeginHeader**, **BeginFooter**, **BeginGraphicsText**, **BeginTableHeader**, **BeginTableRow**, **BeginTableBreak**, und **BeginTableFooter** eingefügt werden. Hintergrundbilder (z.B. Wasserzeichen) sollten im Bereich **BeginBackground** eingefügt werden.

Die **DrawPicture**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Eine nähere Beschreibung zum Einfügen von Bildern und Logos finden Sie auch im Kapitel 4 unter Einfügen von Hintergrundbildern und Logos.

Siehe auch SetDocument.

Beispiel

```
'das Beispiel fügt das Bild mit dem Namen „Bild1.jpg“ in das Formular ein. Das
'Bild hat einen Abstand von 0 mm und wird in Originalgrösse dargestellt
DrawPicture("Bild1.jpg", 0, 0)

'das Beispiel fügt das Bild mit dem Namen „Breit.jpg“ in das Formular ein. Das
'Bild hat einen Abstand von 10 mm, die Breite des Bildes beträgt 150 mm
DrawPicture("Breit.jpg", 100, 100, 1500, -1)

'das Beispiel fügt das Bild mit dem Namen „Hoch.jpg“ in das Formular ein. Das Bild
'hat einen Abstand von 10 mm, die Höhe des Bildes beträgt 100 mm
DrawPicture("Hoch.jpg", 100, 100, -1, 1000)
```

DrawRect(left, top, width, height)

Wird verwendet, um ein Rechteck zu zeichnen.

Syntax

DrawRect(Links, Oben, Breite, Hoehe)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Die Breite des Rechtecks (in zehntel Millimeter).
Hoehe	number	Die Höhe des Rechtecks (in zehntel Millimeter).

Hinweise

Mit der **DrawRect**-Anweisung kann ein Rechteck in der aktuellen Linienfarbe gezeichnet werden. Vor der **DrawRect**-Anweisung muss die Linienstärke mittels **SetLineWidth** definiert werden. Das Rechteck wird unvollständig gezeichnet, wenn die Höhe des Rechtecks grösser ist, als die mit **SetArea** definierte Höhe.

Die **DrawRect**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch SetLineColor, SetLineWidth, DrawRoundRect, DrawOval, SetArea.

Beispiel

```
'das Beispiel zeichnet ein Rechteck mit der Breite 100 mm, der Höhe 200 mm und
'einem Abstand von 0 mm
SetLineWidth(1)
DrawRect(0, 0, 1000, 2000)

'das Beispiel zeichnet ein Rechteck mit der Breite 100 mm, der Höhe 10 mm und
'einem Abstand von 40 mm
SetLineWidth(1)
DrawRect(400, 400, 1000, 100)
```

DrawRoundRect(left, top, width, height, arcWidth, arcHeight)

Wird verwendet, um ein abgerundetes Rechteck zu zeichnen.

Syntax

DrawRoundRect(Links, Oben, Breite, Hoehe, Bogenbreite, Bogenhoehe)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Die Breite des Rechtecks (in zehntel Millimeter).
Hoehe	number	Die Höhe des Rechtecks (in zehntel Millimeter).
Bogenbreite	number	Wert für den Radius des Bogens in horizontaler Richtung (in zehntel Millimeter).
Bogenhoehe	number	Wert für den Radius des Bogens in vertikaler Richtung (in zehntel Millimeter).

Hinweise

Mit der **DrawRoundRect**-Anweisung kann ein abgerundetes Rechteck in der aktuellen Linienfarbe gezeichnet werden. Vor der **DrawRoundRect**-Anweisung muss die Linienstärke mittels **SetLineWidth** definiert werden. Das Rechteck wird unvollständig gezeichnet, wenn die Höhe des Rechtecks grösser ist, als die mit **SetArea** definierte Höhe.

Die **DrawRoundRect**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch SetLineColor, SetLineWidth, DrawRect, DrawOval, SetArea.

Beispiel

```
'das Beispiel zeichnet ein abgerundetes Rechteck mit einem Radius von 10 mm, der
'Breite 100 mm, der Höhe 200 mm und einem Abstand von 0 mm
SetLineWidth(1)
DrawRoundRect(0, 0, 1000, 2000, 100, 100)

'das Beispiel zeichnet ein abgerundetes Rechteck mit der Bogenbreite 10 mm, der
'Bogenhöhe 15 mm, der Rechteckbreite 100 mm, der Rechteckhöhe 10 mm und einem Ab
'stand von 40 mm
```

```
SetLineWidth(1)
DrawRoundRect(400, 400, 1000, 100, 150, 100)
```

DrawOval(left, top, width, height)

Wird verwendet, um einen Kreis zu zeichnen.

Syntax

DrawOval(Links, Oben, Breite, Hoehe)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Die Breite des Kreises (in zehntel Millimeter).
Hoehe	number	Die Höhe des Kreises (in zehntel Millimeter).

Hinweise

Mit der **DrawOval**-Anweisung kann ein Kreis in der aktuellen Linienfarbe gezeichnet werden. Vor der **DrawOval**-Anweisung muss die Linienstärke mittels **SetLineWidth** definiert werden. Der Kreis wird unvollständig gezeichnet, wenn die Höhe des Kreises grösser ist, als die mit **SetArea** definierte Höhe.

Die **DrawOval**-Anweisung kann nicht in den Bereichen **BeginDocument** und **BeginPage** verwendet werden.

Siehe auch SetLineColor, SetLineWidth, DrawRect, DrawRoundRect, SetArea.

Beispiel

```
'das Beispiel zeichnet einen Kreis mit einem Radius von 100 mm
SetLineWidth(1)
DrawOval(0, 0, 1000, 1000)

'das Beispiel zeichnet einen Kreis von 100 cm Höhe und 50 mm Breite
SetLineWidth(1)
DrawOval(0, 0, 500, 1000)
```

FillRect(left, top, width, height)

Wird verwendet, um ein gefülltes Rechteck zu zeichnen.

Syntax

FillRect(Links, Oben, Breite, Hoehe)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Die Breite des Rechtecks (in zehntel Millimeter).
Hoehe	number	Die Höhe des Rechtecks (in zehntel Millimeter).

Hinweise

Mit der **FillRect**-Anweisung kann ein gefülltes Rechteck in der aktuellen Füllfarbe gezeichnet werden. Das Rechteck wird unvollständig gezeichnet, wenn die Höhe des Rechtecks grösser ist, als die mit **SetArea** definierte Höhe.

Siehe auch FillColor, FillRoundRect, FillOval, SetArea.

Beispiel

```
'das Beispiel zeichnet ein gefülltes Rechteck mit der Breite 100 mm, der Höhe 200
'mm und einem Abstand von 0 mm
FillRect(0, 0, 1000, 2000)
```

```
'das Beispiel zeichnet ein rot gefülltes Rechteck mit der Breite 100 mm, der Höhe
'10 mm und einem Abstand von 40 mm
SetFillColor(RGBColor(255, 0, 0))
FillRect(400, 400, 1000, 100)
```

FillRoundRect(left, top, width, height, arcWidth, arcHeight)

Wird verwendet, um ein gefülltes, abgerundetes Rechteck zu zeichnen.

Syntax

FillRoundRect(Links, Oben, Breite, Hoehe, Bogenbreite, Bogenhoehe)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Die Breite des Rechtecks (in zehntel Millimeter).
Hoehe	number	Die Höhe des Rechtecks (in zehntel Millimeter).
Bogenbreite	number	Optionaler Wert für den Radius des Bogens in horizontaler Richtung (in zehntel Millimeter).
Bogenhoehe	number	Optionaler Wert für den Radius des Bogens in vertikaler Richtung (in zehntel Millimeter).

Hinweise

Mit der **FillRoundRect**-Anweisung kann ein gefülltes, abgerundetes Rechteck in der aktuellen Füllfarbe gezeichnet werden. Das Rechteck wird unvollständig gezeichnet, wenn die Höhe des Rechtecks grösser ist, als die mit **SetArea** definierte Höhe.

Siehe auch FillColor, FillRect, FillOval, SetArea.

Beispiel

```
'das Beispiel zeichnet ein gefülltes, abgerundetes Rechteck mit einem Radius von
'10 mm, der Breite 100 mm, der Höhe 200 mm und mit keinem Abstand
FillRoundRect(0, 0, 1000, 2000, 100, 100)

'das Beispiel zeichnet ein rot gefülltes, abgerundetes Rechteck mit der Bogen-
'breite 15mm, der Bogenhöhe 10mm, der Rechteckbreite 100mm, der Rechteckhöhe 10mm
'und einem Abstand von 40mm (von Links und von Oben)
SetFillColor(RGBColor(255, 0, 0))
FillRoundRect(400, 400, 1000, 130, 150, 100)
```

FillOval(left, top, width, height)

Wird verwendet, um einen gefüllten Kreis zu zeichnen.

Syntax

FillOval(Links, Oben, Breite, Hoehe)

Parameter	Datentyp	Beschreibung
Links	number	Der linke Abstand (in zehntel Millimeter).
Oben	number	Der Abstand von oben (in zehntel Millimeter).
Breite	number	Die Breite des Kreises (in zehntel Millimeter).
Hoehe	number	Die Höhe des Kreises (in zehntel Millimeter).

Hinweise

Mit der **FillOval**-Anweisung kann ein gefüllter Kreis in der aktuellen Füllfarbe gezeichnet werden. Der Kreis wird unvollständig gezeichnet, wenn die Höhe des Kreises grösser ist, als die mit **SetArea** definierte Höhe.

Siehe auch FillColor, FillRect, FillRoundRect, SetArea.

Beispiel

```
'das Beispiel zeichnet einen gefüllten Kreis mit der Breite 100 mm, der Höhe 200
'mm und keinem Abstand
FillOval(0, 0, 1000, 2000)

'das Beispiel zeichnet einen rot gefüllten Kreis mit der Breite 100 mm, der Höhe
'10 mm und einem Abstand von 40 mm
SetFillColor(RGBColor(255, 0, 0))
FillOval(400, 400, 1000, 100)
```

Sonstige Anweisungen

PageBreak([condition, beforeArea])

Wird verwendet, um einen manuellen Seitenumbruch einzufügen.

Syntax **PageBreak([Bedingung])**

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Optionaler Wert. Wenn Bedingung **True** ist, wird ein Seitenumbruch eingefügt, ansonsten nicht. Ohne Angabe von Bedingung wird immer ein Seitenumbruch eingefügt.

Hinweise

Die **PageBreak**-Anweisung fügt einen Seitenumbruch an der aktuellen Position ein. Alle Inhalte der nachfolgenden Bereiche werden auf eine neue Seite geschrieben. Alle Inhalte in dem Bereich der **PageBreak**-Anweisung werden auf die gleiche Seite geschrieben, auch wenn sie nach der Anweisung angegeben wurden. Die **PageBreak**-Anweisung kann nur in den Bereichen **BeginGraphicsText** und **BeginTableRow** angewendet werden.

Siehe auch BeginGraphicsText, BeginTableRow.

Beispiel

```
'das Beispiel fügt einen Seitenumbruch ein. Übertrag Text noch auf der ersten Sei-
te!
#BeginGraphicsText
SetArea(1000, AREA_AUTOSIZE)
DrawText("Übertrag", 0, 0, PAGE_WIDTH, 80)
PageBreak(True)
DrawText("Seite 1", 0, 100, PAGE_WIDTH, 80)
'Text auf der 2. Seite
#BeginGraphicsText
SetArea(1000, AREA_AUTOSIZE)
DrawText("Seite 2", 0, 0, PAGE_WIDTH, 80)
```

ExitTable([condition])

Wird verwendet, um die Ausgabe der aktuelle Tabelle abzubrechen.

Syntax **ExitTable([Bedingung])**

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Optionaler Wert. Wenn Bedingung **True** ist, wird die Tabellenausgabe abgebrochen, sonst nicht. Ohne Angabe von Bedingung wird die Tabelle immer abgebrochen.

Hinweise

Mit der **ExitTable**-Anweisung kann die Ausgabe der aktuellen Tabelle abgebrochen werden. Die Anweisung kann nur in den Tabellenbereichen angewendet werden.

Beispiel

```
'das Beispiel gibt nur die erste Tabellenzeile aus
#BeginTableRow
SetArea(200, true)
DrawText("Ende nach der 1. Tabellenzeile", 0, 0, PAGE_WIDTH, 80)
ExitTable()
```

Kapitel 4 Funktionen

Funktionen werden benötigt, um z.B. bestimmte Informationen abzufragen (Auskunftsfunktionen), um Berechnungen durchzuführen (Zahlenfunktionen), um Texte zu verändern (Textfunktionen), um einen Datentypen in einen anderen zu konvertieren, um Werte zu vergleichen (Vergleichsfunktionen) usw.
Nachfolgend werden die enthaltenen Funktionen der Scriptsprache beschrieben.

Auskunftsfunktionen

Booleanfunktionen

BooleanToText(expression)

Wandelt einen Wert vom Typ Boolean in einen Text um.

Syntax

Ergebnis = BooleanToText(Wert)

Parameter	Datentyp	Beschreibung
Wert	boolean	Der Boolean-Wert.
Ergebnis	text	Ergebnis ist Wert als Text (**True** oder **False**).

Hinweise

Die **BooleanToText**-Funktion konvertiert einen Boolean-Wert in einen Text um.

Beispiel

Diese Beispiele verwenden die **BooleanToText**-Funktion, um einen Boolean-Wert in einen Text umzuwandeln.

```
'das Beispiel gibt „True“ zurück
dim a as boolean = true
dim b as text = BooleanToText(a)

'das Beispiel gibt „False“ zurück
dim c as text = BooleanToText(false)
```

Datumsfunktionen

Date(day, month, year)

Erzeugt ein Datum.

Syntax

Ergebnis = Date(Tag, Monat, Jahr)

Parameter	Datentyp	Beschreibung
Tag	number	Der Tag im Monat des Datums als Zahlenwert.
Monat	number	Der Monat des Datums Zahlenwert.
Jahr	number	Das Jahr des Datums.
Ergebnis	date	Ergebnis ist ein Datum mit Tag, Monat und Jahr.

Hinweise

Die **Date**-Funktion erzeugt ein Datumsobjekt aus Tag, Monat und Jahr.

Siehe auch Day, Hour, Minute, Month, Second, Year, DateToText, DateToNumber. TimeToText, SQLDate, SQLDateTime.

Beispiel

.

```
'das Beispiel erzeugt ein Datumsobjekt vom aktuellen Datum
dim a as date = GetCurrentDate

'das Beispiel erzeugt ein Datumsobjekt vom 08.08.2008
dim tag as number = 8
dim monat as number = 8
dim jahr as number = 2008
dim datum as date = Date(tag, monat, jahr)
```

DateToNumber(date)

Wandelt ein Datum in eine Zahl um.

Syntax

Ergebnis = DateToNumber(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist das Datum als Zahl (in Millisekunden).

Hinweise

Die **DateToNumber**-Funktion gibt die Zeitspanne vom Startdatum (1.1.1904 0:00 Uhr) bis zum angegebenen Datum zurück. Die Umwandlung eines Datums in eine Zahl, ist für das einfachere Rechnen mit einem Datum vorgesehen. Dadurch ist es zum Beispiel möglich, zwei unterschiedliche Datumsangaben problemlos zu addieren oder zu subtrahieren.

