FormCalc-Benutzerreferenz

Adobe® LiveCycle® Designer

Version 8.0

Adobe® LiveCycle® Designer 8.0 FormCalc-Benutzerreferenz für Microsoft® Windows®
Oktober 2006

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Inhalt

Vorwort

Adobe® LiveCycle® Designer 8.0 stellt ein Werkzeugsortiment zur Verfügung, mit dem Formularentwickler intelligente Geschäftsdokumente erstellen können. Mit Hilfe von integrierten Berechnungen und Skripten lassen sich Formulare für die Empfänger attraktiver gestalten. Beispielsweise könnten Sie einfache Berechnungen verwenden, um die Kosten in einem Bestellauftrag automatisch zu aktualisieren, oder Sie könnten mit anspruchsvolleren Skripterstellungsfunktionen das Aussehen Ihres Formulars an das Gebietsschema des Benutzers anpassen.

Zur Erleichterung der Erstellung von Berechnungen bietet LiveCycle Designer dem Benutzer FormCalc. FormCalc ist eine einfache Berechnungssprache, die Adobe auf der Grundlage verbreiteter Tabellenkalkulationsanwendungen entwickelt hat. FormCalc ist unkompliziert und kann auch von Benutzern mit wenig oder gar keiner Erfahrung in der Skripterstellung genutzt werden. Da sich FormCalc an zahlreichen Regeln und Konventionen orientiert, die auch in anderen Skriptsprachen gebräuchlich sind, können erfahrene Formulardesigner ihre Kenntnisse auch auf FormCalc anwenden.

In diesem Handbuch

Dieses Handbuch wendet sich an Formularentwickler, die LiveCycle Designer verwenden und FormCalc-Berechnungen in ihren Formularen einsetzen möchten. Es bietet eine Referenz zu den FormCalc-Funktionen, nach Funktionskategorie in Kapitel unterteilt. Außerdem ist eine Einführung in die FormCalc-Sprache und die Bausteine von FormCalc-Ausdrücken enthalten.

Benutzer dieses Handbuchs

Dieses Handbuch enthält Informationen zur Unterstützung von Formulardesignern, die mit Hilfe der FormCalc-Sprache Berechnungen zur Erweiterung ihrer in LiveCycle Designer erstellten Formularentwürfe erstellen möchten.

Weiterführende Dokumentation

Weitere Informationen zur Verwendung von FormCalc-Berechnungen in Ihren Formularen finden Sie unter *Berechnungen und Skripten erstellen* in der *LiveCycle Designer-Hilfe*.

Wenn Sie zusätzliche technische Informationen über FormCalc benötigen, konsultieren Sie bitte die *Adobe XML Forms 2.1 Specification*, die auf der Website des Adobe Solutions Network (ASN) verfügbar ist.

1 Einführung in FormCalc

FormCalc ist eine einfache, aber leistungsfähige Berechnungssprache, die nach dem Vorbild allgemein gebräuchlicher Tabellenkalkulations-Software entwickelt wurde. Diese Sprache soll die schnelle, effiziente Gestaltung von Formularen ohne Kenntnis konventioneller Skripterstellungstechniken oder Skriptsprachen ermöglichen. Benutzer, die FormCalc noch nicht kennen, können mit Hilfe von wenigen integrierten Funktionen in kurzer Zeit Formulare erstellen, welche den Endbenutzern zeitaufwendige Berechnungen, Kontrollen und andere Überprüfungsmaßnahmen abnehmen. Auf diese Weise kann der Entwickler sein Formular bei der Gestaltung mit einer elementaren Intelligenz versehen, mit welcher das so entstehende interaktive Formular auf die eingegebenen Daten reagiert.

Die integrierten Funktionen, aus denen sich FormCalc zusammensetzt, decken eine umfangreiche Palette von Bereichen ab: Mathematik, Datum und Uhrzeit, Zeichenfolgen, Finanzen, Logik und Internet. Diese Bereiche entsprechen den Datentypen, die in Formularen typischerweise verwendet werden. Die Funktionen sollen eine schnelle und einfache Handhabung der Daten auf zweckmäßige Weise ermöglichen.

Skripterstellung in LiveCycle Designer

Bei LiveCycle Designer ist FormCalc die Standard-Skriptsprache, die überall zum Einsatz kommt, wo Skripten erstellt werden; alternativ wird JavaScript™ eingesetzt. Skripterstellung erfolgt bei den verschiedenen Ereignissen, die Formularobjekte betreffen. Sie können eine Mischung aus FormCalc und JavaScript bei interaktiven Formularen verwenden. Wenn Sie aber einen serverbasierten Prozess wie LiveCycle Forms verwenden, um Formulare für die Anzeige in einem Internet-Browser zu erstellen, werden FormCalc-Skripten zu bestimmten Formularobjekt-Ereignissen nicht im HTML-Formular wiedergegeben. Hierdurch soll verhindert werden, dass im Internet-Browser Fehler auftreten, wenn Benutzer mit dem fertigen Formular arbeiten.

2 Sprachreferenz

Bausteine

Die FormCalc-Sprache besteht aus einzelnen Bausteinen, die FormCalc-Ausdrücke bilden. Jeder FormCalc-Ausdruck ist eine Folge bestimmter Kombinationen dieser Komponenten.

Literale

Literale sind konstante Werte, die als Grundlage für alle Werte dienen, die zur Verarbeitung an FormCalc übergeben werden. Es gibt zwei allgemeine Typen von Literalen: Zahlen und Zeichenfolgen.

Zahlenliterale

Ein Zahlenliteral ist eine hauptsächlich aus Ziffern bestehende Folge, die sich aus einem oder mehreren der folgenden Zeichen zusammensetzt: Ganzzahl, Dezimalpunkt, Bruchanteil, Exponentenzeichen („e“ oder „E“) sowie ein optionaler vorzeichenbehafteter Exponentenwert. Die folgenden Beispiele zeigen einige Literalzahlen:

- -12
- 1.5362
- 0.875
- 5.56e-2
- 1.234E10

Entweder der Ganzzahl- oder der Bruchanteil einer Literalzahl kann weggelassen werden, aber nicht beide gleichzeitig. Außerdem kann im Bruchanteil entweder der Dezimalpunkt oder der Exponentenwert weggelassen werden, aber nicht beide gleichzeitig.

Alle Zahlenliterale werden intern gemäß einem Standard des Institute of Electrical and Electronics Engineers (IEEE) in 64 Bit lange Binärwerte umgewandelt. Da die IEEE-Werte aber nur eine endliche Zahlenmenge darstellen können, lassen sich bestimmte Werte in dieser Form nicht als binärer Bruch darstellen, ähnlich wie sich bestimmte Werte (z. B. 1/3) als Dezimalbruch nicht exakt darstellen lassen (ein völlig genauer Dezimalwert müsste unendlich viele Dezimalstellen haben).

Die Werte ohne äquivalente binäre Bruchdarstellung sind in der Regel Zahlenliterale mit mehr als 16 signifikanten Ziffern vor dem Exponenten. FormCalc rundet diese Werte auf den nächsten nach dem IEEE-Standard darstellbaren 64-Bit-Wert. Beispielsweise wird der Wert

```
123456789.012345678
```

auf den (nächsten) Wert gerundet:

```
123456789.01234567
```

In einem zweiten Beispiel wird dagegen das Zahlenliteral

```
9999999999999999
```

auf den (nächsten) Wert gerundet:

```
100000000000000000
```

Dieses Verhalten kann in manchen Fällen zu unerwarteten Ergebnissen führen. FormCalc enthält die Funktion Round, welche die übergebene Zahl auf eine angegebene Zahl von Dezimalstellen gerundet zurückgibt. Wenn die übergebene Zahl genau in der Mitte zwischen zwei darstellbaren Zahlen liegt, wird „von 0 weg“ gerundet; das heißt, eine positive Zahl wird nach oben gerundet, eine negative nach unten. Dazu zwei Beispiele:

```
Round(0.124, 2)
```

liefert `0.12`

und

```
Round(.125, 2)
```

liefert `0.13`.

Folglich würde man erwarten, dass

```
Round(0.045, 2)
```

`0.05` liefert.

