Handbuch Sprachreferenz für den Generator

ShakeHands Kontor 2012
ShakeHands Conto 2012
ShakeHands Faktura 2012
ShakeHands Budget 2012
ShakeHands Reisekosten 2012

Inhalt

Impressum

Handbuch Sprachreferenz 2012
Ausgabe 9.2 (September 2012)

Kontakt

ShakeHands Software Ltd
Sägerei Kröschenbrunnen
CH - 3555 Trubschachen
Telefon: 0878 87 47 77
Email: ch@shakehands.com
Internet: http://www.shakehands.com

Kapitel 1 Einführung

Themenorientiert informieren wir Sie hier über die Funktionen ‚Sprachreferenz' unserer ShakeHands Produkte. Mit diesem Buch skizzieren wir den Ausschnitt des Formulargenerators. Kombinieren Sie je nach Wunsch die Module, insbesondere ‚Formularentwicklung'. Das komplette Anwenderhandbuch finden Sie ebenfalls, wie auch weitere Themen Handbücher unter Hilfe und Dokumentationen.

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'.

Alle Ausführungen sind funktionsgleich sowohl für Mac OS X 10.6, 10.7 und 10.8 als auch für Windows Vista und 7 sowie 8 erhältlich. Detaillierte tabellarische Funktionsvergleiche finden Sie auf der Webseite von ShakeHands Software Ltd unter www.shakehands.com.

In diesem Handbuch wird im Allgemeinen nur der Begriff **ShakeHands Kontor** 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 Modul Anlagenverwaltung** und viel Erfolg.

Ihr ShakeHands-Team

Zitat von Jeff Atwood,

"In software, we rarely have meaningful requirements.

Even if we do, the only measure of success that matters is

whether our solution solves the customer's shifting idea of what their problem is."

P.S. Wir bemühen uns diesem Leitsatz zu folgen. Es braucht immer beide!

Kapitel 2 Anweisungen zur Formularsteuerung

Nachfolgend werden die möglichen Anweisungen zur Formularsteuerung beschrieben.

Mit den Operatoren und Anweisungen können beispielsweise Berechnungen durchgeführt und Bedingungen geprüft werden. Dadurch kann der Programmablauf gesteuert und die Ausgaben angepasst werden.
Nachfolgend werden die möglichen Operatoren und Anweisungen zur Programmsteuerung beschrieben.

Operatoren

()

Wird verwendet um einen Ausdruck zu umklammern.

Syntax

Ergebnis = (Ausdruck)

Parameter	Datentyp	Beschreibung
Ausdruck	variant	Ein beliebiger Ausdruck.
Ergebnis	variant	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

Dieses Beispiel verwendet den **()**-Operator, um einen Ausdruck in Klammern zu setzen.

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

^

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

Diese Beispiele verwenden den **^** Operator, um die Potenz aus den angegebenen Zahlen zu ermitteln.

```
'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

Diese Beispiele verwenden den * Operator, um die angegebenen Zahlen zu multiplizieren und 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

Diese Beispiele verwenden den / Operator, um die angegebenen Zahlen zu dividieren.

```
'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

Diese Beispiele verwenden den **+** Operator, um die angegebenen Zahlen zu addieren und um zwei Zeichenketten miteinander zu verknüpfen.

```
'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 „ProSaldo“ zurück
dim c as text = "Pro" + "Saldo"
```

–

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 Diese Beispiele verwenden den – Operator, um die angegebenen Zahlen zu subtrahieren.

```
'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	variant	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	variant	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 vergleichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel Diese Beispiele verwenden den = Operator, um Vergleiche durchzuführen.

```
'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	variant	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	variant	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 vergleichen 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 ist als ein anderer.

```
'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	variant	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	variant	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 vergleichen 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	variant	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	variant	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 vergleichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

Diese Beispiele verwenden den **>** Operator, um zu überprüfen, ob ein Ausdruck grösser ist als ein anderer.