Siehe auch DateToText, TimeToText.

Beispiel

```
'das Beispiel gibt das aktuelle Datum als Zahl zurück
dim datum as date = GetCurrentDate
dim datumNum as number = DateToNumber(datum)
```

DateToText(date)

Wandelt ein Datum in einen Text um.

Syntax

Ergebnis = DateToText(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist das Datum als Text.

Hinweise

Die **DateToText**-Funktion gibt das angegebene Datum als Text zurück. Das Datumsformat richtet sich dabei an die Formateinstellung (kurzes Datumsformat) des jeweiligen Betriebsystems (Mac, Windows). Ursprünglich ist das kurze Datumsformat beim Mac mit TT.MM.JJ (Beispiel: „08.08.08“) und unter Windows mit TT.MM.JJJJ (Beispiel: „08.08.2008“) definiert.

Siehe auch DateToNumber. TimeToText.

Beispiel

```
'das Beispiel gibt das aktuelle Datum als Text zurück
dim a as date = GetCurrentDate
dim b as text = DateToText(a)

'das Beispiel gibt „01.01.08“ zurück
dim tag as number = 1
dim monat as number = 1
dim jahr as number = 2008
```

```
dim datum as date = Date(tag, monat, jahr)
dim datumText as text = DateToText(datum)
```

Day(date)

Gibt den Tag im Monat eines Datums zurück.

Syntax

Ergebnis = Day(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist der Tag im Monat von Datum.

Hinweise

Die **Day**-Funktion gibt den Tag im Monat des angegebenen Datums als Zahl zurück.

Siehe auch Hour, Minute, Month, Second, Year.

Beispiel

```
'das Beispiel gibt den aktuellen Tag zurück
dim a as date = GetCurrentDate
dim b as number = Day(a)

'das Beispiel gibt „31“ zurück
dim datum as date = Date(31, 12, 2007)
dim tag as number = Day(datum)
```

Hour(date)

Gibt die Stunden aus der Uhrzeit des gespeicherten Datumsfeldes.

Syntax

Ergebnis = Hour(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis sind die Stunden aus der Uhrzeit von Datum.

Hinweise

Die **Hour**-Funktion gibt die Stunden aus der Uhrzeit des angegebenen Datums als Zahl zurück.

Siehe auch Day, Minute, Month, Second, Year.

Beispiel

```
'das Beispiel gibt die aktuelle Stunde zurück
dim datum as date = GetCurrentDate
dim std as number = Hour(datum)
```

Minute(date)

Gibt die Minuten aus der Uhrzeit des gespeicherten Datumsfeldes.

Syntax

Ergebnis = Minute(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis sind die Minuten aus der Uhrzeit von Datum.

Hinweise

Die **Minute**-Funktion gibt die Stunden aus der Uhrzeit des angegebenen Datums als Zahl zurück.

Siehe auch Day, Hour, Month, Second, Year.

Beispiel

```
'das Beispiel gibt die aktuelle Minuten zurück
dim datum as date = GetCurrentDate
dim min as number = Minute(datum)
```

Month(date)

Gibt den Monat eines Datums zurück.

Syntax **Ergebnis = Month(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist der Monat von Datum als Zahl.

Hinweise Die **Month**-Funktion gibt den Monat des angegebenen Datums als Zahl zurück.

Siehe auch Day, Hour, Minute, Second, Year.

Beispiel

```
'das Beispiel gibt den aktuellen Monat zurück
dim datum as date = GetCurrentDate
dim monat as number = Month(datum)
```

Second(date)

Gibt die Sekunden aus der Uhrzeit des gespeicherten Datumsfeldes.

Syntax **Ergebnis = Second(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis sind die Sekunden aus der Uhrzeit von Datum.

Hinweise Die **Second**-Funktion gibt die Sekunden aus der Uhrzeit des angegebenen Datums als Zahl zurück.

Siehe auch Day, Hour, Minute, Month, Year.

Beispiel

```
'das Beispiel gibt die aktuelle Sekunden zurück
dim datum as date = GetCurrentDate
dim sek as number = Second(datum)
```

SQLDate(date)

Wandelt ein Datum in ein SQL konformes Datum um.

Syntax **Ergebnis = SQLDate(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist Datum in SQL-Syntax als Text.

Hinweise Die **SQLDate**-Funktion gibt das angegebene Datum in SQL-Syntax als Text zurück.

Siehe auch SQLDateTime.

Beispiel Dieses Beispiel verwendet die **SQLDate**-Funktion, um ein SQL konformes Datum umzuwandeln.

```
'das Beispiel gibt „2008-12-31“ zurück
dim datum as date = Date(31, 12, 2008)
dim datumSQL as text = SQLDate(datum)
```

SQLDateTime(date)

Wandelt ein Datum in eine SQL konforme Zeit um.

Syntax **Ergebnis = SQLDateTime(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist die Zeit von Datum in SQL-Syntax als Text.

Hinweise Die **SQLDateTime**-Funktion gibt die Zeit des angegebenen Datums in SQL-Syntax als Text zurück.

Siehe auch SQLDate.

Beispiel Dieses Beispiel verwendet die **SQLDateTime**-Funktion, um ein Datum in eine SQL konforme Zeitangabe umzuwandeln.

```
'das Beispiel gibt „2008-12-31“ zurück
dim datum as date = Date(31, 12, 2008)
dim datumSQL as text = SQLDateTime(datum)
```

TimeToText(date)

Wandelt eine Uhrzeit in einen Text um.

Syntax **Ergebnis = TimeToText(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist die Uhrzeit von Datum als Text.

Hinweise Die **TimeToText**-Funktion gibt die Uhrzeit in Stunden und Minuten als Text zurück.

Siehe auch DateToText, DateToNumber. TimeToText.

Beispiel

```
'das Beispiel gibt die aktuelle Uhrzeit als Text zurück
dim datum as date = GetCurrentDate
dim zeit as text = TimeToText(a)
```

Year(date)

Gibt das Jahr eines Datums zurück.

Syntax **Ergebnis = Year(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist das Jahr von Datum als Zahl.

Hinweise Die **Year**-Funktion gibt das Jahr des angegebenen Datums als Zahl zurück.

Siehe auch Day, Hour, Minute, Month, Second.

Beispiel

```
'das Beispiel gibt das aktuelle Jahr zurück
dim datum as date = GetCurrentDate
dim jahr as number = Year(datum)
```

ShortDate(date)

Gibt das Datum mit kurzer Datumsformatierung als Text zurück.

Syntax

Ergebnis = ShortDate(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist das Datum als Text in kurzer Schreibweise.

Hinweise

Die **ShortDate**-Funktion gibt das angegebene Datum mit kurzer Datumsformatierung als Text zurück. Das Ergebnis ist dasselbe, wie bei der Funktion **DateToText**. Das Datumsformat richtet sich dabei an die Formateinstellung (kurzes Datumsformat) des jeweiligen Betriebsystems (Mac, Windows). Ursprünglich ist das kurze Datumsformat beim Mac mit TT.MM.JJ (Beispiel: „08.08.08“) und unter Windows mit TT.MM.JJJJ (Beispiel: „08.08.2008“) definiert.

Hinweis: Das Datum kann auch durch Ermittlung von Tag, Monat und Jahr aus dem Datumsobjekt individuell und systemunabhängig zusammengebaut werden. Der Tag wird über die Day-Funktion, der Monat über die Month-Funktion und das Jahr über die Year-Funktion ermittelt.

Siehe auch MediumDate, LongDate, ShortTime, LongTime.

Beispiel

Dieses Beispiel verwendet die **ShortDate**-Funktion, um ein Datum in kurzer Schreibweise auszugeben.

```
'das Beispiel gibt das aktuelle Datum in kurzer Formatierung zurück
dim a as date = GetCurrentDate
dim b as text = ShortDate(a)
```

MediumDate(date)

Gibt das Datum mit mittlerer Datumsformatierung als Text zurück.

Syntax

Ergebnis = MediumDate(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist das Datum als Text in mittlere Formatierung.

Hinweise

Die **MediumDate**-Funktion gibt das angegebene Datum mit mittlerer Datumsformatierung als Text zurück. Das Datumsformat richtet sich dabei an die Formateinstellung (mittleres Datumsformat bei Mac und langes Datumsformat bei Windows) des jeweiligen Betriebsystems. Bei Windows wird das mittlere Datumsformat aus dem langem Datumsformat abgeleitet. Ursprünglich ist das mittlere Datumsformat beim Mac mit TT.MM.JJJJ (Beispiel: „08.08.2008“) definiert. Unter Windows ist das ursprüngliche lange Datumsformat mit TTTT, T. MMMM JJJJ (Beispiel: „Freitag, 8. August 2008“) definiert. Das davon abgeleitete mittlere Datumsformat wäre TTT, T. MMM JJJJ (Beispiel: „Fr, 8. Aug 2008“).

Hinweis: Das Datum kann auch durch Ermittlung von Tag, Monat und Jahr aus dem Datumsobjekt individuell und systemunabhängig zusammen gebaut werden. Der Tag wird über die Day-Funktion, der Monat über die Month-Funktion und das Jahr über die Year-Funktion ermittelt.

Siehe auch ShortDate, LongDate, ShortTime, LongTime.

Beispiel

Dieses Beispiel verwendet die **MediumDate**-Funktion, um ein Datum in mittlerer Schreibweise auszugeben.

```
'das Beispiel gibt das aktuelle Datum in mittlerer Formatierung zurück
dim a as date = GetCurrentDate
dim b as text = MediumDate(a)
```

LongDate(date)

Gibt das Datum mit langer Datumsformatierung als Text zurück.

Syntax

Ergebnis = LongDate(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	text	Ergebnis ist das Datum als Text in langer Formatierung.

Hinweise

Die **LongDate**-Funktion gibt das angegebene Datum mit langer Datumsformatierung als Text zurück. Das Datumsformat richtet sich dabei an die Formateinstellung (langes Datumsformat) des jeweiligen Betriebsystems (Mac, Windows). Bei Windows wird das mittlere Datumsformat aus dem langen Datumsformat abgeleitet. Ursprünglich ist das lange Datumsformat beim Mac mit T. MMMM JJJJ (Beispiel: „8. August 2008“) und unter Windows mit TTTT, T. MMMM JJJJ (Beispiel: „Freitag, 8. August 2008“) definiert.

Hinweis: Das Datum kann auch durch Ermittlung von Tag, Monat und Jahr aus dem Datumsobjekt individuell und systemunabhängig zusammen gebaut werden. Der Tag wird über die Day-Funktion, der Monat über die Month-Funktion und das Jahr über die Year-Funktion ermittelt.

Siehe auch ShortDate, MediumDate, ShortTime, LongTime.

Beispiel

Dieses Beispiel verwendet die **LongDate**-Funktion, um ein Datum in langer Schreibweise auszugeben.

```
'das Beispiel gibt das aktuelle Datum in langer Formatierung zurück
dim a as date = GetCurrentDate
dim b as text = LongDate(a)
```

ShortTime(date)

Gibt die Zeit mit kurzer Zeitformatierung (ohne Sekunden) als Text zurück.

Syntax

Ergebnis = ShortTime(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Die Zeit.
Ergebnis	text	Ergebnis ist die Zeit als Text in kurzer Schreibweise.

Hinweise

Die **ShortTime**-Funktion gibt die Zeit des angegebenen Datums mit kurzer Zeitformatierung (in Stunden und Minuten) als Text zurück. Das Ergebnis ist dasselbe, wie bei der Funktion **TimeToText**. Das Zeitformat richtet sich dabei an die Formateinstellung des jeweiligen Betriebsystems (Mac, Windows). Ursprünglich ist das kurze Zeitformat beim Mac mit HH:mm (Beispiel: „00:08“, „23:10“) definiert. Unter Windows ist die Zeit mit HH:mm:ss (Beispiel: „13:05:44“) definiert. Das kurze Zeitformat wird davon abgeleitet mit HH:mm (Beispiel: „13:05“).

Hinweis: Die Uhrzeit kann auch durch Ermittlung von Stunde, Minute und Sekunde aus dem Datumsobjekt individuell und systemunabhängig zusammengebaut werden. Die Stunde wird über die Hour-Funktion, die Minuten über die Minute-Funktion und die Sekunden über die Second-Funktion ermittelt.

Siehe auch ShortDate, MediumDate, LongDate, LongTime.

Beispiel

Dieses Beispiel verwendet die **ShortTime**-Funktion, um eine Uhrzeit in kurzer Schreibweise auszugeben.

```
'das Beispiel gibt die aktuelle Uhrzeit in kurzer Formatierung zurück
dim a as date = GetCurrentDate
dim b as text = ShortTime(a)
```

LongTime(date)

Gibt die Zeit mit langer Zeitformatierung (mit Sekunden) als Text zurück.

Syntax

Ergebnis = LongTime(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Die Zeit.
Ergebnis	text	Ergebnis ist die Zeit als Text in langer Schreibweise.

Hinweise

Die **LongTime**-Funktion gibt die Zeit des angegebenen Datums mit langer Zeitformatierung (in Stunden, Minuten und Sekunden) als Text zurück. Das Zeitformat richtet sich dabei an die Formateinstellung des jeweiligen Betriebsystems (Mac, Windows). Ursprünglich ist das lange Zeitformat beim Mac mit HH:mm:ss (Beispiel: „00:08:12“, „23:10:05“) definiert. Unter Windows ist die Zeit mit HH:mm:ss definiert.

Hinweis: Die Uhrzeit kann auch durch Ermittlung von Stunde, Minute und Sekunde aus dem Datumsobjekt individuell und systemunabhängig zusammengebaut werden. Die Stunde wird über die Hour-Funktion, die Minuten über die Minute-Funktion und die Sekunden über die Second-Funktion ermittelt.

Siehe auch ShortDate, MediumDate, LongDate, ShortTime.

Beispiel

Dieses Beispiel verwendet die **LongTime**-Funktion, um eine Uhrzeit in langer Schreibweise auszugeben.

```
'das Beispiel gibt die aktuelle Uhrzeit in langer Formatierung zurück
dim a as date = GetCurrentDate
dim b as text = LongTime(a)
```

DayOfWeek(date)

Gibt den Wochentag des Datums als Nummer zurück.

Syntax

Ergebnis = DayOfWeek(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist die Nummer des Wochentages als Nummer

Hinweise

Die **DayOfWeek**-Funktion gibt den numerischen Wochentag eines Datums an.

Siehe auch Date, DateOfYear, WeekOfYear.

Beispiel

Dieses Beispiel verwendet die **DayOfWeek**-Funktion, um aus dem Belegdatum 21.04.2009 den Wochentag 2 herauszulesen und in einem Fenster darzustellen.

```
'das Beispiel gibt den Tag der Woche als Nummer zurück, wandelt den numerischen
Wert in einen Text und zeigt den Wert am Bildschirm an: 2
msgBox(NumtoText(DayOfWeek(BU_Belegdatum)))
```

DayOfYear(date)

Gibt aus dem Datum die Anzahl Tage des Jahres zurück..