Die IEEE-Norm 754 schreibt aber vor, dass das Zahlenliteral `0.045` als `0.0449999999999999` anzunähern ist. Dieser Näherungswert liegt näher bei `0.04` als bei `0.05`. Daher liefert

```
Round(0.045, 2)
```

das Ergebnis `0.04`.

Dies entspricht ebenfalls der IEEE-Norm 754.

64-Bit-Werte gemäß der IEEE-Norm unterstützen auch Darstellungen wie NaN (Not a Number – keine Zahl), +Inf (positiv unendlich) und -Inf (negativ unendlich). In FormCalc werden diese Werte jedoch nicht unterstützt, und Ausdrücke, die zu NaN, +Inf oder -Inf ausgewertet werden, führen zu einem Ausnahmefehler, der dann auch auf den Rest des Ausdrucks übertragen wird.

Zeichenfolgenliterale

Ein Zeichenfolgenliteral ist eine in Anführungszeichen eingeschlossene Folge von beliebigen Unicode-Zeichen. Beispiel:

```
"The cat jumped over the fence."
"Number 15, Main street, California, U.S.A"
```

Das Zeichenfolgenliteral `""` steht für eine leere Folge von Textzeichen, die als „leere Zeichenfolge“ bezeichnet wird.

Wenn Sie innerhalb eines Zeichenfolgenliterals Anführungszeichen verwenden möchten, müssen Sie vor jedes Anführungszeichen einen umgekehrten Schrägstrich (\) setzen. Beispiel:

```
"The message reads: ""Warning: Insufficient Memory"""
```

Zu jedem Unicode-Zeichen gibt es eine äquivalente, 6 Zeichen lange Escape-Sequenz, die aus `\u` und vier nachfolgenden Hexadezimalziffern besteht. In einem Zeichenfolgenliteral lässt sich jedes Zeichen, einschließlich der Steuerzeichen, mit einer entsprechenden Unicode-Escape-Sequenz darstellen. Beispiel:

```
"\u0047\u006f\u0066\u0069\u0073\u0068\u0021"
"\u000d" (carriage return)
"\u000a" (newline character)
```

Operatoren

Zu FormCalc gehört eine Reihe von Operatoren: unär, multiplikativ, additiv, relational, Gleichheit, logisch und der Zuweisungsoperator.

Zu mehreren FormCalc-Operatoren gibt es ein äquivalentes mnemonisches Operator-Schlüsselwort. Diese Schlüsselwort-Operatoren sind immer dann nützlich, wenn FormCalc-Ausdrücke in HTML- und XML-Quelltext eingebettet sind, wo die Symbole „Kleiner als“ (<) und „Größer als“ (>) sowie das Et-Zeichen (&) eine vordefinierte Bedeutung haben und daher vermieden werden müssen. In der folgenden Tabelle sind alle FormCalc-Operatoren aufgeführt. Neben der symbolischen ist gegebenenfalls auch die mnemonische Darstellungsform angegeben.

Operator-Typ	**Darstellung**
Addition	+
Division	/
Gleichheit	== eq <> ne
Logisches UND	& and
Logisches ODER	| or
Multiplikation	*
Relational	< lt (less than - kleiner als) > gt (greater than - größer als) <= le (less than or equal to - kleiner oder gleich) >= ge (greater than or equal to - größer oder gleich)
Subtraktion	-
Unär	- + not

Kommentare

Kommentare sind Code-Abschnitte, die von FormCalc nicht ausgeführt werden. Kommentare enthalten in der Regel Informationen oder Anleitungen, in denen die Verwendung eines bestimmten Code-Abschnitts erläutert wird. Alle in Kommentaren gespeicherten Informationen werden von FormCalc während der Laufzeit ignoriert.

Kommentare werden entweder durch ein Semikolon (;) oder zwei Schrägstriche (//) gekennzeichnet. In FormCalc erstreckt sich ein Kommentar von seinem Anfang bis zum nächsten Zeilenende-Zeichen.

Zeichenname	Darstellung
Kommentar	; //

Beispiel:

```
// This is a type of comment
First_Name="Tony"
Initial="C" ;This is another type of comment
Last_Name="Blue"
```

Alle FormCalc-Berechnungen für ein Ereignis kommentieren

Wenn sämtliche FormCalc-Berechnungen für ein bestimmtes Ereignis kommentiert sind, wird ein Fehler ausgegeben, wenn Sie sich Ihr Formular über die Registerkarte „PDF-Vorschau“ ansehen oder die fertige PDF-Datei anzeigen. Bei jeder FormCalc-Berechnung muss ein Wert ausgegeben werden. Kommentare werden von FormCalc nicht als Werte akzeptiert.

Gehen Sie folgendermaßen vor, um diese Fehlerausgabe bei kommentiertem FormCalc-Code zu umgehen:

- Entfernen Sie den kommentierten Code aus dem Ereignis.
- Fügen Sie einen Ausdruck hinzu, der einen Wert für den FormCalc-Code des Ereignisses ausgibt.

Verwenden Sie einen der folgenden Ausdrücke, damit der Wert des Ausdrucks keine unerwünschten Ergebnisse in Ihrem Formular ausgibt:

- Einen einfachen, aus einem Zeichen bestehenden Ausdruck, wie im folgenden Beispiel:

```
// First_Name="Tony"
// Initial="C"
// Last_Name="Blue"
//
// The simple expression below sets the value of the event to zero.
0
```

- Einen Zuweisungsausdruck, welcher den Wert des Objekts erhält. Verwenden Sie diesen Ausdruckstyp, wenn sich Ihr kommentierter FormCalc-Code im berechneten Ereignis befindet, damit der eigentliche Objektwert nicht verändert wird (siehe folgendes Beispiel):

```
// First_Name="Tony"
// Initial="C"
// Last_Name="Blue"
//
// The assignment expression below sets the value of the current
// field equal to itself.
$.rawValue = $.rawValue
```

Schlüsselwörter

Schlüsselwörter in FormCalc sind reservierte Wörter, bei denen die Groß-/Kleinschreibung nicht beachtet wird. Diese Schlüsselwörter werden als Teile von Ausdrücken, spezielle Zahlenliterale und Operatoren verwendet.

Die folgende Tabelle zeigt die FormCalc-Schlüsselwörter. Verwenden Sie keines dieser Wörter für die Benennung von Formularobjekten.

and	endif	in	step
break	endwhile	infinity	then
continue	eq	le	this
do	exit	lt	throw
downto	for	nan	upto
else	foreach	ne	var
elseif	func	not	while
end	ge	null	
endfor	gt	or	
endfunc	if	return	

Bezeichner

Ein Bezeichner ist eine Zeichenkette beliebiger Länge, welche den Namen einer Funktion oder Methode angibt. Jeder Bezeichner muss mit einem der folgenden Zeichen beginnen:

- Einem beliebigen alphabetischen Zeichen (gemäß den Unicode-Buchstabenklassifikationen)
- Unterstrich (_)
- Dollarzeichen ($)
- Ausrufezeichen (!)

In FormCalc ist die Groß-/Kleinschreibung der Bezeichner relevant; das heißt, Bezeichner, deren Namen sich nur durch die Groß-/Kleinschreibung von Buchstaben unterscheiden, gelten als verschieden.

Zeichenname	**Darstellung**
Bezeichner	A..Z,a..z $! _

Gültige Bezeichner sind beispielsweise:

```
GetAddr
$primary
_item
!dbresult
```

Zeilenende-Zeichen

Zeilenende-Zeichen werden für die Trennung von Zeilen und die Verbesserung der Lesbarkeit verwendet.

In der folgenden Tabelle sind alle in FormCalc gültigen Zeilenende-Zeichen aufgeführt:

Zeichenname	Unicode-Zeichen
Wagenrücklauf	#xD U+000D
Zeilenvorschub	#xA
 &#D;

Leerraum

Leerraum-Zeichen trennen verschiedene Objekte und mathematische Operationen voneinander. Diese Zeichen dienen ausschließlich zur Verbesserung der Lesbarkeit und werden in FormCalc während der Verarbeitung nicht beachtet.