```
'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	variant	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	variant	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 vergleichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

Diese Beispiele verwenden den **>=** Operator, um zu überprüfen, ob ein Ausdruck grösser oder gleich einem anderen ist.

```
'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	variant	Ein beliebiger alphabetischer oder quantitativer Ausdruck.
Wert2	variant	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 vergleichen werden sollen, muss die **StrComp**-Funktion verwendet werden.

Siehe auch StrComp-Funktion.

Beispiel

Diese Beispiele verwenden den **<>** Operator, um zu überprüfen, ob ein Ausdruck ungleich einem anderen ist.

```
'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 einen logisches „UND“ aus.

Siehe auch OR, XOR.

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

Bedingung1	Bedingung2	AND
True	True	True

Bedingung1	Bedingung2	AND
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)
```

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

Dieses Beispiel verwendet den **NOT**-Operator, um das Ergebnis einer Bedingung zu negieren.

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

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

Bedingung1	Bedingung2	OR
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)
```

Anweisungen

#include

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 werden soll. Das Script kann dann mit der **#include**-Anweisung in jedes beliebige andere Formular-Script eingebunden werden.

Siehe auch Handbuch Formularentwicklung (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 „Include Formular“ ein
#include "Include Formular"
```

' (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
```

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

Dieses Beispiel verwendet die **beep**-Anweisung, um bei einem boole'schen Vergleich für ein bestimmtes Ergebnis einen Warnton auszugeben.

```
'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
```

---

const

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

Diese Beispiele verwenden die **const**-Anweisung, um eine Konstante zu definieren.

```
'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

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

Diese Beispiele verwenden die **dim**-Anweisung, um eine Variable zu definieren.

```
'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 ein Variable vom Typ boolean ohne Wertzuweisung
dim vergleich as boolean
```

---

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 Diese Beispiele verwenden die **exit**-Anweisung, um einen Bereich zu verlassen.

```
'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"
```

---

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 zuvorige 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 folgt

eine **else**-Anweisung, 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** folgenden 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

Diese Beispiele verwenden die **if**-Anweisung, um in Abhängigkeit von einer Bedingung eine Anweisung auszuführen.

```
'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
```

---

msgbox

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

Diese Beispiele verwenden die **msgbox**-Anweisung, um einen Fehler in der Dialogbox anzuzeigen.

```
'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
```

---

set

Wird verwendet, um einer Variable einen Wert zuzuweisen.

Syntax

set Varibale To Wert

Parameter	Datentyp	Beschreibung
Variable		Der Name der Variable, der ein Wert zugewiesen werden soll.
Wert	variant	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 s number
set zahl To 12345

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

Kapitel 3 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

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 2011“, „ShakeHands Conto 2011“, „ShakeHands Faktura 2011“ oder „ShakeHands Budget 2011“

Beispiel Dieses Beispiel verwendet die **GetAppName**-Funktion, um den Namen des Programms zu ermitteln. In diesem Beispiel wurde das Programm ShakeHands Kontor 2011 benutzt. Die **GetAppName**-Funktion liefert „ ShakeHands Kontor 2011 “.

```
'das Beispiel gibt „ ShakeHands Kontor 2011 “ 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 Dieses Beispiel verwendet die **GetAppVersion**-Funktion, um die Version des Programms zu ermitteln. In diesem Beispiel wurde das Programm ShakeHands Kontor 2011 mit der Version 8.3.0 benutzt. Die **GetAppVersion**-Funktion liefert „8.3.0“.

```
'das Beispiel gibt „8.3.0“ zurück
dim a as text = GetAppVersion
```

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 (14.01.2008) zu ermitteln. Die **DateToText**-Funktion wird verwendet, um das Datum als Text auszugeben.

Siehe auch DateToText.

```
'das Beispiel liefert für b „14.01.08“
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 Dieses Beispiel verwendet die **GetPlatform**-Funktion, um das verwendete Betriebssystem zu ermitteln. Die Anwendung läuft in diesem Beispiel auf einem Mac-System.

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

HasConstant

Überprüft, ob eine Konstante schon vorhanden ist.

Syntax **Ergebnis = HasConstant(Name)**

Parameter	Datentyp	Beschreibung
Name	text	Der Name der Konstante.
Ergebnis	boolean	Ergebnis ist **True**, wenn die Konstante 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 Diese Beispiele verwenden die **HasConstant**-Funktion, um zu überprüfen, ob eine Konstante mit dem Namen „pi“ vorhanden ist.