Syntax **Ergebnis = DayOfYear(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist die Anzahl Tage als Nummer

Hinweise Die **DayOfYear**-Funktion gibt die Anzahl Tage von Anfang Jahr bis zum gewählten Datum als numerischen Wert zurück.

Siehe auch Date, DateOfWeek, WeekOfYear.

Beispiel Dieses Beispiel verwendet die **DayOfYear**-Funktion, um aus dem Belegdatum 21.04.2009 die Anzahl Tage seit 01.01.2009 als Nummer zurück zugeben und dieses in einer Ausgabe darzustellen.

```
'das Beispiel gibt 111 zurück
msgBox(NumtoText(DayOfYear(BU_Belegdatum)))
```

WeekOfYear(date)

Gibt aus dem Datum die Anzahl Wochen des Jahres zurück..

Syntax **Ergebnis = WeekOfYear(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist die Anzahl Wochen als Nummer

Hinweise Die **WeekOfYear**-Funktion gibt die Anzahl Wochen von Anfang Jahr bis zum gewählten Datum als numerischen Wert zurück.

Siehe auch Date, DateOfWeek, DayOfWeek

Beispiel Dieses Beispiel verwendet die **WeekOfYear**-Funktion, um aus dem Belegdatum 21.04.2009 die Anzahl Wochen seit 01.01.2009 als Nummer zurück zugeben und in einem Ausgabefenster anzuzeigen.

```
'das Beispiel gibt 17 zurück
msgBox(NumtoText(WeekOfYear(BU_Belegdatum)))
```

GetAppName

Gibt den Namen des Programms zurück.

Syntax **Ergebnis = GetAppName**

Parameter	Datentyp	Beschreibung
Ergebnis	text	Der Name des Programms.

Hinweise Die **GetAppName**-Funktion gibt den Namen des Programms zurück. Zum Beispiel: „ShakeHands Kontor 2008“, „ ShakeHands Conto 2008“, „ ShakeHands Faktura 2008“ oder „ShakeHands Budget 2008“

Beispiel Die Applikation ShakeHands Kontor 2008 ist gestartet.

```
'das Beispiel gibt „ShakeHands Kontor 2008“ zurück
dim a as text = GetAppName
```

GetAppVersion

Gibt die Version des Programms zurück.

Syntax **Ergebnis = GetAppVersion**

Parameter	Datentyp	Beschreibung
Ergebnis	text	Die Version des Programms.

Hinweise Die **GetAppVersion**-Funktion gibt die Version des Programms zurück.

Beispiel In diesem Beispiel wurde das Programm ShakeHands Kontor 2008 mit dem Release 5.3.4 benutzt.

```
'das Beispiel gibt „5.3.4“ zurück
dim a as text = GetAppVersion      GetCurrentDate
```

GetCurrentDate

Gibt das aktuelle Datum zurück.

Syntax **Ergebnis = GetCurrentDate**

Parameter	Datentyp	Beschreibung
Ergebnis	date	Das aktuelle Datum.

Hinweise Die **GetCurrentDate**-Funktion gibt das aktuelle Datum zurück.

Beispiel Dieses Beispiel verwendet die **GetCurrentDate**-Funktion, um das aktuelle Datum (14.01.2009) zu ermitteln. Die **DateToText**-Funktion wird verwendet, um das Datum als Text auszugeben.

Siehe auch DateToText.

```
'das Beispiel liefert für b „14.01.09“
dim a as date = GetCurrentDate
dim b as text = DateToText(a)
```

GetPlatform

Gibt das Betriebssystem zurück.

Syntax **Ergebnis = GetPlatform**

Parameter	Datentyp	Beschreibung
Ergebnis	number	Die Plattform.

Hinweise Die **GetPlatform**-Funktion gibt das verwendete Betriebssystem zurück. Für Mac OS Systeme wird **1** und für Windows Systeme wird **2** zurückgegeben.

Beispiel Die Anwendung läuft in diesem Beispiel auf einem Mac-System.

```
'das Beispiel liefert 1 für das Mac-System
dim a as number = GetPlatform
```

HasConstant(name)

Überprüft, ob eine Konstante schon vorhanden ist.

Syntax **Ergebnis = HasConstant(Name)**

Parameter	Datentyp	Beschreibung
Name	text	Der Name der Konstante.

Parameter	Datentyp	Beschreibung
Ergebnis	boolean	Ergebnis ist **True**, wenn der Konstanten-Name vorhanden ist und **False**, wenn nicht.

Hinweise

Die **HasConstant**-Funktion gibt **True** zurück, wenn die Konstante mit dem angegebenen Namen vorhanden ist und **False**, wenn sie nicht existiert.

Beispiel

```
'das Beispiel gibt „True“ zurück
const pi as number = 3.1415927
dim a as boolean = HasConstant("pi")

'das Beispiel gibt „False“ zurück, da die Definition der Konstante erst nach der
Überprüfung erfolgt
dim a as boolean = HasConstant("pi")
const pi as number = 3.1415927
```

HasField(name)

Überprüft, ob eine Tabellenfeld vorhanden ist.

Syntax

Ergebnis = HasField(Name)

Parameter	Datentyp	Beschreibung
Name	text	Der Name des Tabellenfeldes.
Ergebnis	boolean	Ergebnis ist **True**, wenn das Tabellenfeld Name vorhanden ist und **False**, wenn nicht.

Hinweise

Die **HasField**-Funktion gibt **True** zurück, wenn das Tabellenfeld mit dem angegebenen Namen vorhanden ist und **False**, wenn es nicht existiert.

Für die Überprüfung von Feldern einer Tabelle, muss die entsprechende Tabelle vorhanden sein und mit **SetCurrentTable** vor dem Aufrufen der **HasField**-Funktion im Bereich **BeginTableHeader** angegeben werden. Alle Globalen Felder sind im gesamten Formular gültig und können in jedem Bereich abgefragt werden.

Siehe auch SetCurrentTable, BeginTableHeader.

Beispiel

```
'Überprüfung des globalen Feldes „Firma_Email“ das Beispiel gibt „True“ zurück
dim a as boolean = HasField("Firma_Email")

'das Beispiel gibt „False“ zurück, da das Feld mit dem Namen „Kunde_Email“ nicht
existiert
dim b as boolean = HasField("Kunde_Email")

'Überprüfung des Feldes „J_Datum“ aus der Tabelle „Journal“ das Beispiel gibt
„True“ zurück, wenn die Tabelle „Journal“ vorhanden ist
#BeginTableHeader
SetCurrentTable("Journal")
dim c as boolean = HasField("J_Datum")
```

HasTable(name)

Überprüft, ob eine Tabelle vorhanden ist.

Syntax

Ergebnis = HasTable(Name)

Parameter	Datentyp	Beschreibung
Name	text	Der Name der Tabelle.
Ergebnis	boolean	Ergebnis ist **True**, wenn die Tabelle Name vorhanden ist und **False**, wenn nicht.

Hinweise

Die **HasTable**-Funktion gibt **True** zurück, wenn die Tabelle mit dem angegebenen Namen vorhanden ist und **False**, wenn sie nicht existiert. Die vorhandenen Tabellen werden in einer Liste rechts neben dem Auswahlfeld **Felder(Global)** angezeigt.

Beispiel

```
'das Beispiel gibt „True“ zurück
dim a as boolean = HasTable("Journal")

'das Beispiel gibt „False“ zurück, da die Tabelle mit dem Namen „Tabelle“ nicht
existiert
dim b as boolean = HasTable("Tabelle")
```

HasVariable(name)

Überprüft, ob eine Variable vorhanden ist.

Syntax

Ergebnis = HasVariable(Name)

Parameter	Datentyp	Beschreibung
Name	text	Der Name der Variable.
Ergebnis	boolean	Ergebnis ist **True**, wenn die Variable Name vorhanden ist und **False**, wenn nicht.

Hinweise

Die **HasVariable**-Funktion gibt **True** zurück, wenn die Variable mit dem angegebenen Namen vorhanden ist und **False**, wenn sie nicht existiert.

Beispiel

```
'das Beispiel gibt „True“ zurück
dim a as number = 1
dim b as boolean = HasVariable("a")

'das Beispiel gibt „False“ zurück, da die Variable mit dem Namen „x“ nicht exis-
tiert
dim c as number = 2
dim d as boolean = HasVariable("x")
```

Case(test1, result1[, test2, result2, ...], default)

Gibt in Abhängigkeit einer Bedingung einen bestimmten Wert zurück.

Syntax

Ergebnis = Case(Bedingung1, Ergebnis1, Bedingung2, Ergebnis2, ..., BedingungN, ErgebnisN, Default)

Parameter	Datentyp	Beschreibung
Bedingung1	boolean	Die erste Bedingung.
Ergebnis1	variant	Der Wert, der ausgegeben werden soll, wenn Bedingung1 **True** ist.
Bedingung2	boolean	Die zweite Bedingung.
Ergebnis2	variant	Der Wert, der ausgegeben werden soll, wenn Bedingung2 **True** ist.
BedingungN	boolean	Die n-te Bedingung.
ErgebnisN	variant	Der Wert, der ausgegeben werden soll, wenn BedingungN **True** ist.
Default	variant	Der Default-Wert, der ausgegeben werden soll, wenn alle Bedingungen **False** sind.
Ergebnis	variant	Der Wert von Ergebnis1, Ergebnis2. ..., ErgebnisN oder Default.

Hinweise

Die **Case**-Funktion gibt einen Wert in Abhängigkeit einer Bedingung zurück. Es muss immer ein Default-Wert mit angegeben werden. Der Default-Wert wird ausgegeben, wenn keine der angegebenen Bedingungen eintritt. Für das Ereignis kann ein beliebiger Datentyp gewählt werden. Alle Ereignisse müssen jedoch vom selben Datentyp sein.

Siehe auch Choose, IfThen.

Beispiel

Diese Beispiele verwenden die **Case**-Funktion, um zwei Bedingungen zu überprüfen.

```
'das Beispiel gibt true zurück; das Letzte false ist der Default-Wert
dim schalter as text = "an"
dim test as boolean = Case(schalter="an", true, schalter="aus", false, false)

'das Beispiel gibt false zurück; das Letzte false ist der Default-Wert
dim schalter as text = "aus"
dim test as boolean = Case(schalter="an", true, schalter="aus", false, false)

'das Beispiel gibt false zurück; das Letzte false ist der Default-Wert
dim schalter as text = "defekt"
dim test as boolean = Case(schalter="an", true, schalter="aus", false, false)
```

Choose(index, result0[, result1, ...], default)

Gibt in Abhängigkeit einer Bedingung einen bestimmten Wert zurück.

Syntax

Ergebnis = Choose(Wert, Ergebnis1, Ergebnis2, ..., ErgebnisN, Default)

Parameter	Datentyp	Beschreibung
Wert	number	Der Wert (Zahl) für die Auswahl eines Ergebnisses.
Ergebnis1	variant	Der Wert, der ausgegeben werden soll, wenn Wert 0 ist.
Ergebnis2	variant	Der Wert, der ausgegeben werden soll, wenn Wert 1 ist.
ErgebnisN	variant	Der Wert, der ausgegeben werden soll, wenn Wert N ist.

Parameter	Datentyp	Beschreibung
Default	variant	Der Default-Wert
Ergebnis	variant	Der Wert von Ergebnis1, Ergebnis2. ..., ErgebnisN oder Default.

Hinweise

Die **Choose**-Funktion wählt in Abhängigkeit einer Bedingung einen Wert aus und gibt diesen zurück. Es muss immer ein Default-Wert mit angegeben werden. Der Default-Wert wird ausgegeben, wenn die Bedingung nicht erfüllt wird. Die Bedingung muss eine ganze Zahl vom Typ **number** sein. Das 1. Ereignis wird ausgewählt, wenn der Wert für die Bedingung 0 ist, das zweite Ereignis, wenn die Bedingung 1 ist, usw. Für das Ereignis kann ein beliebiger Datentyp gewählt werden. Alle Ereignisse müssen jedoch vom selben Datentyp sein.

Siehe auch Case, IfThen.

Beispiel

```
'das Beispiel gibt „Frau“ zurück; „Fehler“ ist der Default-Wert
dim auswahl as number = 0
dim anrede as text = Choose(auswahl, "Frau", "Herr", "Fehler")

'das Beispiel gibt „Herr“ zurück; „Fehler“ ist der Default-Wert
dim auswahl as number = 1
dim anrede as text = Choose(auswahl, "Frau", "Herr", "Fehler")

'das Beispiel gibt „Fehler“ zurück; „Fehler“ ist der Default-Wert
dim auswahl as number = 2
dim anrede as text = Choose(auswahl, "Frau", "Herr", "Fehler")
```

IfThen(test, resultTrue, resultFalse)

Gibt in Abhängigkeit einer Bedingung einen bestimmten Wert zurück.

Syntax

Ergebnis = IfThen(Bedingung, Ergebnis1, Ergebnis2)

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Die Bedingung.
Ergebnis1	variant	Der Wert, der ausgegeben werden soll, wenn Bedingung **True** ist.
Ergebnis2	variant	Der Wert, der ausgegeben werden soll, wenn Bedingung **False** ist.
Ergebnis	variant	Der Wert von Ergebnis1 bzw. Ergebnis2.

Hinweise

Die **IfThen**-Funktion gibt in Abhängigkeit einer Bedingung einen bestimmten Wert zurück. Für das Ereignis kann ein beliebiger Datentyp gewählt werden. Alle Ereignisse müssen jedoch vom selben Datentyp sein.

Siehe auch Case, Choose.

Beispiel

```
'das Beispiel gibt „richtig“ zurück
dim test as boolean = true
dim ergebnis as text = IfThen(test=true, "richtig", "falsch")


'das Beispiel gibt „falsch“ zurück
dim test as boolean = false
dim ergebnis as text = IfThen(test=true, "richtig", "falsch")
```

Zahlenfunktionen

Abs(value)

Gibt den Absolut-Wert einer Zahl zurück.

Syntax **Ergebnis = Abs(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, deren Absolutwert Sie ermitteln wollen.
Ergebnis	number	Der Absolutwert von Wert.

Hinweise Die **Abs**-Funktion gibt das positive Äquivalent des angegebenen Wertes zurück.

Beispiel Diese Beispiele verwenden die **Abs**-Funktion, um den Absolutwert der angegebenen Zahlen zu ermitteln.

```
'beide Beispiele geben 24.9 zurück
dim d as number
set d to Abs(24.9)
set d to Abs(-24.9)
```

Bin(value)

Gibt den Binärwert einer Zahl zurück.

Syntax **Ergebnis = Bin(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die ins Binärsystem konvertiert werden soll.
Ergebnis	text	Die Binärzahl von Wert als Text.

Hinweise Die **Bin**-Funktion gibt die Binärzahl des angegebenen Wertes als Text zurück.
Wenn der Wert keine ganze Zahl ist, werden die Nachkommastellen abgeschnitten.

Siehe auch Hex, Oct

Beispiel Diese Beispiele verwenden die **Bin**-Funktion, um den Binärwert der angegebenen Zahlen zu ermitteln.

```
'beide Beispiele geben „1010“ zurück
dim binWert as text
set binWert to Bin(10)
set binWert to Bin(10.4)
```

Ceil(value)

Gibt den nächst grösseren ganzzahligen Wert einer Zahl zurück.