Zeichenname	Unicode-Zeichen
Formularvorschub	#xC
Horizontaler Tabulator	#x9
Leertaste	#x20
Vertikaler Tabulator	#xB

Ausdrücke

Die Literale, Operatoren, Kommentare, Schlüsselwörter, Bezeichner, Zeilenende-Zeichen und Leerräume bilden gemeinsam eine Liste von Ausdrücken, auch wenn die Liste nur einen einzelnen Ausdruck enthält. Allgemein wird jeder Ausdruck in der Liste zu einem Wert aufgelöst; der Wert der gesamten Liste ist der Wert des letzten Ausdrucks in der Liste.

Betrachten Sie zum Beispiel das folgende Szenario mit zwei Feldern auf einem Formularentwurf:

Feldname	Berechnungen	Rückgabe

`Field1`	`5 + Abs(Price)`
	`„Hello World“`
	`10 * 3 + 5 * 4`
		`50`

`Field2`	`10 * 3 + 5 * 4`	`50`

Sowohl `Field1` als auch `Field2` haben nach der Auswertung ihrer jeweiligen Ausdrucksliste den Wert `50`.

FormCalc unterteilt die verschiedenen Arten von Ausdrücken, aus denen eine Ausdrucksliste besteht, in die folgenden Kategorien:

Einfach

In ihrer elementarsten Form sind FormCalc-Ausdrücke Gruppen von Operatoren, Schlüsselwörtern und Literalen, die logisch miteinander verknüpft sind. In den folgenden Fällen zum Beispiel handelt es sich jeweils um einfache Ausdrücke:

```
2
"abc"
2 - 3 * 10 / 2 + 7
```

Jeder FormCalc-Ausdruck wird zu einem einzelnen Wert aufgelöst. Dabei gilt die konventionelle Reihenfolge der Operationen, auch wenn diese Reihenfolge aus der Syntax der Ausdrücke nicht immer offensichtlich ist. Wenn beispielsweise die folgenden Ausdrücke auf Objekte in einem Formularentwurf angewandt werden, sehen die Ergebnisse wie folgt aus:

Ausdruck	Entspricht	Rückgabe
`"abc"`	`"abc"`	`abc`
`2 - 3 * 10 / 2 + 7`	`2 - (3 * (10 / 2)) + 7`	`-6`
`10 * 3 + 5 * 4`	`(10 * 3) + (5 * 4)`	`50`
`0 and 1 or 2 > 1`	`(0 and 1) or (2 >1)`	`1` (TRUE)
`2 < 3 not 1 == 1`	`(2 < 3) not (1 == 1)`	`0` (FALSE)

Wie aus der obigen Tabelle hervorgeht, besitzen alle FormCalc-Operatoren innerhalb von Ausdrücken eine bestimmte Priorität. Die folgende Tabelle zeigt die Hierarchie der Operatoren im Überblick:

Priorität	**Operator**
Höchste	=
	(Unär) - , + , not
	* , /
	+ , -
	< , <= , > , >= , lt , le , gt , ge
	== , <> , eq , ne
	& , and
Niedrigste	| , or

Umwandlung von Operanden

Falls ein oder mehrere Operanden in einer bestimmten Operation nicht dem erwarteten Typ für die betreffende Operation entsprechen, wandelt FormCalc die Operanden in den benötigten Typ um. Wie diese Umwandlung erfolgt, hängt davon ab, welchen Operandentyp die Operation benötigt.

Numerische Operationen

Bei der Ausführung von numerischen Operationen, an denen nicht-numerische Operanden beteiligt sind, werden die nicht-numerischen Operanden zunächst in ihre numerische Entsprechung umgewandelt. Wenn sich der nicht-numerische Operand nicht erfolgreich in einen Zahlenwert konvertieren lässt, ist sein Wert 0. Wenn nullwertige Operanden in Zahlen umgewandelt werden, ist ihr Wert stets 0.

Die folgende Tabelle liefert einige Beispiele für die Anpassung von nicht-numerischen Operanden.

Ausdruck	**Entspricht**	**Rückgabe**
`(5 - "abc") * 3`	`(5 - 0) * 3`	`15`
`"100" / 10e1`	`100 / 10e1`	`1`
`5 + null + 3`	`5 + 0 + 3`	`8`

Boolesche Operationen

Bei der Ausführung von booleschen Operationen, an denen nicht-boolesche Operanden beteiligt sind, werden die nicht-booleschen Operanden zunächst in ihre booleschen Entsprechungen umgewandelt. Wenn sich der nicht-boolesche Operand nicht erfolgreich in einen von 0 verschiedenen Wert umwandeln lässt, hat er den Wert TRUE (1); andernfalls hat er den Wert FALSE (0). Wenn nullwertige Operanden in einen booleschen Wert umgewandelt werden, ist dieser Wert stets FALSE (0). Beispielsweise wird der Ausdruck

```
"abc" | 2
```

als `1` ausgewertet. Das heißt, false | true = true. Hingegen wird

```
if ("abc") then
10
else
20
endif
```

als `20` ausgewertet.

Zeichenfolgenoperationen

Wenn Zeichenfolgenoperationen bei Operanden ausgeführt werden, die keine Zeichenfolgen sind, werden diese Operanden zunächst in Zeichenfolgen umgewandelt. Dabei wird jeweils der Wert des Operanden als Zeichenfolge verwendet. Wenn nullwertige Operanden in Zeichenfolgen umgewandelt werden, entspricht ihr Wert stets der leeren Zeichenfolge. Beispielsweise wird der Ausdruck

```
concat("The total is ", 2, " dollars and ", 57, " cents.")
```

als `"The total is 2 dollars and 57 cents."` ausgewertet.

Hinweis: Wenn während der Auswertung eines Ausdrucks ein Zwischenschritt das Ergebnis NaN, +Inf oder -Inf liefert, erzeugt FormCalc einen Ausnahmefehler und überträgt diesen Fehler auf den Rest des Ausdrucks. Dadurch liefert dieser Ausdruck stets den Wert 0, z. B.:

```
3 / 0 + 1
```

wird als 0 ausgewertet.

Zuweisung

Ein Zuweisungsausdruck setzt die von einer gegebenen Referenz-Syntax bezeichnete Eigenschaft als Wert eines einfachen Ausdrucks fest. Beispiel:

```
$template.purchase_order.name.first = "Tony"
```

Hierdurch wird der Wert des Formularentwurfsobjekts „first“ auf `Tony` gesetzt.

Weitere Informationen zur Verwendung der Referenz-Syntax finden Sie unter „Referenz-Syntax“ auf Seite 24.

Logisches ODER

Ein logischer ODER-Ausdruck gibt TRUE (1) zurück, wenn mindestens einer der Operanden TRUE (1) ist, oder FALSE (0), wenn beide Operanden FALSE (0) sind. Wenn beide Operanden null sind, liefert der Ausdruck das Ergebnis null.

Ausdruck	Zeichendarstellung
Logisches ODER	| or

Die folgenden Beispiele verdeutlichen die Verwendung des logischen ODER-Ausdrucks:

Ausdruck	Rückgabe
`1 or 0`	`1` (TRUE)
`0 | 0`	`0` (FALSE)
`0 or 1 | 0 or 0`	`1` (TRUE)

Logisches UND

Ein logischer UND-Ausdruck gibt TRUE (1) zurück, wenn beide Operanden TRUE (1) sind, oder FALSE, wenn mindestens einer der Operanden FALSE (0) ist. Wenn beide Operanden null sind, liefert der Ausdruck das Ergebnis null.

Ausdruck	Zeichendarstellung
Logisches UND	& and

Die folgenden Beispiele verdeutlichen die Verwendung des logischen UND-Ausdrucks:

Ausdruck	Rückgabe
`1 and 0`	`0` (FALSE)
`0 & 0`	`1` (TRUE)
`0 and 1 & 0 and 0`	`0` (FALSE)

Unär

Unäre Ausdrücke liefern je nach verwendetem unärem Operator unterschiedliche Ergebnisse.

Ausdruck	Zeichendarstellung	Rückgabe
Unär	-	Die arithmetische Negierung des Operanden bzw. null, wenn der Operand null ist.

Ausdruck	Zeichendarstellung	Rückgabe
	+	Der arithmetische Wert des Operanden (unverändert) bzw. null, wenn der Operand null ist.
	not	Die logische Negierung des Operanden.