```
'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

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

Diese Beispiele verwenden die **HasField**-Funktion, um zu überprüfen, ob ein bestimmtes Feld vorhanden ist.

```
'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

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 Diese Beispiele verwenden die **HasTable**-Funktion, um zu überprüfen, ob eine bestimmte Tabelle vorhanden ist.

```
'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

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 Diese Beispiele verwenden die **HasVariable**-Funktion, um zu überprüfen, ob eine Variable vorhanden ist.

```
'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")
```

Booleanfunktionen

BooleanToText

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

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, DayOfWeek, DayOfYear, WeekOfYear, DateToText, DateToNumber. TimeToText, SQLDate, SQLDateTime.

Beispiel

Diese Beispiele verwenden die **Date**-Funktion, um ein Datumsobjekt zu erzeugen.

```
'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

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

Dieses Beispiel verwendet die **DateToNumber**-Funktion, um ein Datum in eine Zahl umzuwandeln.

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

DateToText

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

Diese Beispiele verwenden die **DateToText**-Funktion, um ein Datum in einen Text umzuwandeln.

```
'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

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, DayOfWeek, DayOfYear, WeekOfYear.

Beispiel

Diese Beispiele verwenden die **Day**-Funktion, um den Tag im Monat des Datums auszugeben.

```
'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)
```

DayOfWeek

Gibt den Wochentag eines Datums als Zahl zurück.

Syntax

Ergebnis = DayOfWeek(Datum)

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

Hinweise

Die **DayOfWeek**-Funktion gibt den Wochentag des angegebenen Datums als Zahl zurück. 1=Montag, 2=Dienstag, 3=Mittwoch, 4=Donnerstag, 5=Freitag, 6=Samstag, 7=Sonntag.

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

Beispiel

Dieses Beispiel verwendet die **DayOfWeek**-Funktion, um den Wochentag eines Datums auszugeben.

```
'das Beispiel gibt den aktuellen Wochentag zurück
dim d as date      = GetCurrentDate
dim dw as number   = DayOfWeek(d)
dim wt as text     = Choose(dw - 1, "Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag", "Sonntag", "")
DrawText("Wochentag: " + wt, 10, 10, 400, 50)
```

DayOfYear

Gibt den Tag im Jahr eines Datums zurück.

Syntax

Ergebnis = DayOfYear(Datum)

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist der Tag im Jahr von Datum als Zahl.

Hinweise

Die **DayOfYear**-Funktion gibt den Tag im Jahr des angegebenen Datums als Zahl zurück.

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

Beispiel

Diese Beispiele verwenden die **DayOfYear**-Funktion, um den Tag im Jahr eines Datums auszugeben.

```
'das Beispiel gibt den aktuellen Tag im Jahr zurück
dim d as date      = GetCurrentDate
dim dy as number   = DayOfYear(d)
DrawText(NumToText(dy), 10, 10, 400, 50)

'das Beispiel gibt 1 zurück
dim d as date      = Date(1, 1, 2009)
dim dy as number   = DayOfYear(d)
DrawText(NumToText(dy), 10, 10, 400, 50)
```

Hour

Gibt die Stunden aus der Uhrzeit eines Datums zurück.

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, DayOfWeek, DayOfYear, WeekOfYear.

Beispiel

Dieses Beispiel verwendet die **Hour**-Funktion, um die Stunden aus der Uhrzeit des Datums auszugeben.