Syntax **Ergebnis = Ceil(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die auf Ceil gerundet werden soll.
Ergebnis	text	Der Ceil gerundete Wert.

Hinweise Die **Ceil**-Funktion gibt den übergebenen Wert, gerundet auf die nächst grössere ganze Zahl, zurück.

Siehe auch Floor.

Beispiel

Diese Beispiele verwenden die **Ceil**-Funktion, um den nächst grösseren ganzzahligen Wert der angegebenen Zahlen zu ermitteln.

```
'beide Beispiele geben 25 zurück
dim a as number
set a to Ceil(24.9)
set a to Ceil(24.4)

'beide Beispiele geben -24 zurück
dim b as number
set b to Ceil(-24.9)
set b to Ceil(-24.4)
```

Chr(value)

Gibt für einen ASCII-Wert das entsprechende Zeichen zurück.

Syntax

Ergebnis = Chr(Wert)

Parameter	Datentyp	Beschreibung
Wert	number	Der numerische Wert (Zeichen-Code) des gewünschten Zeichens, in einem Bereich zwischen **0** und **255**.
Ergebnis	text	Das Zeichen, des übergebenen ASCII-Werts.

Hinweise

Die **Chr**-Funktion gibt das Zeichen, des angegebenen Zeichen-Codes zurück. Diese Funktion ist nur für ASCII-Zeichen gültig. Bei einem Wert grösser **255** wird der nicht-teilbare Rest, welcher bei der Teilung des entsprechenden Wertes mit 256 entsteht (Modulorechnung, siehe Mod-Funktion), für die Ermittlung des Zeichens verwendet (Beispiel: für 363 ist der nicht-teilbare Rest bei der Division mit 256 gleich 107; Das Zeichen mit dem ASCII-Code 363 entspricht somit dem Zeichen mit dem Code 107). Bei Werten kleiner **0** wird der nicht-teilbare Rest von der Division mit 256, erweitert um 256 für die Ermittlung des Zeichens zugrunde gelegt (Beispiel: für -405 ist der nicht-teilbare Rest -149, -149 + 256 ergibt 107; Das Zeichen mit dem ASCII-Code -405 entspricht somit dem Zeichen mit dem Code 107).

Siehe auch Asc.

Beispiel

Diese Beispiele verwenden die **Chr**-Funktion, um die Zeichen der angegebenen ASCII-Werte zu ermitteln.

```
'Der ASCII-Code 64 liefert das Zeichen "@"
dim ascii_1 as number = 64
dim z_1 as text = Chr(ascii_1)
```

```
'Der ASCII-Code 9 liefert das Tabulator-Zeichen "TAB"
dim ascii_2 as number = 9
dim z_2 as text = Chr(ascii_2)
```

Div(value, divisor)

Gibt den ganzzahligen Anteil bei der Division zweier Zahlen zurück.

Syntax

Ergebnis = Div(Wert, Divisor)

Parameter	Datentyp	Beschreibung
Wert	number	Der numerische Wert, der geteilt werden soll.
Divisor	number	Der numerische Wert, durch den geteilt werden soll.
Ergebnis	number	Der ganzzahlige Anteil der Division von Wert und Divisor.

Hinweise

Die Div-Funktion gibt den ganzzahligen Anteil der Division zweier Zahlen zurück.

Siehe auch / Operator.

Beispiel

Diese Beispiele verwenden die Div-Funktion, um den ganzzahligen Anteil der Division für die angegebenen Werte zu ermitteln.

```
'Der ganzzahlige Anteil bei der Division von 23 durch 4 ist 5
dim a as number
set a to Div(23, 4)

'Der ganzzahlige Anteil bei der Division von 23 durch 5 ist 4
dim b as number
set b to Div(23, 5)
```

Exp(value)

Berechnet für einen Wert die Exponentialfunktion.

Syntax

Ergebnis = Exp(Wert)

Parameter	Datentyp	Beschreibung
Wert	number	Der numerische Wert, mit dem gerechnet werden soll.
Ergebnis	number	Das Ergebnis der Exponentialfunktion mit dem Exponenten Wert.

Hinweise

Die **Exp**-Funktion gibt, für den angegebenen Exponenten, das Ergebnis der Exponentialfunktion zurück.

Beispiel

Dieses Beispiel verwendet die **Exp**-Funktion, um e^{10} zu berechnen.

```
'Das Ergebnis der Exponentialfunktion mit dem Exponent 10 ist
'22026,46579480671789497
dim a as number
set a to Exp(10)
```

Floor(value)

Gibt den nächst kleineren ganzzahligen Wert einer Zahl zurück.

Syntax

Ergebnis = Floor(Wert)

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die auf Floor gerundet werden soll.
Ergebnis	text	Der Floor gerundete Wert.

Hinweise

Die **Floor**-Funktion gibt den übergebenen Wert, gerundet auf die nächst kleinere ganze Zahl, zurück.

Siehe auch Ceil.

Beispiel

Diese Beispiele verwenden die **Floor**-Funktion, um den nächst kleineren ganzzahligen Wert der angegebenen Zahlen zu ermitteln.

```
'beide Beispiele geben 24 zurück
dim a as number
set a to Floor(24.9)
set a to Floor(24.4)

'beide Beispiele geben -25 zurück
dim b as number
set b to Floor(-24.9)
set b to Floor(-24.4)
```

Format(value, format)

Formatiert eine Zahl anhand von Parametern und liefert das formatierte Ergebnis als Text zurück. Die Format-Funktion ist ähnlich der Art und Weise, in der in Tabellenkalkulations-Anwendungen Zahlen formatiert ausgegeben werden können.

Syntax

Ergebnis = Format(Wert, Format)

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die formatiert werden soll.
Format	text	Definiert das Format, das auf die Zahl angewendet werden soll.
Ergebnis	text	Die nach Format formatierte Zahl.

Hinweise

Der **Format**-Parameter ist ein Text aus einem oder mehreren besonderen Zeichen, die bestimmen, wie eine Zahl formatiert wird:

Zeichen	Beschreibung
#	Platzhalter, der ein Zeichen der Zahl ausgibt, falls an der Stelle ein Zeichen vorhanden ist.
0	Platzhalter, der ein Zeichen der Zahl ausgibt, falls an der Stelle ein Zeichen vorhanden ist. Ist kein Zeichen vorhanden, so wird stattdessen 0 (null) ausgegeben.
.	Platzhalter für die Position des Dezimalpunktes.
,	Platzhalter, der angibt, ob die Zahl mit Tausender-Punkten formatiert werden
%	Die mit 100 multiplizierte Zahl in Prozent.
(	Gibt eine offene Klammer aus.
)	Gibt eine geschlossene Klammer aus.
+	Gibt links von der Zahl das Vorzeichen der Zahl aus.
-	Gibt ein negatives Vorzeichen links von der Zahl aus, falls die Zahl negativ ist. Auf positive Zahlen hat dies keine Auswirkung.
E oder e	Zeigt die Zahl in wissenschaftlicher Notation an.
\character	Zeigt das Zeichen, das dem Backslash (\) folgt.

Die **Format**-Funktion zeigt immer den absoluten Wert der Zahl an. Damit die richtigen Vorzeichen ausgegeben werden, müssen die Zeichen „+“ oder „-“ verwendet werden. Bei Angabe des Dezimalpunktes, wird die zuletzt angegebene Nachkommastelle nach den Rundungsregeln entsprechend auf- bzw. abgerundet.
Bitte beachten Sie, dass die tatsächlich ausgegebenen Zeichen zusätzlich von den „Zahlen“-Formatierungseinstellungen des Systems abhängen.

FormatDef kann aus bis zu drei Formatierungszuweisungen bestehen, die durch jeweils ein Semikolon unterteilt werden. Das erste Format wird zur Formatierung positiver Werte verwendet, das zweite Format für negative Werte. Das dritte Format wird verwendet, um den Wert 0 (Null) zu formatieren.

Beispiel

In der nachfolgenden Tabelle sind mehrere Beispiele zur Formatierung mit Hilfe der Sonderzeichen zu finden.

Format	Zahl	Ausgegebener Text
#.##	1234	1,23
#.0000	1.2	1,2000
0000	1	0001
#%	0.25	25%
#,#.#	1234567.89	1.234.567,9
#.##e+	1234567.89	1,23e+6
-#.##	-1.23	-1,23
+#.##	1.23	+1,23
#.##;(#.##);\n\u\l\l	1.23	1,23
#.##;(#.##);\n\u\l\l	-1.23	(1,23)
#.##;(#.##);\n\u\l\l	0	null

Das folgende Beispiel verwendet die **Format**-Funktion, um die Zahl 3550,5 in 3.550,50€ zu formatieren

```
'Währungsformatierung mit Tausenderpunkt,2 Nachkommastellen und „€“-Zeichen
dim a as number = 3550.5
Format(a,"#,0.00\€"))
```

Hex(value)

Gibt den Hexadezimalwert einer Zahl zurück.

Syntax

Ergebnis = Hex(Wert)

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die ins Hexadezimalsystem konvertiert werden soll.
Ergebnis	text	Die übergebene Zahl in hexadezimaler Schreibweise.

Hinweise

Die **Hex**-Funktion gibt die Hexadezimalzahl des angegebenen Wertes als Text zurück.
Wenn der Wert keine ganze Zahl ist, werden die Nachkommastellen abgeschnitten.

Siehe auch Bin, Oct.

Beispiel Diese Beispiele verwenden die **Hex**-Funktion, um den Hexadezimalwert der angegebenen Zahlen zu ermitteln.

```
'beide Beispiele geben „A“ zurück
dim hexWert as text
set hexWert to Hex(10)
set hexWert to Hex(10.4)
```

Log(value)

Gibt den natürlichen Logarithmus (Logarithmus zur Basis „e“) einer Zahl zurück.

Syntax **Ergebnis = Log(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, dessen natürlicher Logarithmus berechnet werden soll.
Ergebnis	number	Der natürliche Logarithmus von Wert.

Hinweise Die **Log**-Funktion gibt den natürlichen Logarithmus des angegebenen Wertes zurück.

Beispiel Dieses Beispiel verwendet die **Log**-Funktion, um den natürlichen Logarithmus der angegebenen Zahl zu ermitteln. Anschliessend wird das Ergebnis mit der Format-Funktion mit 7 Nachkommastellen formatiert.

```
'das Beispiel gibt für b 2,3025851 zurück
dim a as number = Log(10)
dim b as text = Format(a,"#.#######")
```

Max(value1, value2)

Gibt die grössere von zwei angegebenen Zahlen zurück.

Syntax **Ergebnis = Max(Wert1, Wert2)**

Parameter	Datentyp	Beschreibung
Wert1	number	Die erste Zahl, die verglichen werden soll.
Wert2	number	Die zweite Zahl, die verglichen werden soll.
Ergebnis	number	Die grössere Zahl von Wert1 und Wert2.

Hinweise Die **Max**-Funktion vergleicht zwei Zahlen und gibt die grössere von Beiden zurück.

Siehe auch Min.

Beispiel Diese Beispiele verwenden die **Max**-Funktion, um den grösseren der beiden angegebenen Werte zu ermitteln.

```
'das Beispiel gibt 4 zurück
dim a as number = Max(4,-2)

'das Beispiel gibt -2 zurück
dim b as number = Max(-4,-2)
```

Min(value1, value2)

Gibt die kleinere von zwei angegebenen Zahlen zurück.

Syntax **Ergebnis = Min(Wert1, Wert2)**

Parameter	Datentyp	Beschreibung
Wert1	number	Die erste Zahl, die verglichen werden soll.
Wert2	number	Die zweite Zahl, die verglichen werden soll.
Ergebnis	number	Die kleinere Zahl von Wert1 und Wert2.

Hinweise

Die **Min**-Funktion vergleicht zwei Zahlen und gibt die kleinere von Beiden zurück.

Beispiel

Diese Beispiele verwenden die **Min**-Funktion, um den kleineren der beiden angegebenen Werte zu ermitteln.

```
'das Beispiel gibt -2 zurück
dim a as number = Max(4,-2)

'das Beispiel gibt -4 zurück
dim b as number = Max(-4,-2)
```

Mod(value1, value2)

Gibt den nicht-teilbaren Rest nach der Division von zwei angegebenen Zahlen zurück.

Syntax

Ergebnis = Mod(Wert1, Wert2)

Parameter	Datentyp	Beschreibung
Wert1	number	Ein beliebiger numerischer Wert (Dividend).
Wert2	number	Ein beliebiger numerischer Wert (Divisor).
Ergebnis	number	Der Rest der Division von Wert1 durch Wert2.

Hinweise

Die **Mod**-Funktion teilt zwei Zahlen und gibt den nicht-teilbaren Rest als Ergebnis zurück. Wenn Wert2 Null ist, liefert Mod Wert1. Ist Wert1 positiv, so ist das Ergebnis immer positiv, auch wenn Wert2 negativ ist. Ist Wert1 negativ, so ist das Ergebnis immer negativ. Zum Beispiel:

```
Mod(17,-10) ergibt 7, denn 17/-10=-1 Rest 7, da -10*(-1)+7=17
Mod(-17,10) ergibt -7, denn -17/10=-1 Rest -7, da 10*(-1)-7=-17
Mod(-17,-10) ergibt -7, denn -17/-10=1 Rest -7, da -10*1-7=-17
```

Der Operator **Mod** arbeitet mit ganzen Zahlen. Wenn Wert1 oder Wert2 keine ganze Zahl ist, werden die Nachkommastellen abgeschnitten. Zum Beispiel:

```
Mod(7,2) ergibt 1
Mod(7,1.9999) ergibt 0
```

Beispiel

Diese Beispiele verwenden die **Mod**-Funktion, um den nicht-teilbaren Rest der beiden angegebenen Werte zu ermitteln.

```
'das Beispiel gibt 1 zurück
dim a as number = Mod(10,3)

'das Beispiel gibt 2 zurück
dim b as number = Max(2,4)

'das Beispiel gibt 0 zurück
dim b as number = Max(4,2)

'das Beispiel gibt 4 zurück
dim c as number = Max(4,0)

'das Beispiel gibt 0 zurück
dim d as number = Max(0,4)
```

```
'das Beispiel gibt 0 zurück
dim e as number = Max(4.5,1)

'das Beispiel gibt 1 zurück
dim f as number = Max(9.5,2.75)
```

NumToDate(value)

Gibt das Datum zu der angegebenen Zahl zurück.

Syntax **Ergebnis = NumToDate(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die in ein Datum umgewandelt werden soll, angegeben in Sekunden.
Ergebnis	date	Das Datum von Wert.

Hinweise Die **NumToDate**-Funktion wandelt eine Zahl in ein Datum um. Die angegebene Zahl entspricht der Zeit in Sekunden, ausgehend vom 01.01.1904. Das Format des ausgegebenen Datums richtet sich nach der Datumseinstellung des Systems.

Beispiel Diese Beispiele verwenden die **NumToDate**-Funktion, um die angegebenen Zahlen in ein Datum umzuwandeln.

```
'das Beispiel gibt 01.01.1904 zurück
dim a as date = NumToDate(0)

'das Beispiel gibt 01.01.2008 zurück
dim b as date = NumToDate(3311280000)
```

NumToText(value)

Gibt die angegebene Zahl als Text zurück.