Hinweis: Die arithmetische Negierung eines Null-Operanden liefert das Ergebnis null, die logische Negierung eines Null-Operanden dagegen liefert das boolesche Ergebnis TRUE. Dem entspricht die folgende, aus der Alltagserfahrung stammende Aussage: Wenn „null“ „nichts“ heißt, muss „nicht nichts“ gleichbedeutend mit „etwas“ sein.

Die folgenden Beispiele verdeutlichen die Verwendung unärer Ausdrücke:

Ausdruck	Rückgabe
`-(17)`	`-17`
`-(-17)`	`17`
`+(17)`	`17`
`+(-17)`	`-17`
`not("true")`	`1` (TRUE)
`not(1)`	`0` (FALSE)

Gleichheit und Ungleichheit

Gleichheits- und Ungleichheits-Ausdrücke liefern das Ergebnis eines Vergleichs der Operanden auf Gleichheit zurück.

Ausdruck	Zeichendarstellung	Rückgabe
Gleichheit	== eq	TRUE (1), wenn die beiden Operanden beim Vergleich identisch sind, bzw. FALSE (0), wenn sie beim Vergleich nicht identisch sind.
Ungleichheit	<> ne	TRUE (1), wenn die beiden Operanden beim Vergleich nicht identisch sind, bzw. FALSE (0), wenn sie beim Vergleich identisch sind.

Außerdem gelten für die Verwendung der Gleichheitsoperatoren die folgenden Spezialfälle:

- Wenn einer der Operanden null ist, wird ein Null-Vergleich durchgeführt. Nullwertige Operanden werden beim Vergleich als identisch gewertet, wenn beide Operanden null sind, und als verschieden, wenn einer der Operanden nicht null ist.
- Wenn beide Operanden Referenzen sind, werden die beiden Operanden als identisch gewertet, wenn sie sich beide auf das gleiche Objekt beziehen, und als verschieden, wenn sie sich nicht auf das gleiche Objekt beziehen.
- Wenn beide Operanden Zeichenfolgen sind, wird ein lexikografischer Zeichenfolgenvergleich der Operanden unter Berücksichtigung des Gebietsschemas durchgeführt. Andernfalls werden die Operanden, sofern nicht beide Operanden null sind, in Zahlenwerte umgewandelt und ein numerischer Vergleich wird durchgeführt.

Die folgenden Beispiele verdeutlichen die Verwendung der Gleichheits- und Ungleichheits-Ausdrücke:

Ausdruck	**Rückgabe**
`3 == 3`	`1` (TRUE)
`3 <> 4`	`1` (TRUE)
`"abc" eq "def"`	`0` (FALSE)
`"def" ne "abc"`	`1` (TRUE)
`5 + 5 == 10`	`1` (TRUE)
`5 + 5 <> "10"`	`0` (FALSE)

Relational

Ein relationaler Ausdruck gibt das boolesche Ergebnis eines relationalen Vergleichs der Operanden zurück.

Ausdruck	**Zeichendarstellung**	**Rückgabe**
Relational	< lt	TRUE (1), wenn der erste Operand kleiner als der zweite Operand ist, und FALSE (0), wenn der erste Operand größer als der zweite Operand ist.
	> gt	TRUE (1), wenn der erste Operand größer als der zweite Operand ist, und FALSE (0), wenn der erste Operand kleiner als der zweite Operand ist.
	<= le	TRUE (1), wenn der erste Operand kleiner als der zweite Operand oder gleich groß ist, und FALSE (0), wenn der erste Operand größer als der zweite Operand ist.
	>= ge	TRUE (1), wenn der erste Operand größer als der zweite Operand oder gleich groß ist, und FALSE (0), wenn der erste Operand kleiner als der zweite Operand ist.

Außerdem gelten für die Verwendung der relationalen Operatoren die folgenden Spezialfälle:

- Wenn einer der Operanden null ist, wird ein Null-Vergleich durchgeführt. Nullwertige Operanden werden beim Vergleich als identisch bewertet, wenn beide Operanden null sind und der relationale Operator „kleiner gleich“ oder „größer gleich“ verwendet wird; andernfalls werden sie als verschieden bewertet.
- Wenn beide Operanden Zeichenfolgen sind, wird ein lexikografischer Zeichenfolgenvergleich der Operanden unter Berücksichtigung des Gebietsschemas durchgeführt. Andernfalls werden die Operanden, sofern nicht beide Operanden null sind, in Zahlenwerte umgewandelt und ein numerischer Vergleich wird durchgeführt.

Die folgenden Beispiele verdeutlichen die Verwendung relationaler Ausdrücke:

Ausdruck	Rückgabe
`3 < 3`	`0` (FALSE)
`3 > 4`	`0` (FALSE)
`"abc" <= "def"`	`1` (TRUE)
`"def" > "abc"`	`1` (TRUE)
`12 >= 12`	`1` (TRUE)
`"true" < "false"`	`0` (FALSE)

if-Ausdrücke

Ein if-Ausdruck ist eine bedingte Anweisung, die prüft, ob ein gegebener einfacher Ausdruck wahr ist, und anschließend das Ergebnis einer Liste von Ausdrücken zurückgibt, die diesem Wahrheitswert entsprechen. Wenn der erste einfache Ausdruck als FALSE (0) ausgewertet wird, prüft FormCalc für alle vorhandenen elseif- und else-Bedingungen, ob diese wahr sind, und gibt gegebenenfalls das Ergebnis ihrer Ausdruckslisten zurück.

Ausdruck	Syntax	Rückgabe

if	`if ( simple expression ) then`
	`  list of expressions`
	`elseif ( simple expression ) then`
	`  list of expressions`
	`else`
	`  list of expressions`
	`endif`
		Das Ergebnis der Liste von Ausdrücken, welche den gültigen Bedingungen (sofern vorhanden) im if-Ausdruck zugeordnet ist.
		Hinweis: Sie müssen keine elseif(...)- oder else-Anweisungen als Teil des if-Ausdrucks verwenden; das Ende des Ausdrucks muss aber mit endif gekennzeichnet werden.

Die folgenden Beispiele verdeutlichen die Verwendung des if-Ausdrucks:

Ausdruck	Rückgabe

`if ( 1 < 2 ) then`
`  1`
`endif`
	`1`

`if ( "abc" > "def") then`
`  1 and 0`
`else`
`  0`
`endif`
	`0`

`if ( Field1 < Field2 ) then`
`  Field3 = 0`
`elseif ( Field1 > Field2 ) then`
`  Field3 = 40`
`elseif ( Field1 = Field2 ) then`
`  Field3 = 10`
`endif`
	Je nach den Werten von `Field1` und `Field2` unterschiedlich. Beispiel: Wenn Field1 den Wert 20 und Field2 den Wert 10 hat, setzt dieser Ausdruck Field3 auf den Wert 40.

while-Ausdrücke

Ein while-Ausdruck ist eine iterative Anweisung oder Schleife, die einen gegebenen einfachen Ausdruck prüft. Wenn das Ergebnis der Prüfung TRUE (1) ist, prüft FormCalc wiederholt die do-Bedingung und gibt die Ergebnisse der Ausdruckslisten zurück. Wenn das Ergebnis FALSE (0) ist, wird die Kontrolle an die nächste Anweisung übergeben.

Ein while-Ausdruck ist besonders dann geeignet, wenn bedingte Wiederholungen benötigt werden. Im Gegensatz hierzu sind for-Ausdrücke besser für Situationen geeignet, in denen nicht bedingte Wiederholungen benötigt werden.

Ausdruck	Syntax	Rückgabe

while	`while ( simple expression ) do`
	`  expression list`
	`endwhile`
		Das Ergebnis der Liste von Ausdrücken, welche der do-Bedingung zugeordnet sind.

Im folgenden Beispiel werden die Werte der Elemente einer Dropdown-Liste aus einer XML-Datei hinzugefügt, indem die addItem-Methode für alle XML-Elemente verwendet wird, die in list1 aufgelistet und ungleich 3 sind.

```
var List = ref(xfa.record.lists.list1)
var i = 0
while ( List.nodes.item(i+1).value ne "3")do
$.addItem (List.nodes.item(i).value,List.nodes.item(i+1).value)
i = i + 2
endwhile
```

for-Ausdrücke

Ein for-Ausdruck ist eine bedingte iterative Anweisung oder Schleife.