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

LongDate

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 langem 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 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 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)
```

LongTime

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 kurzer Schreibweise auszugeben.

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

MediumDate

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 mittlere Schreibweise auszugeben.

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

Minute

Gibt die Minuten aus der Uhrzeit eines Datums zurück.

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, DayOfWeek, DayOfYear, WeekOfYear.

Beispiel Dieses Beispiel verwendet die **Minute**-Funktion, um die Minuten aus der Uhrzeit des Datums auszugeben.

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

Month

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, DayOfWeek, DayOfYear, WeekOfYear.

Beispiel Dieses Beispiel verwendet die **Month**-Funktion, um die Monate des Datums auszugeben.

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

Second

Gibt die Sekunden aus der Uhrzeit eines Datums zurück.

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, DayOfWeek, DayOfYear, WeekOfYear.

Beispiel Dieses Beispiel verwendet die **Second**-Funktion, um die Sekunden aus der Uhrzeit des Datums auszugeben.

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

ShortDate

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)
```

ShortTime

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)
```

SQLDate

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 „2007-12-31“ zurück
dim datum as date = Date(31, 12, 2007)
dim datumSQL as text = SQLDate(datum)
```

SQLDateTime

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 „2007-12-31“ zurück
dim datum as date = Date(31, 12, 2007)
dim datumSQL as text = SQLDateTime(datum)
```

TimeToText

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. Das Zeitformat richtet sich dabei an die Formateinstellung des jeweiligen Betriebsystems (Mac, Windows).

Siehe auch DateToText, DateToNumber. TimeToText.

Beispiel Dieses Beispiele verwendet die **TimeToText**-Funktion, um die Zeit als Text auszugeben.

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

WeekOfYear

Gibt die Kalenderwoche eines Datums als Zahl zurück.

Syntax **Ergebnis = WeekOfYear(Datum)**

Parameter	Datentyp	Beschreibung
Datum	date	Das Datum.
Ergebnis	number	Ergebnis ist die Kalenderwoche von Datum als Zahl.

Hinweise

Die **WeekOfYear**-Funktion gibt die Kalenderwoche des angegebenen Datums als Zahl zurück.

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

Beispiel

Diese Beispiele verwenden die **WeekOfYear**-Funktion, um die Kalenderwoche eines Datums auszugeben.

```
'das Beispiel gibt die aktuelle Kalenderwoche zurück
dim d as date      = GetCurrentDate
dim wy as number   = WeekOfYear(d)
DrawText(NumToText(wy), 10, 10, 400, 50)

'das Beispiel gibt 1 zurück
dim d as date      = Date(1, 1, 2009)
dim wy as number   = WeekOfYear(d)
DrawText(NumToText(wy), 10, 10, 400, 50)
```

Year

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, DayOfWeek, DayOfYear, WeekOfYear.

Beispiel

Dieses Beispiel verwendet die **Year**-Funktion, um das Jahr des Datums auszugeben.

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

Farbfunktionen

CMYColor

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 nehmen, um eine Farbe zuzuweisen.

Siehe auch SetDocument, BeginDocument, RGBColor, HSVColor.

Beispiel

Dieses Beispiel verwendet die **CMYColor**-Funktion, um eine Farbe zu definieren. Die **SetTextColor**-Funktion wird verwendet, um die ausgewählte Farbe als Textfarbe auszuwählen.

```
'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

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

Farbe	Farbton	Sättigung	Helligkeit
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 nehmen, um eine Farbe zuzuweisen.

Siehe auch SetDocument, BeginDocument, RGBColor, CMYColor.

Beispiel

Dieses Beispiel verwendet die **HSVColor**-Funktion, um eine Farbe zu definieren. Die **SetTextColor**-Funktion wird verwendet, um die ausgewählte Farbe als Textfarbe auszuwählen.

```
'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

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, Gruen 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

Farbe	Rot	Grün	Blau
Schwarz	0	0	0

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

Siehe auch SetDocument, BeginDocument, CMYColor, HSVColor.

Beispiel

Dieses Beispiel verwendet die **RGBColor**-Funktion, um eine Farbe zu definieren. Die **SetTextColor**-Funktion wird verwendet, um die ausgewählte Farbe als Textfarbe auszuwählen.

```
'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)
```

Logikfunktionen

Case

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

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.
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

Diese Beispiele verwenden die **Choose**-Funktion, um einen Wert in Abhängigkeit einer Bedingung auszugeben.