Syntax **Ergebnis = NumToText(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die als Text ausgegeben werden soll
Ergebnis	text	Der Wert als Text.

Hinweise Die **NumToText**-Funktion wandelt eine Zahl in einen Text um.

Beispiel Diese Beispiele verwenden die **NumToText**-Funktion, um die angegebenen Werte als Text auszugeben.

```
'das Beispiel gibt „1234,5“ zurück
dim a as text = NumToText(1234.5)

'das Beispiel gibt „-0,1234“ zurück
dim b as text = NumToText(-0.1234)
```

Oct(value)

Gibt den Oktalwert einer Zahl zurück

Syntax **Ergebnis = Oct(Wert)**

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die ins Oktalsystem konvertiert werden soll.
Ergebnis	text	Die Oktalzahl von Wert als Text.

Hinweise

Die **Oct**-Funktion gibt die Oktalzahl des angegebenen Wertes als Text zurück.
Wenn der Wert keine ganze Zahl ist, werden die Nachkommastellen abgeschnitten.

Siehe auch Bin, Hex.

Beispiel

Diese Beispiele verwenden die **Oct**-Funktion, um den Oktalwert der angegebenen Zahlen zu ermitteln.

```
'beide Beispiele geben „12“ zurück
dim octWert as text
set octWert to Oct(10)
set octWert to Oct(10.4)
```

Pow(base, exponent)

Gibt die Potenz einer Zahl zurück

Syntax

Ergebnis = Pow(Basis, Exponent)

Parameter	Datentyp	Beschreibung
Basis	number	Die Basiszahl, die mit dem Exponent potenziert werden
Exponent	number	Der Exponent, mit der die Basis potenziert wird.
Ergebnis	number	Basis potenziert mit Exponent

Hinweise

Die **Pow**-Funktion berechnet die Potenz aus Basis und Exponent: Ergebnis = $\text{Basis}^{\text{Exponent}}$.

Siehe auch ^-Operator.

Beispiel

Diese Beispiele verwenden die **Pow**-Funktion, um die Potenz aus den angegebenen Zahlen zu ermitteln.

```
'das Beispiel gibt 16 zurück
dim a as number = Pow(4,2)

'das Beispiel gibt 4 zurück
dim b as number = Pow(16,0.5)
```

Random(rangeMin, rangeMax)

Gibt eine Zufallszahl zurück

Syntax

Ergebnis = Random(Min, Max)

Parameter	Datentyp	Beschreibung
Min	number	Unterer Grenzwert für die Ermittlung der Zufallszahl.
Max	number	Oberer Grenzwert für die Ermittlung der Zufallszahl.
Ergebnis	number	Zufallszahl zwischen Min und Max

Hinweise

Die **Random**-Funktion gibt eine Zufallszahl im Bereich **Min** bis **Max** zurück. Wenn der **Min** oder der **Max** keine ganze Zahl ist, werden die Nachkommastellen abgeschnitten.

Beispiel

Diese Beispiele verwenden die **Random**-Funktion, um eine Zufallszahl in dem angegebenen Bereich zu ermitteln

```
'das Beispiel gibt eine Zufallszahl zwischen 0 und 49 zurück
dim a as number = Random(0,49)

'das Beispiel gibt eine Zufallszahl zwischen -100 und 100 zurück
dim b as number = Random(-100,100)
```

Round(value)

Gibt den gerundeten Wert einer Zahl zurück.

Syntax

Ergebnis = Round(Wert)

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, die gerundet werden soll.
Ergebnis	number	Der gerundete Wert.

Hinweise

Die **Round**-Funktion gibt den übergebenen Wert, auf die ganze Zahl gerundet, zurück. Das Runden erfolgt konform zu den Rundungsregeln, d.h. aufrunden bei ≥ 0.5 und abrunden bei < 0.5.

Beispiel

Diese Beispiele verwenden die **Round**-Funktion, um den angegebenen Wert auf eine ganze Zahl zu runden.

```
'das Beispiel gibt 25 zurück
dim a as number = Round(24.9)

'das Beispiel gibt 24 zurück
dim b as number = Round(24.4)

'das Beispiel gibt -25 zurück
dim c as number = Round(-24.95)

'das Beispiel gibt -24 zurück
dim d as number = Round(-24.45)
```

Sign(value)

Gibt das Vorzeichen einer Zahl zurück.

Syntax

Ergebnis = Sign(Wert)

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, deren Vorzeichen ausgegeben werden soll.
Ergebnis	number	Das Vorzeichen von Wert.

Hinweise

Die **Sign**-Funktion gibt das Vorzeichen für die übergebene Zahl zurück. Liefert **-1**, wenn Wert negativ ist, **0**, wenn Wert Null ist und **1**, wenn Wert positiv ist.

Beispiel

Diese Beispiele verwenden die **Sign**-Funktion, um das Vorzeichen der angegebenen Zahl zu ermitteln.

```
'das Beispiel gibt 1 zurück
dim a as number = Sign(24.9)

'das Beispiel gibt 0 zurück
dim b as number = Sign(0)

'das Beispiel gibt -1 zurück
dim c as number = Sign(-24.9)
```

Sqrt(value)

Gibt die Quadratwurzel einer Zahl zurück.

Syntax

Ergebnis = SqrtWert)

Parameter	Datentyp	Beschreibung
Wert	number	Die Zahl, deren Quadratwurzel berechnet werden soll.
Ergebnis	number	Die Quadratwurzel von Wert

Hinweise

Die **Sqrt**-Funktion gibt die Quadratwurzel für die übergebene Zahl zurück. Die Quadratwurzel für negative Zahlen ist nicht definiert.

Beispiel

```
'das Beispiel gibt 4 zurück
dim a as number = Sqrt(16)

'das Beispiel gibt 0,5 zurück
dim b as number = Sqrt(0.25)

'das Beispiel gibt 10 zurück
dim c as number = Sqrt(100)
```

Textfunktionen

Asc(source)

Gibt den ASCII-Code des ersten Buchstaben eines Textes zurück.

Syntax

Ergebnis = Asc(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Ergebnis	number	Der ASCII-Wert des ersten Zeichens der Quelle.

Hinweise

Die **Asc**-Funktion gibt den Zeichen-Code des ersten Zeichens eines übergebenen Textes zurück.

Zeichen von **0** bis **255** gehören zum ASCII-Zeichensatz. Sie sind auf praktisch jeder Plattform identisch. **Asc** liefert den Zeichen-Code für das Encoding, das dem Text zugrunde liegt. Ist der Text mit MacRoman kodiert, so erhalten Sie den Zeichen-Code entsprechend der MacRoman-Kodierung.

Siehe auch Chr.

Beispiel

```
'das Beispiel gibt 64 für das Zeichen „@“ zurück
dim zeichen as text = "@"
dim ascii as number = Asc(zeichen)

'das Beispiel gibt 49 für das Zeichen „1“ zurück
dim zeichen as text = "1.Zeichen"
dim ascii as number = Asc(zeichen)
```

CountFields(source, separator)

Gibt die Anzahl an Feldern (Werten) des übergebenen Textes zurück, die durch ein angegebenes Trennzeichen getrennt sind.

Syntax **Ergebnis = CountFields(Quelle, Separator)**

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text der geprüft werden soll.
Separator	text	Das Trennzeichen, das die Felder in der Quelle voneinander trennt.
Ergebnis	number	Die Anzahl der Felder in der Quelle, die durch Separator getrennt sind.

Hinweise

Die **CountFields**-Funktion gibt die Anzahl der Felder eines Textes zurück, die durch ein angegebenes Trennzeichen getrennt werden.

Die Funktion **CountFields** wird üblicherweise verwendet, um spaltenweise Daten aus einer Textdatei einzulesen, die durch ein bestimmtes Zeichen voneinander getrennt werden. Wenn der Separator in Source nicht gefunden wird, liefert **CountFields 1**. Wenn Source leer ist, liefert **CountFields 0**.

Siehe auch NthField.

Beispiel

```
'das Beispiel gibt 3 zurück
dim daten as text = "Rot,Grün,Blau"
dim anzahlFelder as number = CountFields(daten, ",")

'das Beispiel gibt 4 zurück, da es das leere „Feld“ nach dem (unnötigen) letzten
Feldbegrenzer mitzählt
dim daten as text = "Rot;Grün;Blau;"
dim anzahlFelder as number = CountFields(daten, ";")
```

FTextToNumber

Gibt die numerische Form eines Textes (gemäss den Landeseinstellungen) zurück.

Syntax **Ergebnis = FTextToNumber(Quelle)**

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text der eine Zahl enthält.
Ergebnis	number	Die numerische Entsprechung des übergebenen Textes.

Hinweise

Die **FTextToNumber**-Funktion konvertiert einen Text in eine Zahl. Die Funktion hört auf, den Text zu untersuchen, sobald es ein Zeichen nicht mehr als Teil einer Zahl erkennt. Alle anderen Zeichen werden automatisch entfernt.

FTextToNumber verwendet zur Umwandlung des Textes in eine Zahl die länderspezifischen Einstellungen des aktuellen Systems (im Gegensatz zur Funktion **TextToNumber).**

Die **FTextToNumber**-Funktion gibt **0** zurück, wenn der Text keine Zahlen enthält.

Siehe auch TextToBoolean, TextToDate.

Beispiel Diese Beispiele verwenden die **FTextToNumber**-Funktion, um die numerische Form eines Textes auszugeben.

```
'das Beispiel gibt 1.5 zurück
dim zahlwort as text = "1,50 Euro"
dim zahl as number = FTextToNumber(zahlwort)
```

Left(source, count)

Gibt für einen Text die ersten n-Zeichen von links beginnend zurück.

Syntax

Ergebnis = Left(Quelle, Anzahl)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text aus dem die Zeichen entnommen werden sollen.
Anzahl	number	Die Anzahl an Zeichen, die aus der Quelle entnommen werden sollen.
Ergebnis	text	Die ersten Anzahl Zeichen von links aus der Quelle

Hinweise

Die **Left**-Funktion gibt von links beginnend die ersten n-Zeichen eines Textes zurück. n ist die Anzahl der Zeichen die entnommen werden sollen. Ist n grösser als die Anzahl an Zeichen im Text, so werden alle Zeichen zurückgeliefert.

Siehe auch Right.

Beispiel

```
'das Beispiel gibt „Buch“ zurück
dim wort as text = "Buchhaltung"
dim wortLinks as text = Left(wort, 4)

'das Beispiel gibt „Kosten“ zurück
dim wort as text = "Kosten"
dim wortLinks as text = Left(wort, 10)
```

Length(source)

Gibt die Anzahl der Zeichen eines Textes zurück.

Syntax

Ergebnis = Length(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Ergebnis	text	Die Anzahl der Zeichen in der Quelle.

Hinweise

Die **Length**-Funktion gibt die Anzahl der Zeichen des übergebenen Textes zurück.

Beispiel

```
'das Beispiel gibt 18 zurück
dim wort as text = "Anzahl der Zeichen"
dim laenge as number = Length(wort)

'das Beispiel gibt 0 zurück
dim wort as text = ""
dim laenge as number = Length(wort)
```

Lower(source)

Wandelt alle alphabetischen Zeichen eines Textes in kleingeschriebene Zeichen um.

Syntax

Ergebnis = Lower(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.

Parameter	Datentyp	Beschreibung
Ergebnis	text	Der Text aus der Quelle, bei dem alle Zeichen kleingeschrieben wurden.

Hinweise

Die **Lower**-Funktion gibt einen Text kleingeschrieben zurück. Um alle Zeichen gross zu schreiben muss die **Upper**-Funktion verwendet werden.

Siehe auch Upper**.**

Beispiel

```
'das Beispiel gibt „gmbh“ zurück
dim wort as text = "GmbH"
dim wortKlein as text = Lower(wort)

'das Beispiel gibt „100 euro“ zurück
dim wort as text = "100 EURO"
dim wortKlein as text = Lower(wort)
```

Ltrim(source)

Gibt einen Text ohne führende Leerzeichen zurück.

Syntax

Ergebnis = LTrim(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text, bei dem die führenden Leerzeichen entfernt werden sollen.
Ergebnis	text	Der Text aus Quelle ohne führende Leerzeichen.

Hinweise

Die **LTrim**-Funktion entfernt die führenden Leerzeichen (von der linken Seite) des übergebenen Textes. Alle Leerzeichen auf der rechten Seite bleiben erhalten. Um die Leerzeichen auf der rechten Seite zu entfernen, muss die **RTrim**-Funktion verwendet werden.

Siehe auch RTrim**,** Trim.

Beispiel

```
'das Beispiel gibt „PLZ“ zurück
dim wort as text = "   PLZ"
dim wortTrim as text = LTrim(wort)

'das Beispiel gibt „Tel.: “ zurück
dim wort as text = "  Tel.: "
dim wortTrim as text = LTrim(wort)
```

Middle(source, start, count)

Gibt einen Ausschnitt aus einem Text zurück.

Syntax

Ergebnis = Middle(Quelle, Start, Laenge)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text, aus dem der Ausschnitt entnommen werden soll.
Start	number	Die Anfangsposition des Ausschnitts.

Parameter	Datentyp	Beschreibung
Laenge	number	Die Anzahl von Zeichen (Länge) des Ausschnitts aus der Quelle.
Ergebnis	text	Der Ausschnitt aus der Quelle, der bei Start beginnt und Laenge Zeichen lang ist.

Hinweise

Die **Middle**-Funktion gibt einen Ausschnitt von dem übergebenen Text zurück. Ist die Startposition grösser als die Anzahl der Zeichen im übergebenen Text, wird ein leerer Text zurückgegeben. Mit der Funktion **Length** kann die Anzahl der Zeichen eines Textes bestimmt werden.

Siehe auch Length.

Beispiel

```
'das Beispiel gibt „Hans“ zurück
dim wort as text = "Name: Hans Löwenzahn"
dim ausschnitt as text = Middle(wort, 7, 4)

'das Beispiel gibt „100“ zurück
dim wort as text = "Kosten: 100 Euro"
dim ausschnitt as text = Middle(wort, 9, 3)
```

NthField(source, separator, index)

Gibt ein Feld aus einem Text zurück.

Syntax

Ergebnis = NthField(Quelle, Separator, Feldnummer)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text, welcher Datenfelder enthält, die durch ein Separatorzeichen voneinander getrennt sind.
Separator	text	Das Zeichen, das die Datenfelder voneinander trennt.
Feldnummer	number	Die Spaltennummer des gewünschten Datenfeldes.
Ergebnis	text	Der gewünschte Wert des Feldes.

Hinweise

Die **NthField**-Funktion gibt den Wert aus einem Text zurück, der vor dem n-ten Auftreten des Trennungszeichens im Text steht. Wenn n ausserhalb des Bereichs liegt, wird ein leerer Text zurückgeliefert. Das erste Feld trägt die Nummer 1. Mit der Funktion **CountField** kann die Anzahl der Felder ermittelt werden.

Siehe auch CountField.