Ein for-Ausdruck ist besonders in Schleifenlogiksituationen geeignet, in denen nicht bedingte Wiederholungen benötigt werden. Im Gegensatz hierzu sind while-Ausdrücke besser für Situationen geeignet, in denen bedingte Wiederholungen benötigt werden.

Der Wert des for-Ausdrucks ist der Wert der letzten Auswertungsliste, die ausgewertet wurde, oder FALSE (0).

Die for-Bedingung initialisiert eine FormCalc-Variable, welche die Schleifenlogikaktion steuert.

In der upto-Variante iteriert der Wert der Schleifenvariablen vom start-Ausdruck zum end-Ausdruck in step-Ausdrucksinkrementen. Wenn Sie den step-Ausdruck weglassen, ist das step-Inkrement standardmäßig 1.

In der downto-Variante iteriert der Wert der Schleifenvariablen vom start-Ausdruck zum end-Ausdruck in step-Ausdrucksdekrementen. Wen der step-Ausdruck weggelassen wird, ist das step-Dekrement standardmäßig -1.

Die Iterationen der Schleife werden vom Wert des end-Ausdrucks gesteuert. Vor jeder Iteration wird der end-Ausdruck ausgewertet und mit der Schleifenvariablen verglichen. Wenn das Ergebnis TRUE (1) ist, wird die Ausdrucksliste ausgewertet. Nach jeder Auswertung wird der step-Ausdruck ausgewertet und der Schleifenvariablen hinzugefügt.

Vor jeder Iteration wird der end-Ausdruck ausgewertet und mit der Schleifenvariablen verglichen. Außerdem wird nach jeder Auswertung der do-Bedingung der step-Ausdruck ausgewertet und der Schleifenvariablen hinzugefügt.

Eine for-Schleife endet, wenn der start-Ausdruck den end-Ausdruck überschritten hat. Der start-Ausdruck kann den end-Ausdruck entweder aufwärts überschreiten, wenn Sie upto verwenden, oder abwärts, wenn Sie downto verwenden.

Ausdruck	Syntax	Rückgabe

for	`for variable = start expression`
	`(upto | downto ) end expression`
	`(step step expression ) do`
	`expression list`
	`endfor`
	Hinweis: Die start-, end- und step-Ausdrücke müssen alle einfache Ausdrücke sein.
		Das Ergebnis der Liste von Ausdrücken, welche der do-Bedingung zugeordnet sind.

Im folgenden Beispiel werden die Werte der Elemente einer Dropdown-Liste aus einer XML-Datei hinzugefügt, indem die addItem-Methode für alle XML-Elemente verwendet wird, die in list1 aufgelistet sind.

```
var List = ref(xfa.record.lists.list1)
for i=0 upto List.nodes.length - 1 step 2 do
$.addItem (List.nodes.item(i).value,"")
endfor
```

foreach-Ausdrücke

Ein foreach-Ausdruck iteriert über die Liste der Ausdrücke für jeden Wert in seiner Liste von Argumenten.

Der Wert des foreach-Ausdrucks ist der Wert der letzten Ausdrucksliste, die ausgewertet wurde, oder null (0), wenn die Schleife nicht in Gang gesetzt wurde.

Die in-Bedingung, die nur einmal ausgeführt wird (nachdem die Schleifenvariable deklariert wurde), steuert die Iteration der Schleife. Vor jeder Iteration werden der Schleifenvariablen aufeinander folgende Werte aus der Liste der Argumente zugewiesen. Die Liste der Argumente kann nicht leer sein.

Ausdruck	Syntax	Rückgabe

foreach	`foreach variable in( argument list )do`
	`expression list`
	`endfor`
	Hinweis: Verwenden Sie ein Komma (,), um einfache Ausdrücke in der Liste der Argumente voneinander zu trennen.
		Der Wert der letzten Ausdrucksliste, die ausgewertet wurde, oder null (0), wenn die Schleife nicht in Gang gesetzt wurde.

Im folgenden Beispiel werden der foreach-Dropdown-Liste nur die Werte der XML-Elemente „display“ hinzugefügt.

```
foreach Item in (xfa.record.lists.list1.display[*]) do
$.addItem(Item,"")
endfor
```

break-Ausdrücke

Ein break-Ausdruck löst das unmittelbare Verlassen der innersten umschließenden while-, for- oder foreach-Ausdrucksschleife aus. Die Steuerung wird an den Ausdruck übergeben, welcher der beendeten Schleife folgt.

Der Wert des break-Ausdrucks ist immer der Wert null (0).

Ausdruck	Syntax	Rückgabe
break	`break`	Übergibt die Steuerung an den Ausdruck, welcher der beendeten Schleife folgt.

Im folgenden Beispiel wird eine if-Bedingung in der while-Schleife platziert, um zu prüfen, ob der aktuelle Wert gleich „Display data for 2“ ist. Falls TRUE, wird der break-Ausdruck ausgeführt und verhindert, dass die Schleife fortgesetzt wird.

```
var List = ref(xfa.record.lists.list1)
var i=0
while (List.nodes.item(i+1).value ne "3") do
$.addItem(List.nodes.item(i).value,List.nodes.item(i+1).value)
i = i + 2
if (List.nodes.item(i) eq "Display data for 2" then
break
endif
endwhile
```

continue-Ausdrücke

Ein continue-Ausdruck löst die nächste Iteration der innersten umschließenden while-, for- oder foreach-Schleife aus.

Der Wert des continue-Ausdrucks ist immer der Wert null (0).

Ausdruck	Syntax	Rückgabe
continue	`continue`	Wenn dieser Ausdruck in einem while-Ausdruck verwendet wird, wird die Steuerung an die while-Bedingung übergeben. Wenn dieser Ausdruck in einem for-Ausdruck verwendet wird, wird die Steuerung an den step-Ausdruck übergeben.

Das Ziel des folgenden Beispiels ist es, die Dropdown-Liste mit Werten aus der XML-Datei zu füllen. Wenn der Wert des aktuellen XML-Elements „Display data for 3“ ist, wird die while-Schleife über den break-Ausdruck beendet. Wenn der Wert des aktuellen XML-Elements „Display data for 2“ ist, fügt das Skript der Variablen `i` (dem Zähler) „2“ hinzu und die Schleife beginnt sofort ihren nächsten Zyklus. Die letzten beiden Zeilen werden ignoriert, wenn der Wert des aktuellen XML-Elements „Display data for 2“ ist.

```
var List = ref(xfa.record.lists.list1)
var i = 0
while (List.nodes.item(i+1).value ne "5") do
if (List.nodes.item(i) eq "Display data for 3") then
break
endif
if (List.nodes.item(i) eq "Display data for 2" then
i=i+2
continue
endif
$.addItem(List.nodes.item(i).value,List.nodes.item(i+1).value)
i=i+2
endwhile
```

Variablen

FormCalc bietet Ihnen die Möglichkeit, innerhalb von Berechnungen Variablen für die Speicherung von Daten zu erstellen und zu manipulieren. Der Name, den Sie jeder erstellten Variablen zuweisen, muss ein eindeutiger Bezeichner sein.

Beispiel: Die folgenden FormCalc-Ausdrücke definieren die Variable „userName“ und legen fest, dass der Wert eines Textfelds dem Wert von „userName“ entsprechen soll.

```
var userName = "Tony Blue"
TextField1.rawValue = userName
```

Sie können Variablen, die Sie auf der Registerkarte „Variablen“ des Dialogfelds „Formulareigenschaften“ definieren, auf die gleiche Weise referenzieren. Der folgende FormCalc-Ausdruck verwendet die Concat-Funktion, um mit den Formularvariablen „salutation“ und „name“ den Wert des Textfelds festzulegen.

```
TextField1.rawValue = Concat(salutation, name)
```

Hinweis: Eine Variable, die Sie mit FormCalc erstellen, ersetzt eine gleichnamige Variable, die Sie auf dem Register „Variablen“ des Dialogfelds „Formulareigenschaften“ definiert haben.

Referenz-Syntax

FormCalc ermöglicht über eine Referenz-Syntax den Zugriff auf die Eigenschaften und Werte von Objekten, die im Formularentwurf verwendet werden. Das folgende Beispiel zeigt, wie Objektwerte sowohl zugewiesen als auch abgerufen werden können:

```
Invoice_Total.rawValue = Invoice_SubTotal.rawValue * (8 / 100)
```

In diesem Fall weist die Referenz-Syntax `Invoice_Total` den Wert `Invoice_SubTotal * (8 / 100)` dem Feld `Invoice_Total` zu.