```
'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

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.

Parameter	Datentyp	Beschreibung
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

Diese Beispiele verwenden die **IfThen**-Funktion, um einen Wert in Abhängigkeit einer Bedingung auszugeben.

```
'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")
```

Textfunktionen

Asc

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 von 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

Diese Beispiele verwenden die **Asc**-Funktion, um den ASCII-Code eines Zeichens auszugeben.

```
'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

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 Quelle voneinander trennt.
Ergebnis	number	Die Anzahl der Felder in 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

Diese Beispiele verwenden die **CountFields**-Funktion, um die Anzahl der Felder eines Textes auszugeben, die durch ein bestimmtes Trennzeichen getrennt werden.

```
'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 F**TextToNumber**-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

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 Quelle entnommen werden sollen.
Ergebnis	text	Die ersten Anzahl Zeichen von links aus 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

Diese Beispiele verwenden die **Left**-Funktion, um die ersten n-Zeichen eines Textes auszugeben.

```
'das Beispiel gibt „Buch“ zurück
dim wort as text = "Buchführung"
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

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 Quelle

Hinweise

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

Beispiel

Diese Beispiele verwenden die **Length**-Funktion, um die Anzahl der Zeichen eines Textes zu ermitteln.

```
'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

Wandelt alle alphabetischen Zeichen eines Textes in kleingeschriebene Zeichen um.

Syntax

Ergebnis = Lower(Quelle)

Parameter	Datentyp	Beschreibung
Quelle	text	Ein beliebiger Text.
Ergebnis	text	Der Text aus 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

Diese Beispiele verwenden die **Lower**-Funktion, um alle Zeichen in einem Text kleinzuschreiben.

```
'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

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

Diese Beispiele verwenden die **LTrim**-Funktion, um die führenden Leerzeichen aus einem Text zu entfernen.

```
'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

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 sollen.
Start	number	Die Anfangsposition des Ausschnitts.
Laenge	number	Die Anzahl von Zeichen (Länge) des Ausschnitts aus Quelle.
Ergebnis	text	Der Ausschnitt aus 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

Diese Beispiele verwenden die **Middle**-Funktion, um einen Ausschnitt aus einem Text auszugeben.

```
'das Beispiel gibt „Hans“ zurück
dim wort as text = "Name: Hans Mustermann"
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

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

Diese Beispiele verwenden die **NthField**-Funktion, um ein Datenfeld aus einem Text auszugeben.

```
'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

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 Quelle gezählt werden soll.
Ergebnis	number	Die Anzahl des Auftretens von Suche in 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

Diese Beispiele verwenden die **PatternCount**-Funktion, um das Auftreten eines Textes in einem anderen Text zu zählen.

```
'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

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 Quelle gesucht werden soll.
Start	number	Position ab der die Suche in Quelle beginnen soll.
Vorkommen	number	Der Wert des Auftretens von Suche, für das die Position ermittelt werden soll.
Ergebnis	number	Die Position des Vorkommens des Auftretens von Suche in 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 Vorkommensanzahl ab Startposition. Gross- und Kleinschreibung wird bei der Suche nicht berücksichtigt.

Beispiel

Diese Beispiele verwenden die **Position**-Funktion, um die Position des Auftretens eines Textes in einem anderen Text zu bestimmen.

```
'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„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

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 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

Dieses Beispiel verwendet die **Proper**-Funktion, um die Anfangsbuchstaben eines Textes in Grossbuchstaben umzuwandeln.

```
'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

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 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 unveränderte zurück.

Siehe auch ReplaceAll.

Beispiel

Diese Beispiele verwenden die **Replace**-Funktion, um das erste Vorkommen eines Zeichens oder einer Zeichenkette aus einem Text durch andere Zeichen zu ersetzen.

```
'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

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 soll.
Ersetzung	text	Die ersetzenden Zeichen.
Ergebnis	text	Der Text aus 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 unveränderte zurück.

Siehe auch Replace.