Beispiel

```
'das Beispiel gibt „Grün“ zurück
dim daten as text = "Rot,Grün,Blau"
dim feld as text = NthField(daten, ",", 2)

'das Beispiel gibt einen leeren Text zurück
dim daten as text = "Rot,Grün,Blau"
dim feld as text = NthField(daten, ",", 4)
```

PatternCount(source, search)

Gibt die Anzahl des Auftretens eines Textes in einem anderen Text zurück.

Syntax

Ergebnis = PatternCount(Quelle, Suche)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Suche	text	Der Text, dessen Auftreten in der Quelle gezählt werden
Ergebnis	number	Die Anzahl des Auftretens von Suche in der Quelle.

Hinweise

Die **PatternCount**-Funktion gibt die Anzahl des Auftretens eines Strings innerhalb eines Textes zurück. Wenn der Suchtext nicht im Quelltext vorhanden ist, wird **0** zurückgegeben. Gross- und Kleinschreibung wird bei der Suche nicht berücksichtigt.

Beispiel

```
'das Beispiel gibt „3“ zurück
dim daten as text = "Der Name besteht aus Vorname und Nachname."
dim anzahlText as number = PatternCount(daten, "Name")

'das Beispiel gibt „1“ zurück
dim daten as text = "Der Name besteht aus Vorname und Nachname."
dim anzahlText as number = PatternCount(daten, "Vorname")
```

Position(source, search, start, occurence)

Gibt die Position des Auftretens eines Textes in einem anderen Text zurück.

Syntax

Ergebnis = Position(Quelle, Suche, Start, Vorkommen)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Suche	text	Der Text, dessen Auftreten in der Quelle gesucht werden
Start	number	Position ab der die Suche in der Quelle beginnen soll.
Vorkommen	number	Der Wert des Auftretens von Suche, für das die Position ermittelt werden soll.
Ergebnis	number	Die Position des Vorkommen n-ten Auftretens von Suche in der Quelle.

Hinweise

Die **Position**-Funktion gibt die Position des n-ten Auftretens eines Textes innerhalb eines anderen Textes zurück. Wenn der Suchtext nicht im Quelltext vorhanden ist, wird **0** zurückgegeben. Ist die Startposition grösser als die Anzahl der Zeichen im Quelltext wird **0** zurückgegeben. Ebenso wird 0 zurückgegeben, wenn der Wert für das Auftreten von Suchtext im Quelltext grösser ist als die tatsächliche Anzahl des Vorkommens ab Startposition. Gross- und Kleinschreibung wird bei der Suche nicht berücksichtigt.

Beispiel

```
'Das Beispiel gibt „5“ zurück, für die Position des ersten Vorkommens von „Name“
ab Position 1
dim daten as text = "Der Name besteht aus Vorname und Nachname."
dim posText as number = Position(daten, "Name", 1, 1)

'Das Beispiel gibt „38“ zurück, für die Position des zweiten Vorkommens von Nach-
„Name“ ab Position 20
dim daten as text = "Der Name besteht aus Vorname und Nachname."
dim posText as number = Position(daten, "Name", 20, 2)
```

Proper(source)

Wandelt alle Anfangsbuchstaben der Wörter in einem Text in Grossbuchstaben um.

Syntax

Ergebnis = Proper(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Ergebnis	text	Der Text aus der Quelle, in dem die Anfangsbuchstaben der Wörter grossgeschrieben sind.

Hinweise

Die **Proper**-Funktion gibt einen übergebenen Text zurück, in dem alle Anfangsbuchstaben der Wörter grossgeschrieben sind.

Beispiel

```
'Das Beispiel gibt „Der Name Besteht Aus Vorname Und Nachname.“ zurück.
dim daten as text = "der name besteht aus vorname und nachname."
dim propText as number = Proper(daten)
```

Replace(source, search, replacement)

Ersetzt das erste Vorkommen eines Zeichens oder einer Zeichenkette mit anderen Zeichen.

Syntax

Ergebnis = Replace(Quelle, Suche, Ersetzung)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Suche	text	Die Zeichen, die ersetzt werden soll.
Ersetzung	text	Die ersetzenden Zeichen.
Ergebnis	text	Der Text aus der Quelle, in der das erste Vorkommen von Alttext durch Neutext ersetzt wurde.

Hinweise

Die **Replace**-Funktion ersetzt das erste Vorkommen eines Zeichens oder einer Zeichenkette in einem Text mit anderen Zeichen. Gross- und Kleinschreibung wird bei der **Replace**-Funktion nicht berücksichtigt.

Wenn der Ersetzungstext leer ("") ist, entfernt die Funktion **Replace** das erste Vorkommen von Suchtext in der Quelle. Ist der Ersetzungstext leer (""), so gibt die **Replace**-Funktion den Text aus Quelle unverändert zurück.

Siehe auch ReplaceAll.

Beispiel

```
'das Beispiel gibt „Nass und rutschig ist die Straße.“ zurück (das zweite „ß“ in
„Straße“ wird nicht ersetzt)
dim datenAlt as text = "Naß und rutschig ist die Straße."
dim datenNeu as text = Replace(datenAlt, "ß", "ss")

'das Beispiel gibt „Neuer Text“ zurück
dim datenAlt as text = "Alter Text"
dim datenNeu as text = Replace(datenAlt, "alt", "neu")

'das Beispiel gibt „100 EUR“ zurück
dim datenAlt as text = "1100 EUR"
dim datenNeu as text = Replace(datenAlt, "1", "")
```

ReplaceAll(source, search, replacement)

Ersetzt alle Vorkommen eines Zeichens oder einer Zeichenkette mit anderen Zeichen.

Syntax

Ergebnis = ReplaceAll(Quelle, Suche, Ersetzung)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Suche	text	Die Zeichen, die ersetzt werden sollen.
Ersetzung	text	Die ersetzenden Zeichen.
Ergebnis	text	Der Text aus der Quelle , in der alle Vorkommen von Suche durch Ersetzung ersetzt wurden.

Hinweise

Die **ReplaceAll**-Funktion ersetzt alle Vorkommen eines Zeichens oder einer Zeichenkette in einem Text mit anderen Zeichen. Gross- und Kleinschreibung wird bei der **ReplaceAll**-Funktion nicht berücksichtigt.

Wenn der Ersetzungstext leer ("") ist, entfernt die Funktion **ReplaceAll** alle Vorkommen von Suchtext in der Quelle. Ist der Ersetzungstext leer (""), so gibt die **ReplaceAll**-Funktion den Text aus Quelle unverändert zurück.

Siehe auch Replace.

Beispiel

Hier die Unterschiede leicht ersichtlich zur Replace-Funktion:

```
'das Beispiel gibt „Nass und rutschig ist die Strasse.“ zurück (auch das zweite
„ß“ in „Straße“ wird ersetzt)
dim datenAlt as text = "Naß und rutschig ist die Straße."
dim datenNeu as text = ReplaceAll(datenAlt, "ß", "ss")

'das Beispiel gibt „0 EUR“ zurück
dim datenAlt as text = "110 EUR"
dim datenNeu as text = ReplaceAll(datenAlt, "1", "")
```

Right(source, count)

Gibt für einen Text die ersten n-Zeichen von rechts beginnend zurück.

Syntax

Ergebnis = Right(Quelle, Anzahl)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text aus dem die Zeichen entnommen werden sollen.
Anzahl	number	Die Anzahl an Zeichen, die aus der Quelle entnommen werden sollen.
Ergebnis	text	Die ersten Anzahl Zeichen von rechts aus der Quelle

Hinweise

Die **Right**-Funktion gibt von rechts beginnend die ersten n-Zeichen eines Textes zurück. n ist die Anzahl der Zeichen die entnommen werden sollen. Ist n grösser als die Anzahl an Zeichen im Text, so werden alle Zeichen zurückgeliefert.

Siehe auch Left.

Beispiel

```
'das Beispiel gibt „haltung“ zurück
dim wort as text = "Buchhaltung"
dim wortRechts as text = Right(wort, 7)
```

```
'das Beispiel gibt „Kosten“ zurück
dim wort as text = "Kosten"
dim wortRechts as text = Right(wort, 10)
```

Rtrim(source)

Gibt einen Text ohne nachfolgende Leerzeichen zurück.

Syntax

Ergebnis = RTrim(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text, bei dem die nachfolgenden Leerzeichen entfernt werden sollen.
Ergebnis	text	Der Text aus der Quelle ohne nachfolgende Leerzeichen.

Hinweise

Die **RTrim**-Funktion entfernt die nachfolgenden Leerzeichen (von der rechten Seite) des übergebenen Textes. Alle Leerzeichen auf der linken Seite bleiben erhalten. Um die Leerzeichen auf der linken Seite zu entfernen, muss die **LTrim**-Funktion verwendet werden.

Siehe auch LTrim, Trim.

Beispiel

```
'das Beispiel gibt „PLZ“ zurück
dim wort as text = "PLZ  "
dim wortTrim as text = RTrim(wort)
'das Beispiel gibt „  Tel.:“ zurück
dim wort as text = "  Tel.:    "
dim wortTrim as text = RTrim(wort)       StrComp(source1, source2)
```

Führt einen Vergleich zweier Texte durch.

Syntax

Ergebnis = StrComp(Text1, Text2)

Parameter	Datentyp	Beschreibung
Text1	text	Der erste Vergleichstext.
Text2	text	Der zweite Vergleichstext.
Ergebnis	number	Wenn Text1 < Text2 gibt die Funktion **-1** zurück. Wenn Text1 = Text2 gibt die Funktion **0** zurück. Wenn Text1 > Text2 gibt die Funktion **1** zurück.

Hinweise

Die **StrComp**-Funktion vergleicht zwei Texte miteinander. Der Textvergleich arbeitet lexikographisch, nicht case insensitive. Dies bedeutet, dass die Gross-/Kleinschreibung genau dann bewertet wird, wenn die Texte selbst gleich sind.

Siehe auch =, <>, <, >, <=, >=.

Beispiel

```
'das Beispiel gibt „-1“ zurück
dim wort1 as text = "Tal"
dim wort2 as text = „Talent“
dim vergleich as number = StrComp(wort1, wort2)

'das Beispiel gibt „1“ zurück, da der ASCII-Wert von „u“ größer ist von „U“
dim wort1 as text = "Euro"
dim wort2 as text = „EURO“
dim vergleich as number = StrComp(wort1, wort2)

'das Beispiel gibt „0“ zurück
dim wort1 as text = "gleich"
```

```
dim wort2 as text = „gleich“
dim vergleich as number = StrComp(wort1, wort2)
```

StrComp (Source1, Source2)

Führt einen Vergleich zweier Texte durch.

Syntax

Ergebnis = StrComp(Text1, Text2)

Parameter	Datentyp	Beschreibung
Text1	text	Der erste Vergleichstext.
Text2	text	Der zweite Vergleichstext.
Ergebnis	number	Wenn Text1 < Text2 gibt die Funktion **-1** zurück. Wenn Text1 = Text2 gibt die Funktion **0** zurück. Wenn Text1 > Text2 gibt die Funktion **1** zurück.

Hinweise

Die **StrComp**-Funktion vergleicht zwei Texte miteinander. Der Textvergleich arbeitet lexikographisch, nicht case insensitive. Dies bedeutet, dass die Gross-/Kleinschreibung genau dann bewertet wird, wenn die Texte selbst gleich sind.

Siehe auch =, <>, <, >, <=, >=.

Beispiel

Diese Beispiele verwenden die **StrComp**-Funktion, um zwei Texte miteinander zu vergleichen.

```
'das Beispiel gibt „-1“ zurück
dim wort1 as text = "Tal"
dim wort2 as text = „Talent“
dim vergleich as number = StrComp(wort1, wort2)

'das Beispiel gibt „1“ zurück, da der ASCII-Wert von „u“ grösser ist von „U“
dim wort1 as text = "Euro"
dim wort2 as text = „EURO“
dim vergleich as number = StrComp(wort1, wort2)

'das Beispiel gibt „0“ zurück
dim wort1 as text = "gleich"
dim wort2 as text = „gleich“
dim vergleich as number = StrComp(wort1, wort2)
```

TextToBoolean(source)

Wandelt einen Text in einen Boolean-Wert um.

Syntax

Ergebnis = TextToBoolean(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text der in einen Boolean-Wert umgewandelt werden soll.
Ergebnis	boolean	Der Boolean-Wert von der Quelle.

Hinweise

Die **TextToBoolean**-Funktion konvertiert einen Text in einen Boolean-Wert. Die Funktion gibt **True** zurück, wenn der Text gleich „True“ ist. Ansonsten wird **False** zurückgegeben. Die Gross- und Kleinschreibung wird nicht beachtet.

Siehe auch TextToDate, TextToNumber.

Beispiel

```
'das Beispiel gibt True zurück
dim wort as text = "true"
dim bool as boolean = TextToBoolean(wort)


'das Beispiel gibt False zurück
dim wort as text = "False"
dim bool as boolean = TextToBoolean(wort)

'das Beispiel gibt False zurück
dim wort as text = "wahr"
dim bool as boolean = TextToBoolean(wort)
```

TextToDate(source)

Wandelt einen Text in ein Datumstyp um.

Syntax

Ergebnis = TextToDate(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text (numerische Datumsangabe als Text), der in einen Datumstyp umgewandelt werden soll.
Ergebnis	date	Das Datum von der Quelle als Datumstyp.

Hinweise

Die **TextToDate**-Funktion konvertiert eine numerische Datumsangabe vom Typ **text** in ein Datum vom Typ **date**. Das Datum als Text muss im Format **Tag.Monat.Jahr** vorliegen (z.B. „1.1.2009“, „01.01.2009“, „1.1.09“, oder „01.01.09“). Die Funktion gibt das aktuelle Datum zurück, wenn der Text keine numerische Datumsangabe enthält.

Siehe auch TextToBoolean, TextToNumber.

Beispiel

```
'das Beispiel gibt 01.01.09 zurück
dim datText as text = "1.1.2009"
dim datum as date = TextToDate(datText)

'das Beispiel gibt das aktuelle Datum zurück, da die Datumsangabe im Text ein
'falsches Format besitzt.
dim datText as text = "1.Januar.2009"
dim datum as date = TextToDate(datText)
```

TextToNumber(source)

Gibt die numerische Form eines Textes zurück.

Syntax

Ergebnis = TextToNumber(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text der eine Zahl enthält.
Ergebnis	number	Die numerische Entsprechung des übergebenen Textes.

Hinweise

Die **TextToNumber**-Funktion konvertiert einen Text in eine Zahl. Die Funktion **TextToNumber** hört auf, den Text zu untersuchen, sobald es ein Zeichen nicht mehr als Teil einer Zahl erkennt. Alle anderen Zeichen werden automatisch entfernt.

TextToNumber erkennt die Präfixe „&o“ (oktal), „&b“ (Binär) und „&h“ (hexadezimal). Leerzeichen vor dem "Kaufmanns-Und" sind jedoch nicht erlaubt. Das bedeutet, " &HFF" liefert keinen Wert, "&H FF" liefert jedoch **255**.

Die **TextToNumber**-Funktion gibt **0** zurück, wenn der Text keine Zahlen enthält.

Siehe auch TextToBoolean, TextToDate.