Beim Formularentwurf ermöglicht eine vollständig qualifizierte Referenz-Syntax den Zugriff auf alle Objekte im Formularentwurf.

Mit Hilfe einer Reihe von Kurzbefehlen lassen sich Referenzen in FormCalc bequemer erstellen, wodurch der Zugriff auf Objekteigenschaften und -werte erleichtert wird. Die folgende Tabelle bietet einen Überblick über die für FormCalc verfügbaren Referenz-Syntax-Kurzbefehle.

Notation	Beschreibung
`$`	Bezeichnet das aktuelle Feld oder Objekt. Beispiel: `$ = "Tony Blue"` Das obige Beispiel setzt den Wert des aktuellen Feldes oder Objekts auf `Tony Blue`.

`$data`	Bezeichnet den Stammknoten des Datenmodells `xfa.datasets.data`. Beispiel:
	`$data.purchaseOrder.total`
	entspricht
	`xfa.datasets.data.purchaseOrder.total`

Notation	Beschreibung

`$event`	Bezeichnet das aktuelle Formularobjekt-Ereignis. Beispiel:
	`$event.name`
	entspricht
	`xfa.event.name`

`$form`	Bezeichnet den Stammknoten des Formularmodells `xfa.form`. Beispiel:
	`$form.purchaseOrder.tax`
	entspricht
	`xfa.form.purchaseOrder.tax`

`$host`	Bezeichnet das Host-Objekt. Beispiel:
	`$host.messageBox("Hello world")`
	entspricht
	`xfa.host.messageBox("Hello world")`

`$layout`	Bezeichnet den Stammknoten des Layoutmodells `xfa.layout`. Beispiel:
	`$layout.ready`
	entspricht
	`xfa.layout.ready`

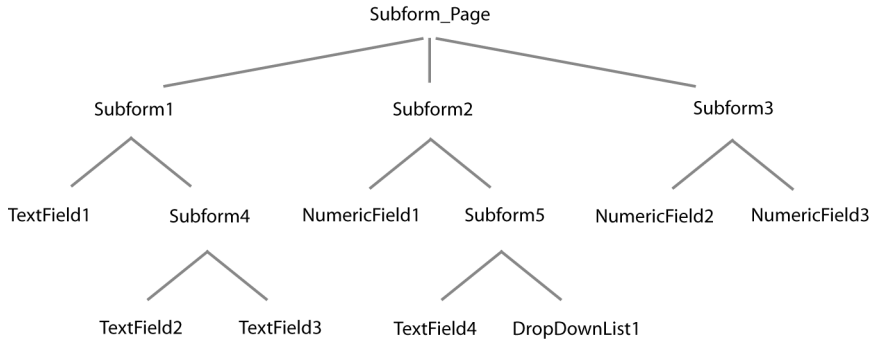
`$record`	Bezeichnet den aktuellen Datensatz aus einer Zusammenstellung von Daten, z. B. aus einer XML-Datei. Beispiel: `$record.header.txtOrderedByCity` referenziert den Knoten `txtOrderedByCity` im Knoten `header` der aktuellen XML-Daten.

`$template`	Bezeichnet den Stammknoten des Vorlagenmodells `xfa.template`. Beispiel:
	`$template.purchaseOrder.item`
	entspricht
	`xfa.template.purchaseOrder.item`

`!`	Bezeichnet den Stammknoten des Datenmodells `xfa.datasets`. Beispiel:
	`!data`
	entspricht
	`xfa.datasets.data`

`*`	Wählt alle Formularobjekte innerhalb eines gegebenen Containers, z. B. in einem Teilformular, unabhängig von ihrem Namen bzw. alle Objekte mit einem ähnlichen Namen aus. Beispielsweise wählt der folgende Ausdruck alle Objekte namens `item` in einem Formular aus: `xfa.form.form1.item[*]`

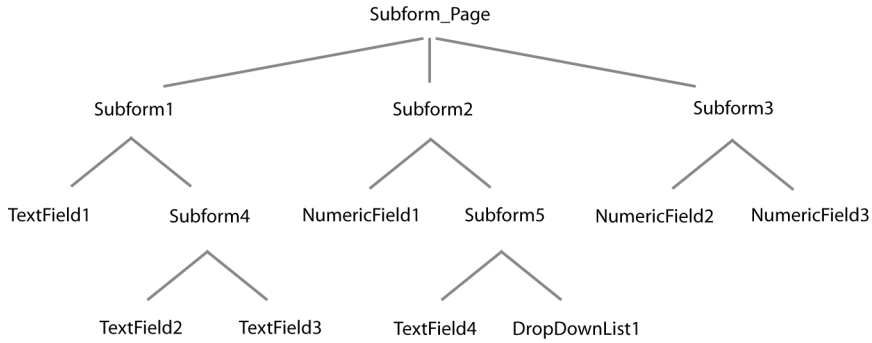
Notation	Beschreibung

`..`	Sie können an jeder Stelle in Ihrer Referenz-Syntax zwei Punkte einsetzen, um nach Objekten zu suchen, die Teil eines Unter-Containers des aktuellen Container-Objekts (z. B. eines Teilformulars) sind. Beispielsweise bedeutet der Ausdruck `Subform_Page..Subform2`: Knoten `Subform_Page` lokalisieren (wie üblich) und einen `Subform_Page` untergeordneten Knoten namens `Subform2` suchen.
	 
	
	Aus dem obigen Beispiel-Baumdiagramm ergibt sich:
	`Subform_Page..TextField2`
	entspricht
	`Subform_Page.Subform1[0].Subform3.TextField2[0]`
	da sich `TextField2[0]` im ersten `Subform1`-Knoten befindet, auf den FormCalc bei der Suche stößt. Ein zweites Beispiel:
	`Subform_Page..Subform3[*]`
	gibt alle vier Instanzen des Objekts `TextField2` zurück.

#	Das Nummernzeichen (#) dient zur Bezeichnung eines der folgenden Elemente in einer Referenz-Syntax:
	• Ein unbenanntes Objekt. Beispielsweise greift die folgende Referenz-Syntax auf ein unbenanntes Teilformular zu:
	`xfa.form.form1.#subform`
	• Geben Sie in einer Referenz-Syntax eine Eigenschaft an, wenn eine Eigenschaft und ein Objekt den gleichen Namen tragen. Beispielsweise greift die folgende Referenz-Syntax auf die Eigenschaft `name` eines Teilformulars zu, wenn das Teilformular auch ein Feld namens `name` hat:
	`xfa.form.form1.#subform.#name`

Notation	Beschreibung

`[ ]`	Die eckige Klammer (`[ ]`) gibt den Vorkommenswert eines Objekts an. Um eine Referenz mit einem solchen Wert zu erstellen, setzen Sie eckige Klammern (`[ ]`) hinter einen Objektnamen und fügen Sie zwischen den eckigen Klammern einen der folgenden Werte ein:
	• `[ n ]`, wobei `n` eine absolute Vorkommens-Indexnummer ist (der Index beginnt bei 0). Wenn eine Vorkommensnummer außerhalb des zulässigen Bereichs liegt, wird ein Fehler zurückgegeben. Beispiel:
	`xfa.form.form1.#subform.Quantity[3]`
	`refers to the fourth occurrence of the Quantity object.`
	• `[ +/- n ]`, wobei `n` ein Vorkommen relativ zum Vorkommen des Objekts angibt, von welchem die Referenz ausgeht. Positive Werte liefern höhere Vorkommensnummern und negative Werte liefern niedrigere. Beispiel:
	`xfa.form.form1.#subform.Quantity[+2]`
	Diese Referenz liefert das Vorkommen von `Quantity`, dessen Vorkommensnummer um 2 höher ist als die Vorkommensnummer des Containers, von welchem die Referenz ausgeht. Stünde diese Referenz beispielsweise mit dem Objekt `Quantity[2]` in Verbindung, so wäre sie gleichbedeutend mit:
	`xfa.template.Quantity[4]`
	Wenn die berechnete Indexnummer außerhalb des zulässigen Bereichs liegt, gibt die Referenz einen Fehler zurück.
	Die häufigste Anwendung dieser Syntax ist die Ansteuerung des vorigen oder nächsten Vorkommens eines bestimmten Objekts. Beispielsweise könnte man bei jedem Vorkommen des Objekts `Quantity` (außer beim ersten) mit `Quantity[-1]` den Wert des vorigen `Quantity`-Objekts abrufen.
	• `[*]` gibt mehrere Vorkommen eines Objekts an. Das erste benannte Objekt wird gefunden und die mit dem ersten Objekt verwandten Objekte desselben Namens werden zurückgegeben. Beachten Sie, dass bei dieser Notation eine Zusammenstellung von mehreren Objekten zurückgegeben wird. Beispiel:
	`xfa.form.form1.#subform.Quantity[*]`
	Dieser Ausdruck bezieht sich auf alle Objekte mit dem Namen `Quantity`, welche mit dem ersten von der Referenz gefundenen Vorkommen von `Quantity` verwandt sind.
	Hinweis: In sprachspezifischen Formularen für Arabisch, Hebräisch, Thailändisch und Vietnamesisch befindet sich die Referenz-Syntax grundsätzlich rechts (auch bei Sprachen, die von rechts nach links geschrieben werden).
	Hinweis: LiveCycle Forms 7.1 bietet keine Unterstützung für Formularentwürfe mit arabischen, hebräischen, thailändischen und vietnamesischen Schriftzeichen.