Beispiel

Diese Beispiele verwenden die **ReplaceAll**-Funktion, um alle Vorkommen eines Zeichens oder einer Zeichenkette aus einem Text durch andere Zeichen zu ersetzen.

```
'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

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 Quelle entnommen werden sollen.
Ergebnis	text	Die ersten Anzahl Zeichen von rechts aus 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

Diese Beispiele verwenden die **Right**-Funktion, um die ersten n-Zeichen eines Textes auszugeben.

```
'das Beispiel gibt „führung“ zurück
dim wort as text = "Buchführung"
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

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 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

Diese Beispiele verwenden die **RTrim**-Funktion, um die führenden Leerzeichen aus einem Text zu entfernen.

```
'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

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

Wandelt einen Text in einen Boolean-Wert um.

Syntax

Ergebnis = TextToBoolean(Quelle)

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

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

```
'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

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 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. „1.1.2008“, „01.01.2008“, „1.1.08“, oder „01.01.08“). Die Funktion gibt das aktuelle Datum zurück, wenn der Text keine numerische Datumsangabe enthält.

Siehe auch TextToBoolean, TextToNumber.

Beispiel

Diese Beispiele verwenden die **TextToDate**-Funktion, um einen Text in einen Datums-Wert umzuwandeln.

```
'das Beispiel gibt 01.01.08 zurück
dim datText as text = "1.1.2008"
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.2008"
dim datum as date = TextToDate(datText)
```

TextToNumber

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

Diese Beispiele verwenden die **TextToNumber**-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 = TextToNumber(zahlwort)

'das Beispiel gibt 0 zurück
dim zahlwort as text = "Tel: 12345"
dim zahl as number = TextToNumber(zahlwort)
```

Trim

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 Quelle ohne führende und nachfolgende Leerzeichen.

Hinweise

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

Siehe auch RTrim, LTrim.

Beispiel

Dieses Beispiel verwendet die **Trim**-Funktion, um die führenden und nachfolgenden Leerzeichen aus einem Text zu entfernen.

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

Upper

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 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

Diese Beispiele verwenden die **Upper**-Funktion, um alle Zeichen in einem Text grosszuschreiben.

```
'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)
```

Zahlenfunktionen

Abs

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

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

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

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

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.

Parameter	Datentyp	Beschreibung
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

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

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

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 soll.
%	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.
\charac-	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

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

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

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

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

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

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

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

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

Gibt die Potenz einer Zahl zurück

Syntax **Ergebnis = Pow(Basis, Exponent)**

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

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

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

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

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 Diese Beispiele verwenden die **Sqrt**-Funktion, um die Quadratwurzel der angegebenen Zahl zu ermitteln.

```
'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)
```

Kapitel 4 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 erfolgt über unsere Hotline. Voraussetzung ist die vorher vollzogene Registrierung als Anwender für das betreffende Produkt.

Bevor Sie den Support beanspruchen, versuchen Sie das Problem bitte mit Hilfe der entsprechenden Dokumentation zu lösen. Bitte versuchen Sie vor Kontaktaufnahme das Problem zu reproduzieren und die genaue Art und Weise des Zustandekommens (unter welchen Bedingungen) zu beschreiben. Überlegen Sie 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 3.00 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, Lausanne Kursdauer: 4 Stunden Teilnehmerzahl: 2 bis 4 Personen
Einzelschulung nach Mass	Programme: ShakeHands Buchhaltungen, ShakeHands ERP-Lösungen, Unilohn Schulungsort: in Ihrem Unternehmen Kursdauer: ein halber Tag Teilnehmerzahl: bis 4 MitarbeiterInnen

Support/Service	Erreichbar über
Kostenlose Supportviertelstunde	Supportviertelstunde: Registrierte UserInnen erhalten in der Kulanz-Zeit (6 Monate nach dem Kauf) bei Shake-Hands-Eigenprodukten eine Viertelstunde kostenlosen Support per Email, Telefon oder Fernwartung. Nach der Kulanz-Zeit oder ab der sechzehnten Minute Support stehen Ihnen die obigen kostenpflichtigen Optionen offen.
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. http://www.monkey-office.de/forum/index.php