Beispiel

```
'das Beispiel gibt 1.5 zurück
dim zahlwort as text = "1.50 Euro"
dim zahl as number = TextToNumber(zahlwort)

'das Beispiel gibt 0 zurück
dim zahlwort as text = "Tel: 0878874777"
dim zahl as number = TextToNumber(zahlwort)
```

Trim(source)

Gibt einen Text ohne führende und nachfolgende Leerzeichen zurück

Syntax

Ergebnis = Trim(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Der Text, bei dem die führenden und nachfolgenden Leerzeichen entfernt werden sollen.
Ergebnis	text	Der Text aus der Quelle ohne führende und nachfolgende Leerzeichen.

Hinweise

Die **Trim**-Funktion entfernt die führenden und nachfolgenden Leerzeichen des übergebenen Textes.

Siehe auch RTrim, LTrim.

Beispiel

```
'das Beispiel gibt „12345“ zurück
dim wort as text = "  12345  "
dim wortTrim as text = RTrim(wort)
```

Upper(source)

Wandelt alle alphabetischen Zeichen eines Textes in grossgeschriebene Zeichen um.

Syntax

Ergebnis = Upper(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Ergebnis	text	Der Text aus der Quelle, bei dem alle Zeichen grossgeschrieben wurden.

Hinweise

Die **Upper**-Funktion gibt einen Text grossgeschrieben zurück. Um alle Zeichen klein zu schreiben muss die **Lower**-Funktion verwendet werden.

Siehe auch Lower.

Beispiel

```
'das Beispiel gibt „GMBH“ zurück
dim wort as text = "GmbH"
dim wortGross as text = Upper(wort)

'das Beispiel gibt „100 EURO“ zurück
dim wort as text = "100 euro"
dim wortGross as text = Upper(wort)
```

Farbfunktionen

CMYColor(Cyan, Magenta, Yellow)

Erzeugt eine Farbe, die auf dem CMY-Farbmodell (Cyan, Magenta, Yellow) basiert.

Syntax

Ergebnis = CMYColor(Cyan, Magenta, Yellow)

Parameter	Datentyp	Beschreibung
Cyan	number	Der Wert von Cyan in der Farbe
Magenta	number	Der Wert von Magenta in der Farbe
Yellow	number	Der Wert von Yellow in der Farbe
Ergebnis	color	Ergebnis ist ein Objekt, das die Farbe repräsentiert, die sich aus den angegebenen Werten für Cyan, Magenta und Yellow ergibt.

Hinweise

Die **CMYColor**-Funktion erzeugt eine Farbe, die auf den Werten für Cyan, Magenta und Yellow basiert. Die angegebenen Werte müssen zwischen **0** und **1** liegen.

In der folgenden Tabelle sind die CMY-Farbwerte für verschiedene Farben dargestellt.

Farbe	Cyan-Wert	Magenta-Wert	Yello-Wert
Weiss	0.0	0.0	0.0
Cyan	1.0	0.0	0.0
Magenta	0.0	1.0	0.0
Gelb	0.0	0.0	1.0
Rot	0.0	1.0	1.0
Grün	1.0	0.0	1.0
Blau	1.0	1.0	0.0
Grau	0.5	0.5	0.5
Schwarz	1.0	1.0	1.0

Sie können ebenso entweder die **RGBColor-** oder **HSVColor**-Funktion verwenden, um eine Farbe zuzuweisen.

Siehe auch SetDocument, BeginDocument, RGBColor, HSVColor.

Beispiel

```
'Das Beispiel erzeugt ein reines Grau im CMY-Modell. Die ausgewählte Farbe wird
als Textfarbe definiert.
dim farbeText as color = CMYColor(0.5, 0.5, 0.5)
SetTextColor(farbeText)
```

HSVColor(hue, saturation, value)

Erzeugt eine Farbe, die auf dem HSV-Farbmodell (hue, saturation, value – Farbton, Sättigung, Helligkeit) basiert.

Syntax

Ergebnis = HSVColor(Farbton, Saettigung, Helligkeit)

Parameter	Datentyp	Beschreibung
Farbton	number	Der Farbton der Farbe
Saettigung	number	Die Sättigung der Farbe
Helligkeit	number	Die Helligkeit der Farbe
Ergebnis	color	Ergebnis ist ein Objekt, das die Farbe repräsentiert, die sich aus den angegebenen Werten für Farbton, Sättigung und Helligkeit ergibt.

Hinweise

Die **HSVColor**-Funktion erzeugt eine Farbe, die auf den Werten für Farbton, Sättigung und Helligkeit basiert. Die angegebenen Werte müssen zwischen **0** und **1** liegen.

In der folgenden Tabelle sind die HSV-Farbwerte für verschiedene Farben dargestellt.

Farbe	Farbton	Sättigung	Helligkeit
Weiss	0.0	0.0	1.0
Cyan	0.5	1.0	1.0
Magenta	0.833	1.0	0.0
Gelb	0.167	1.0	1.0
Rot	0.0	1.0	1.0
Grün	0.333	1.0	1.0
Blau	0.667	1.0	0.0
Grau	0.0	0.0	0.5
Schwarz	0.0	0.0	0.0

Sie können ebenso entweder die **RGBColor-** oder **CMYColor**-Funktion verwenden, um eine Farbe zuzuweisen.

Siehe auch SetDocument, BeginDocument, RGBColor, CMYColor.

Beispiel

```
'Das Beispiel erzeugt die Farbe Cyan im HSV-Modell. Die ausgewählte Farbe wird als
Textfarbe definiert.
dim farbeText as color = HSVColor(0.5, 1.0, 1.0)
SetTextColor(farbeText)
```

RGBColor(red, green, blue)

Erzeugt eine Farbe, die auf dem RGB-Farbmodell (rot, grün, blau) basiert.

Syntax

Ergebnis = RGBColor(Rot, Gruen, Blau)

Parameter	Datentyp	Beschreibung
Rot	number	Der Rot-Anteil der Farbe
Gruen	number	Die Grün-Anteil der Farbe
Blau	number	Die Blau-Anteil der Farbe
Ergebnis	color	Ergebnis ist ein Objekt, das die Farbe repräsentiert, die sich aus den Farbanteilen Rot, Grün und Blau ergibt.

Hinweise

Die **RGBColor**-Funktion erzeugt eine Farbe, die auf den Werten für Rot, Grün und Blau basiert. Die angegebenen Werte müssen zwischen **0** und **255** liegen.

In der folgenden Tabelle sind die RGB-Farbwerte für verschiedene Farben dargestellt.

Farbe	Rot	Grün	Blau
Weiss	255	255	255
Cyan	0	255	255
Magenta	255	0	255
Gelb	255	255	0
Rot	255	0	0
Grün	0	255	0
Blau	0	0	255
Grau	128	128	128
Schwarz	0	0	0

Sie können ebenso entweder die **CMYColor-** oder **HSVColor**-Funktion verwenden, um eine Farbe zuzuweisen.

Siehe auch SetDocument, BeginDocument, CMYColor, HSVColor.

Beispiel

```
'Das Beispiel erzeugt die Farbe Gelb im RGB-Modell. Die ausgewählte Farbe wird als
Textfarbe definiert.
dim farbeText as color = RGBColor(255, 255, 0)
SetTextColor(farbeText)
```

Sonstige

GetTextWidth(text, result)

Wird verwendet, um die Länge eines Textes zu ermitteln.

Syntax

GetTextWidth(Text, VarName)

Parameter	Datentyp	Beschreibung
Text	text	Der Text, dessen Länge ermittelt werden soll.

Parameter	Datentyp	Beschreibung
VarName	text	Name der Variable, die den Wert für die Länge übergeben bekommt.

Hinweise

Die **GetTextWidth**-Anweisung ermittelt die Länge eines Textes. Dadurch lassen sich zum Beispiel Texte mehrerer **DrawText**-Anweisungen hintereinander ausgeben und differenziert formatieren. Bei dem zweiten Parameter **VarName** ist zu beachten, dass **nicht** die Variable selbst angegeben wird, sondern der Name der Variable als String. Beispiel: Die Länge des Textes „Text“ soll der Variable *laenge* übergeben werden. Die Anweisung sieht dann wie folgt aus: *GetTextWidth(„Text“, „laenge“)* und nicht *GetTextWidth(„Text“, laenge)*

Siehe auch DrawText.

Beispiel

Dieses Beispiel verwendet die **GetTextWidth**-Anweisung, um mehrere DrawText-Anweisungen hintereinander auszugeben.

```
'das Beispiel ermittelt die Länge eines Textes und gibt einen weiteren Text fett-
gedruckt dahinter aus
#BeginGraphicsText
SetArea(1000, AREA_AUTOSIZE)
dim t1 as text = "Hallo" 'erster Text
dim t2 as text = "Welt" ' zweiter Text
dim l as number 'Variable für die Länge

GetTextWidth(t1, "l")

DrawText(t1, 10, 10, 200, 80) 'Ausgabe des ersten Textes
setTextBold(true)
DrawText(t2, 10 + l + 10, 10, 200, 80) 'Ausgabe des zweiten Textes hinter dem ers-
ten Text
setTextBold(false)
```

Kapitel 5 Operatoren zur Programmsteuerung

Mit den Operatoren und Anweisungen erstellen Sie Berechnungen und prüfen Bedingungen. Dadurch kann der Programmablauf gesteuert und die Ausgaben angepasst werden.

Operatoren

()

Wird verwendet um einen Ausdruck zusammenzufassen.

Syntax

Ergebnis = (Ausdruck)

Parameter	Datentyp	Beschreibung
Ausdruck	diverse	Ein beliebiger Ausdruck.
Ergebnis	diverse	Der Ausdruck in Klammern gesetzt.

Hinweise

Der **()**-Operator wird verwendet, um einen Ausdruck in Klammern zu setzen. Der Ausdruck kann ein beliebiger Datentyp sein.

Beispiel

```
'das Beispiel nutzt die Umklammerung, um zuerst die Addition und dann die Multi-
plikation auszuführen
dim a as number = (1+2)*3
```

NOT

Wird verwendet, um einen boole'schen Ausdruck zu negieren.

Syntax

Ergebnis = NOT Bedingung

Parameter	Datentyp	Beschreibung
Bedingung	boolean	Boole'scher Ausdruck, der negiert werden soll.
Ergebnis	boolean	Der negierte boole'sche Ausdruck.

Hinweise

Der **NOT**-Operator wird verwendet, um einen boole'schen Ausdruck zu negieren.

Beispiel

```
'das Beispiel gibt False zurück, da die Bedingung „anzahl > 0“ True liefert
dim anzahl as number = 100
dim vergleich as boolean = NOT anzahl > 0
```

^

Wird verwendet, um zwei Zahlen zu potenzieren.

Syntax

Ergebnis = Wert1 ^ Wert2

Parameter	Datentyp	Beschreibung
Wert1	number	Die Basiszahl, die mit Wert2 potenziert werden soll.
Wert2	number	Der Exponent, mit der Wert1 potenziert wird.
Ergebnis	number	Wert1 potenziert mit Wert2.

Hinweise

Der ^ Operator berechnet die Potenz aus Wert1 und Wert2: Ergebnis = $Wert1^{Wert2}$.

Siehe auch Pow-Funktion.

Beispiel

```
'das Beispiel gibt 16 zurück
dim a as number = 4 ^ 2

'das Beispiel gibt 4 zurück
dim b as number = 16 ^ 0.5
```

*

Wird verwendet, um zwei Zahlen zu multiplizieren oder eine Zeichenkette zu wiederholen.

Syntax

Der * Operator hat zwei mögliche Syntax-Formen

Ergebnis = Wert1 * Wert2

Parameter	Datentyp	Beschreibung
Wert1	number	Ein beliebiger numerischer Wert.
Wert2	number	Ein beliebiger numerischer Wert.
Ergebnis	number	Das Produkt aus Wert1 und Wert2.

oder

Parameter	Datentyp	Beschreibung
Wert1	text/number	Eine beliebige Zeichenkette/Zahl.
Wert2	number/text	Eine beliebige Zahl/Zeichenkette.
Ergebnis	text	Wert1/Wert2 der Wert2/Wert1 mal wiederholt wurde.

Hinweise

Der * Operator berechnet das Produkt aus Wert1 und Wert2 oder wiederholt eine Zeichenkette mit der angegebenen Anzahl. Ist Wert1 eine Zeichenkette, muss Wert2 immer eine Zahl sein und umgekehrt. Für den * Operator gilt die Kommutativität (Vertauschung der Argumente).

Beispiel

Das erste Beispiel verwendet den * Operator, um die angegebenen Zahlen zu multiplizieren. Das zweite Beispiel um eine Zeichenkette zu wiederholen.

```
'das Beispiel gibt -8 zurück
dim a as number = 16 * -0.5
```

```
'das Beispiel gibt für b und c „3-mal 3-mal 3-mal“ zurück
dim b as number = 3 * "3-mal "
dim c as number = "3-mal " * 3
```

/

Wird verwendet, um eine Fliesskomma-Division zweier Zahlen durchzuführen.

Syntax

Ergebnis = Wert1 / Wert2

Parameter	Datentyp	Beschreibung
Wert1	number	Der Dividend, ein beliebiger numerischer Wert.
Wert2	number	Der Divisor, ein beliebiger numerischer Wert.
Ergebnis	number	Das Ergebnis der Division von Wert1 und Wert2.

Hinweise

Der / Operator berechnet den Quotienten aus Wert1 und Wert2. Bei einer Division durch Null erscheint die Fehlermeldung „Division by zero“.

Verwenden Sie diesen Operator, wenn Sie eine Division benötigen, die den Nachkommateil berücksichtigt.

Siehe auch Div.

Beispiel

```
'das Beispiel gibt 0.125 zurück
dim a as number = 1 / 8

'das Beispiel gibt -32 zurück
dim b as number = -16 / 0.5
```

+

Wird verwendet, um zwei Zahlen zu addieren oder zwei Zeichenketten aneinander zu hängen.

Syntax

Der **+** Operator hat zwei mögliche Syntax-Formen.

Ergebnis = Wert1 + Wert2

Parameter	Datentyp	Beschreibung
Wert1	number	Ein beliebiger numerischer Wert.
Wert2	number	Ein beliebiger numerischer Wert.
Ergebnis	number	Die Summe von Wert1 und Wert2.

oder

Parameter	Datentyp	Beschreibung
Wert1	text	Eine beliebige Zeichenkette.
Wert2	text	Eine beliebige Zeichenkette.
Ergebnis	text	Wert2 ist an Wert1 angehängt.

Hinweise

Der **+** Operator berechnet die Summe aus Wert1 und Wert2 oder verknüpft zwei Zeichenketten miteinander.

Beispiel

Das erste Beispiel verwendet den **+** Operator, um die angegebenen Zahlen zu addieren, das zweite Beispiel verknüpft zwei Zeichenketten miteinander.

```
'das Beispiel gibt 20 zurück
dim a as number = 16 + 4
'das Beispiel gibt 10 zurück
dim b as number = -14.5 + 4.5

'das Beispiel gibt „ShakeHands“ zurück
dim c as text = "Shake" + "Hands"
```

–

Wird verwendet, um zwei Zahlen zu subtrahieren.