Notation	Beschreibung
`[ ]` `(Fortsetzung)`	 In dem gezeigten Baumdiagramm geben diese Ausdrücke jeweils die folgenden Objekte zurück: • `Subform_Page.Subform1[*]` gibt beide `Subform1`-Objekte zurück. • `Subform_Page.Subform1.Subform3.TextField2[*]` gibt die beiden `TextField2`-Objekte zurück. `Subform_Page.Subform1` wird zum ersten `Subform1`-Objekt auf der linken Seite aufgelöst und `TextField2[*]` wird relativ zum Objekt `Subform3` ausgewertet. • `Subform_Page.Subform1[*].TextField1` gibt beide `TextField1`-Instanzen zurück. `Subform_Page.Subform1[*]` wird zu den beiden `Subform1`-Objekten aufgelöst und `TextField1` wird relativ zu den `Subform1`-Objekten ausgewertet. • `Subform_Page.Subform1[*].Subform3.TextField2[1]` gibt das zweite und vierte `TextField2`-Objekt von links zurück. `Subform_Page.Subform1[*]` wird zu den beiden `Subform1`-Objekten aufgelöst und `TextField2[1]` wird relativ zu den `Subform3`-Objekten ausgewertet. • `Subform_Page.Subform1[*].Subform3[*]` gibt beide Instanzen des `Subform3`-Objekts zurück. • `Subform_Page.*` gibt beide `Subform1`-Objekte und das `Subform2`-Objekt zurück. • `Subform_Page.Subform2.*` gibt die beiden Instanzen des `NumericField2`-Objekts zurück.

Eigenschaften- und Methodenaufrufe

LiveCycle Designer definiert für alle Objekte in einem Formularentwurf eine Vielzahl von Eigenschaften und Methoden. Mit FormCalc können Sie auf diese Eigenschaften und Methoden zugreifen und so das Aussehen und Verhalten von Formularobjekten gezielt ändern. Ähnlich wie bei einem Funktionsaufruf können Sie Eigenschaften und Methoden aufrufen, indem Sie ihnen Argumente in einer bestimmten Reihenfolge übergeben. Wie viele und welche Argumente eine Eigenschaft oder Methode enthält, hängt vom jeweiligen Objekttyp ab.

Hinweis: Die verschiedenen Formularentwurfsobjekte unterstützen jeweils unterschiedliche Eigenschaften und Methoden. Eine vollständige Liste der von Objekten unterstützten Eigenschaften und Methoden finden Sie in *Adobe XML Form Object Model Reference* im LiveCycle Designer Developer Center unter www.adobe.com/devnet/livecycle/designing_forms.html.

Integrierte Funktionsaufrufe

FormCalc unterstützt eine große Zahl integrierter Funktionen für eine umfangreiche Palette von Aufgaben. Bei den Namen der Funktionen wird die Groß-/Kleinschreibung nicht beachtet, aber im Gegensatz zu den Schlüsselwörtern sind die Namen der Funktionen in FormCalc nicht reserviert. Das heißt, dass bei Berechnungen in Formularen mit Objekten, deren Namen mit den Namen von FormCalc-Funktionen übereinstimmen, keine Konflikte auftreten.

Manche Funktionen benötigen einen bestimmten Satz von Argumenten für die Ausführung und zur Rückgabe eines Wertes, andere nicht. Viele Funktionen besitzen optionale Argumente; das heißt, es liegt beim Benutzer, zu entscheiden, ob das Argument in der betreffenden Situation benötigt wird oder nicht.

FormCalc wertet alle Funktionsargumente der Reihe nach aus, beginnend mit dem führenden Argument. Wenn versucht wird, einer Funktion weniger als die benötigte Zahl von Argumenten zu übergeben, erzeugt die Funktion einen Ausnahmefehler.

Jede Funktion erwartet die einzelnen Argumente in einem bestimmten Format, d. h. entweder als Zahlen- oder als Zeichenfolgenliteral. Wenn der Wert eines Arguments nicht mit dem von der Funktion erwarteten Format übereinstimmt, wandelt FormCalc den Wert entsprechend um. Beispiel:

```
Len(35)
```

Die Funktion Len erwartet ein Zeichenfolgenliteral. In diesem Fall wandelt FormCalc das Argument aus der Zahl 35 in die Zeichenfolge „35“ um und die Funktion wird als 2 ausgewertet.

Die Umwandlung eines Zeichenfolgenliterals in ein Zahlenliteral ist dagegen weniger einfach. Beispiel:

```
Abs("abc")
```

Die Funktion Abs erwartet ein Zahlenliteral. FormCalc wandelt den Wert aller Zeichenfolgenliterale in 0 um. Dies kann bei Funktionen wie z. B. Apr problematisch sein, bei denen der Wert 0 einen Fehler verursacht.

Einige Funktionsargumente benötigen ausschließlich ganzzahlige Werte; in diesen Fällen werden die übergebenen Argumente stets durch Abschneiden des Bruchanteils in Ganzzahlen umgewandelt.

Die wichtigsten Eigenschaften der integrierten Funktionen im Überblick:

- Bei den Namen der integrierten Funktionen wird die Groß-/Kleinschreibung nicht berücksichtigt.
- Die integrierten Funktionen sind vordefiniert, ihre Namen sind aber keine reservierten Wörter. Das heißt, es kann in keinem Fall ein Konflikt zwischen der integrierten Funktion Max und einem Objekt, einer Objekteigenschaft oder einer Objektmethode namens Max auftreten.
- Viele integrierte Funktionen besitzen eine obligatorische Zahl von Argumenten, auf die eine optionale Zahl weiterer Argumente folgen kann.
- Einige integrierte Funktionen, Avg, Count, Max, Min, Sum und Concat, akzeptieren beliebig viele Argumente.

Eine vollständige Aufstellung sämtlicher Funktionen in FormCalc enthält die „Alphabetische Liste der Funktionen“ auf Seite 30.

3 Alphabetische Liste der Funktionen

Die folgende Tabelle enthält alle verfügbaren FormCalc-Funktionen, liefert eine Beschreibung für jede Funktion und gibt den Kategorietyp an, zu dem die jeweilige Funktion gehört.