Syntax

Ergebnis = Wert1 – Wert2

Parameter	Datentyp	Beschreibung
Wert1	number	Ein beliebiger numerischer Wert.
Wert2	number	Ein beliebiger numerischer Wert.
Ergebnis	number	Die Differenz von Wert1 und Wert2.

Hinweise

Der **–** Operator berechnet die Differenz aus Wert1 und Wert2.

Beispiel

```
'das Beispiel gibt -10 zurück
dim a as number = 10 - 20

'das Beispiel gibt -20 zurück
dim b as number = -14.5 - 5.5
```

=

Wird verwendet, um zwei Ausdrücke auf Gleichheit zu prüfen.

Syntax

Ergebnis = Wert1 = Wert2

Parameter	Datentyp	Beschreibung
Wert1	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Ergebnis	boolean	Ergibt **True**, wenn Wert1 gleich Wert2 ist, sonst **False**.

Hinweise

Der **=** Operator überprüft Wert1 und Wert2 auf Gleichheit. Die Datentypen von Wert1 und Wert2 müssen zueinander passen. Werte vom Datentyp **text**, **number**, **date** oder **boolean** können verwendet werden.

Bei dem Vergleich von Zeichenketten, wird die Gross- und Kleinschreibung nicht berücksichtigt. Wenn Zeichenketten unter Berücksichtigung der Gross- und Kleinschreibung verglichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

```
'das Beispiel gibt False zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl = 0)

'das Beispiel gibt True zurück
dim wort as text = "wahr"
dim vergleich as boolean = (wort = "WAHR")
```

<

Wird verwendet, um festzustellen, ob ein alphabetischer oder quantitativer Ausdruck kleiner als ein anderer ist.

Syntax

Ergebnis = Wert1 < Wert2

Parameter	Datentyp	Beschreibung
Wert1	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Ergebnis	boolean	Ergibt **True**, wenn Wert1 kleiner Wert2 ist, sonst **False**.

Hinweise

Der **<** Operator überprüft, ob Wert1 kleiner als Wert2 ist. Die Datentypen von Wert1 und Wert2 müssen zueinander passen. Werte vom Datentyp **text**, **number** oder **date** können verwendet werden.

Bei dem Vergleich von Zeichenketten, wird die Gross- und Kleinschreibung nicht berücksichtigt. Wenn Zeichenketten unter Berücksichtigung der Gross- und Kleinschreibung verglichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

```
'das Beispiel gibt True zurück
dim zahl as number = 10
dim vergleich as boolean = (0 < zahl)

'das Beispiel gibt True zurück
dim wort as text = "A"
dim vergleich as boolean = (wort < "B")
```

<=

Wird verwendet, um festzustellen, ob ein alphabetischer oder quantitativer Ausdruck kleiner oder gleich einem Anderen ist.

Syntax

Ergebnis = Wert1 <= Wert2

Parameter	Datentyp	Beschreibung
Wert1	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Ergebnis	boolean	Ergibt **True**, wenn Wert1 kleiner oder gleich Wert2 ist, sonst **False**.

Hinweise

Der **<=** Operator überprüft, ob Wert1 kleiner oder gleich Wert2 ist. Die Datentypen von Wert1 und Wert2 müssen zueinander passen. Werte vom Datentyp **text**, **number** oder **date** können verwendet werden.

Bei dem Vergleich von Zeichenketten, wird die Gross- und Kleinschreibung nicht berücksichtigt. Wenn Zeichenketten unter Berücksichtigung der Gross- und Kleinschreibung verglichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

Diese Beispiele verwenden den **<=** Operator, um zu überprüfen, ob ein Ausdruck kleiner oder gleich einem Anderen ist.

```
'das Beispiel gibt False zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl <= 0)

'das Beispiel gibt False zurück
dim wort as text = "AA"
dim vergleich as boolean = (wort <= "A")
```

>

Wird verwendet, um festzustellen, ob ein alphabetischer oder quantitativer Ausdruck grösser als ein anderer ist.

Syntax

Ergebnis = Wert1 > Wert2

Parameter	Datentyp	Beschreibung
Wert1	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Ergebnis	boolean	Ergibt **True**, wenn Wert1 grösser Wert2 ist, sonst **False**.

Hinweise

Der **>** Operator überprüft, ob Wert1 grösser als Wert2 ist. Die Datentypen von Wert1 und Wert2 müssen zueinander passen. Werte vom Datentyp **text**, **number** oder **date** können verwendet werden.

Bei dem Vergleich von Zeichenketten, wird die Gross- und Kleinschreibung nicht berücksichtigt. Wenn Zeichenketten unter Berücksichtigung der Gross- und Kleinschreibung verglichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

```
'das Beispiel gibt True zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl > 0)

'das Beispiel gibt False zurück
dim wort as text = "A"
dim vergleich as boolean = (wort > "B")
```

>=

Wird verwendet, um festzustellen, ob ein alphabetischer oder quantitativer Ausdruck grösser oder gleich einem Anderen ist.

Syntax

Ergebnis = Wert1 >= Wert2

Parameter	Datentyp	Beschreibung
Wert1	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Ergebnis	boolean	Ergibt **True**, wenn Wert1 grösser oder gleich Wert2 ist, sonst **False**.

Hinweise

Der **>=** Operator überprüft, ob Wert1 grösser oder gleich Wert2 ist. Die Datentypen von Wert1 und Wert2 müssen zueinander passen. Werte vom Datentyp **text**, **number** oder **date** können verwendet werden.

Bei dem Vergleich von Zeichenketten, wird die Gross- und Kleinschreibung nicht berücksichtigt. Wenn Zeichenketten unter Berücksichtigung der Gross- und Kleinschreibung verglichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

```
'das Beispiel gibt True zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl >= 0)

'das Beispiel gibt True zurück
dim wort as text = "AA"
dim vergleich as boolean = (wort >= "A")
```

<>

Wird verwendet, um festzustellen, ob ein alphabetischer oder quantitativer Ausdruck ungleich einem anderen ist.

Syntax

Ergebnis = Wert1 <> Wert2

Parameter	Datentyp	Beschreibung
Wert1	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	diverse	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Ergebnis	boolean	Ergibt **True**, wenn Wert1 ungleich Wert2 ist, sonst **False**.

Hinweise

Der **<>** Operator überprüft, ob Wert1 ungleich Wert2 ist. Die Datentypen von Wert1 und Wert2 müssen zueinander passen. Werte vom Datentyp **text**, **number**, **date** oder **boolean** können verwendet werden.

Bei dem Vergleich von Zeichenketten, wird die Gross- und Kleinschreibung nicht berücksichtigt. Wenn Zeichenketten unter Berücksichtigung der Gross- und Kleinschreibung verglichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

```
'das Beispiel gibt False zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl <> 10)

'das Beispiel gibt True zurück
dim wort as text = "AB"
dim vergleich as boolean = (wort <> "A")
```

AND

Wird verwendet, um einen logischen Vergleich mit „UND“ für zwei boole’sche Ausdrücke vorzunehmen.

Syntax

Ergebnis = Bedingung1 AND Bedingung2

Parameter	Datentyp	Beschreibung
Bedingung1	boolean	Ein beliebiger boole’scher Ausdruck.
Bedingung2	boolean	Ein beliebiger boole’scher Ausdruck.
Ergebnis	boolean	Ein boole’scher Wert.

Hinweise

Der **AND**-Operator führt ein logisches „UND“ aus.

Siehe auch OR, XOR.

Folgende Ergebnisse sind für den **AND**-Operator möglich:

Bedingung1	Bedingung2	AND
True	True	True
True	False	False
False	True	False
False	False	False

Beispiel

Diese Beispiele verwenden den **AND**-Operator, um einen logischen Vergleich mit „UND“ für zwei boole’sche Ausdrücke vorzunehmen.

```
'das Beispiel gibt False zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl > 10) AND (zahl < 20)

'das Beispiel gibt True zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl >= 10) AND (zahl < 20)
```

OR

Wird verwendet, um einen logischen Vergleich mit „ODER“ für zwei boole’sche Ausdrücke vorzunehmen.

Syntax

Ergebnis = Bedingung1 OR Bedingung2

Parameter	Datentyp	Beschreibung
Bedingung1	boolean	Ein beliebiger boole’scher Ausdruck.
Bedingung2	boolean	Ein beliebiger boole’scher Ausdruck.
Ergebnis	boolean	Ein boole’scher Wert.

Hinweise

Der **OR**-Operator führt ein logisches „ODER“ aus.

Siehe auch AND, XOR.

Folgende Ergebnisse sind für den **OR**-Operator möglich:

Bedingung1	Bedingung2	OR
True	True	True
True	False	True
False	True	True
False	False	False

Beispiel

Diese Beispiele verwenden den **OR**-Operator, um einen logischen Vergleich mit „ODER“ für zwei boole’sche Ausdrücke vorzunehmen.

```
'das Beispiel gibt True zurück
dim zahl as number = 50
dim vergleich as boolean = (zahl < 10) OR (zahl > 20)

'das Beispiel gibt True zurück
dim zahl as number = 0
dim vergleich as boolean = (zahl < 10) OR (zahl > 20)

'das Beispiel gibt False zurück
dim zahl as number = 15
dim vergleich as boolean = (zahl < 10) OR (zahl > 20)
```

XOR

Wird verwendet, um einen logischen Vergleich mit „XODER“ für zwei boole’sche Ausdrücke vorzunehmen.

Syntax

Ergebnis = Bedingung1 XOR Bedingung2

Parameter	Datentyp	Beschreibung
Bedingung1	boolean	Ein beliebiger boole’scher Ausdruck.
Bedingung2	boolean	Ein beliebiger boole’scher Ausdruck.
Ergebnis	boolean	Ein boole’scher Wert.

Hinweise

Der **XOR**-Operator führt ein logisches „XODER“ aus.

Siehe auch AND, OR.

Folgende Ergebnisse sind für den **XOR**-Operator möglich:

Bedingung1	Bedingung2	XOR
True	True	False
True	False	True
False	True	True
False	False	False

Beispiel

Diese Beispiele verwenden den **XOR**-Operator, um einen logischen Vergleich mit „XODER“ für zwei boole’sche Ausdrücke vorzunehmen.

```
'das Beispiel gibt False zurück
dim zahl as number = 5
dim vergleich as boolean = (zahl < 10) XOR (zahl > 0)
```

```
'das Beispiel gibt True zurück
dim zahl as number = 10
dim vergleich as boolean = (zahl <= 10) XOR (zahl > 20)

'das Beispiel gibt False zurück
dim zahl as number = 15
dim vergleich as boolean = (zahl < 10) XOR (zahl > 20)
```

Kapitel 6 Support

Nicht immer funktioniert Software so reibungslos, wie der Anwender dies erwartet. Aber auch in diesem Fall versuchen wir Ihnen so schnell wie möglich weiterzuhelfen. Der ShakeHands-Support ist sehr preiswert, Voraussetzung für die Inanspruchnahme ist aber die vorher erfolgte Registrierung als Anwender für das betreffende Produkt. Wenn Sie ein Produkt direkt bei uns erworben haben, entfällt diese zusätzliche Registrierung selbstverständlich.

Es kann sein, dass die vorliegenden Ausführungen technisch sind und Sie doch auf Hilfe zur Entwicklung eines neuen Formulars angewiesen sind. Gerne offerieren wir Ihnen die Erstellung eines massgeschneiderten Formulars. Senden Sie uns auf der Grundlage einer Vorlage Ihre Wünsche und wir programmieren Ihnen den Report.

Bevor Sie den Support beanspruchen, versuchen Sie das Problem bitte mit Hilfe der entsprechenden Dokumentation zu lösen (auch ReadMe-Texte sowie evtl. Installations- oder Updateanleitungen). Bitte versuchen Sie vor Kontaktaufnahme das Problem zu reproduzieren und die genaue Art und Weise des Zustandekommens (unter welchen Bedingungen) zu beschreiben. Überlegen Sie sich auch, ob Sie vor Auftreten des Problems Änderungen an Ihrer Hard- oder Softwarekonfiguration vorgenommen haben.

Support/Service	Erreichbar über
Produktregistrierung/Supportanfrage/Feedback	Internet: www.shakehands.com Email: support-de@shakehands.com
Hotline	Telefon Hotline/Fernwartung Schweiz: **0900 57 52 38 (CHF 2.50 pro Minute)** Fax Hotline Schweiz: **034 495 70 25**
Technischer Support per Email und Reparatur Service	Supportanfragen via Email verrechnen wir in Viertelstunden-Takten. Die erste Viertelstunde ist gleich auch die Grundtaxe unabhängig, ob die Anfrage weniger als eine Viertelstunde in Anspruch nimmt. Falls Ihre Buchhaltung repariert werden muss und es sich nicht um einen Programmfehler handelt, können Sie defekte Mandanten-Daten an unsere Support-Abteilung senden.
Technischer Support per Fernzugriff	Support via Fernzugriff auf Ihren lokalen Rechner bieten wir mit Teamviewer an. Laden Sie die Zugriffssoftware ab unserer Partnerseite (Anleitung auf unseren Webseiten unter Support beachten) und rufen Sie uns via Hotline an und melden Sie uns ID-Nummer und Passwort. Wir greifen dann direkt auf Ihren Rechner zu. Wir rechnen über die Hotlinegebühr ab.
Technischer Support per Vororteinsatz	Für technische Probleme, die sich nicht telefonisch lösen lassen, fordern Sie unseren Servicemitarbeiter für einen einen Vor-Ort-Service an.
Anwenderkurse	Programme: ShakeHands Buchhaltungen, ShakeHands ERP-Lösungen Schulungsort: Bern, Yverdon Kursdauer: 4 Stunden Teilnehmerzahl: 2 bis 4 Personen

Support/Service	Erreichbar über
Einzelschulung nach Mass	Programme: ShakeHands Buchhaltungen, ShakeHands ERP-Lösungen Schulungsort: in Ihrem Unternehmen Kursdauer: ein halber Tag Teilnehmerzahl: bis 4 MitarbeiterInnen
Kostenlose Dienstleistungen Handbücher	Handbücher: Der Einsteiger findet in unseren Anwenderhandbüchern Hilfe, wie auch Wissen um die doppelte Buchführung im kostenlosen Ratgeber ‚Buchführungsgrundlagen. Der Profi und die Buchhalterin findet in Checklisten und Formularhandbüchern wertvolle Tipps und Tricks. Versuchen Sie bitte mit Hilfe der Handbücher und Dokumentationen das Problem zu lösen. Sie sind in allen Produkten als PDF enthalten oder unter unseren Downloads frei verfügbar. http://www.shakehands.com/de/download/index.html
Kostenlose Dienstleistungen FaQ	FaQ: Fragen und Antworten von allgemeinen Standardauskünften finden Sie in unserer FaQ-Datenbank. http://www.shakehands.com/de/faq/index.html
Kostenlose Dienstleistungen: Forum	Forum: Für unsere Buchhaltungsprodukte führen wir gemeinsam mit Deutschland und Oesterreich ein AnwenderInnen-Forum. In den Bereichen Technik und Anwendungen finden Sie informative Einträge und können da eigene Fragen stellen. Der Bereich Buchhaltung und Steuern werden wir nach Nachfrage 2011 als eigene Sektion implementieren. http://www.monkey-office.de/forum/index.php