Funktion	Beschreibung	Typ
„Abs“ auf Seite 35	Gibt den Betragswert eines Zahlenwerts oder Ausdrucks zurück.	Arithmetik
„Apr“ auf Seite 65	Gibt die jährliche Gesamtbelastung für einen Kredit zurück.	Finanz
„At“ auf Seite 85	Findet die Anfangs-Zeichenposition einer Zeichenfolge innerhalb einer anderen Zeichenfolge.	Zeichenfolge
„Avg“ auf Seite 36	Wertet einen Satz von Zahlenwerten und/oder Ausdrücken aus und gibt den Mittelwert der von null verschiedenen Elemente innerhalb dieses Satzes zurück.	Arithmetik
„Ceil“ auf Seite 37	Gibt die ganze Zahl zurück, die größer oder gleich einer angegebenen Zahl ist.	Arithmetik
„Choose“ auf Seite 76	Wählt einen Wert aus einem gegebenen Satz von Parametern.	Logisch
„Concat“ auf Seite 85	Gibt die Verkettung von zwei oder mehr Zeichenfolgen zurück.	Zeichenfolge
„Count“ auf Seite 37	Wertet einen Satz von Werten und/oder Ausdrücken aus und gibt die Anzahl der von null verschiedenen Elemente innerhalb dieses Satzes zurück.	Arithmetik
„CTerm“ auf Seite 66	Gibt die Anzahl der Perioden zurück, die erforderlich sind, damit eine Anlage mit einer festen Verzinsung mit Zinseszins auf einen Endwert anwächst.	Finanz
„Date“ auf Seite 54	Gibt das aktuelle Systemdatum als Anzahl von Tagen seit Beginn der Epoche zurück.	Datum und Uhrzeit
„Date2Num“ auf Seite 54	Gibt für eine angegebene Datums-Zeichenfolge die Anzahl von Tagen seit Beginn der Epoche zurück.	Datum und Uhrzeit
„DateFmt“ auf Seite 55	Gibt für einen angegebenen Datumsformat-Stil eine Datumsformat-Zeichenfolge zurück.	Datum und Uhrzeit
„Decode“ auf Seite 86	Gibt die dekodierte Version einer angegebenen Zeichenfolge zurück.	Zeichenfolge

Funktion	Beschreibung	Typ
„Encode“ auf Seite 87	Gibt die kodierte Version einer angegebenen Zeichenfolge zurück.	Zeichenfolge
„Eval“ auf Seite 80	Gibt den Wert einer angegebenen Formular-berechnung zurück.	Sonstiges
„Exists“ auf Seite 77	Bestimmt, ob der angegebene Parameter eine Referenz-Syntax zu einem vorhandenen Objekt ist.	Logisch
„Floor“ auf Seite 38	Gibt die größte Ganzzahl zurück, die kleiner oder gleich dem angegebenen Wert ist.	Arithmetik
„Format“ auf Seite 88	Formatiert die angegebenen Daten gemäß der angegebenen Musterformat-Zeichenfolge.	Zeichenfolge
„FV“ auf Seite 67	Gibt den Endwert von Anlagebeträgen zurück, die regelmäßig und in gleich bleibender Höhe bei einem konstanten Zinssatz eingezahlt werden.	Finanz
„Get“ auf Seite 101	Lädt den Inhalt der angegebenen URL herunter.	URL
„HasValue“ auf Seite 78	Ermittelt, ob der angegebene Parameter ein Akzessor mit einem von null verschiedenen, nicht leeren oder vom Leerzeichen verschiedenen Wert ist.	Logisch
„IPmt“ auf Seite 68	Gibt den Zinsbetrag zurück, der in einer bestimmten Zeitspanne für einen Kredit gezahlt wurde.	Finanz
„IsoDate2Num“ auf Seite 56	Gibt für eine angegebene gültige Datums-Zeichen-folge die Anzahl der Tage seit Beginn der Epoche zurück.	Datum und Uhrzeit
„IsoTime2Num“ auf Seite 56	Gibt für eine angegebene gültige Uhrzeit-Zeichenfolge die Anzahl der Millisekunden seit Beginn der Epoche zurück.	Datum und Uhrzeit
„Left“ auf Seite 89	Extrahiert eine angegebene Anzahl von Zeichen aus einer Zeichenfolge, beginnend beim ersten Zeichen auf der linken Seite.	Zeichenfolge
„Len“ auf Seite 89	Gibt die Anzahl der Zeichen in einer angegebenen Zeichenfolge zurück.	Zeichenfolge
„LocalDateFmt“ auf Seite 57	Gibt eine lokalisierte Datumsformat-Zeichenfolge mit dem angegebenen Datumsformat-Stil zurück.	Datum und Uhrzeit
„LocalTimeFmt“ auf Seite 58	Gibt eine lokalisierte Uhrzeitformat-Zeichenfolge mit dem angegebenen Uhrzeitformat-Stil zurück.	Datum und Uhrzeit
„Lower“ auf Seite 90	Wandelt alle Großbuchstaben in einer angegebenen Zeichenfolge in Kleinbuchstaben um.	Zeichenfolge

Funktion	Beschreibung	Typ
„Ltrim“ auf Seite 91	Gibt eine Zeichenfolge zurück, bei der alle Leerraum-Zeichen am Anfang entfernt wurden.	Zeichenfolge
„Max“ auf Seite 39	Gibt den Maximalwert der von null verschiedenen Elemente im angegebenen Satz von Zahlen zurück.	Arithmetik
„Min“ auf Seite 40	Gibt den Minimalwert der von null verschiedenen Elemente im angegebenen Satz von Zahlen zurück.	Arithmetik
„Mod“ auf Seite 41	Gibt den Modulus der Division einer Zahl durch eine andere zurück.	Arithmetik
„NPV“ auf Seite 69	Gibt den Kapitalwert einer Anlage auf der Grundlage eines Diskontsatzes und einer Folge von zukünftigen periodischen Cashflows zurück.	Finanz
„Null“ auf Seite 81	Gibt den Null-Wert zurück. („Null-Wert“ ist gleichbedeutend mit „kein Wert“.)	Sonstiges
„Num2Date“ auf Seite 59	Gibt für eine angegebene Anzahl von Tagen seit Beginn der Epoche eine Datums-Zeichenfolge zurück.	Datum und Uhrzeit
„Num2GMTime“ auf Seite 60	Gibt für eine angegebene Anzahl von Millisekunden seit Beginn der Epoche eine GMT-Uhrzeit-Zeichenfolge zurück.	Datum und Uhrzeit
„Num2Time“ auf Seite 61	Gibt für eine angegebene Anzahl von Millisekunden seit Beginn der Epoche eine Uhrzeit-Zeichenfolge zurück.	Datum und Uhrzeit
„Oneof“ auf Seite 78	Gibt TRUE (1) zurück, wenn sich ein Wert in einem angegebenen Satz befindet, bzw. FALSE (0), wenn dies nicht der Fall ist.	Logisch
„Parse“ auf Seite 91	Analysiert die angegebenen Daten gemäß dem angegebenen Musterformat.	Zeichenfolge
„Pmt“ auf Seite 70	Gibt die Höhe des Rückzahlungsbetrags für einen Kredit bei konstanten Zahlungsbeträgen und konstantem Zinssatz zurück.	Finanz
„Post“ auf Seite 102	Sendet die angegebenen Daten an die genannte URL.	URL
„PPmt“ auf Seite 71	Gibt den Tilgungsbetrag zurück, der in einer bestimmten Zeitspanne für einen Kredit gezahlt wurde.	Finanz
„Put“ auf Seite 104	Lädt die angegebenen Daten zu der genannten URL hoch.	URL
„PV“ auf Seite 72	Gibt den Gegenwartswert einer Anlage mit regelmäßigen konstanten Einzahlungen und konstantem Zinssatz zurück.	Finanz

Funktion	Beschreibung	Typ
„Rate“ auf Seite 73	Gibt den Zinssatz pro Verzinsungsperiode zurück, der benötigt wird, damit eine Anlage mit Zinseszins in einem gegebenen Zeitraum von einem gegebenen Gegenwartswert auf einen Endwert anwächst.	Finanz
„Ref“ auf Seite 81	Gibt eine Referenz auf ein vorhandenes Objekt zurück.	Sonstiges
„Replace“ auf Seite 92	Ersetzt innerhalb einer angegebenen Zeichenfolge alle Fundstellen einer Zeichenfolge durch eine andere.	Zeichenfolge
„Right“ auf Seite 93	Extrahiert mehrere Zeichen aus einer angegebenen Zeichenfolge, beginnend beim letzten Zeichen auf der rechten Seite.	Zeichenfolge
„Round“ auf Seite 42	Wertet einen angegebenen Zahlenwert oder Ausdruck aus und gibt eine auf die angegebene Anzahl von Dezimalstellen gerundete Zahl zurück.	Arithmetik
„Rtrim“ auf Seite 93	Gibt eine Zeichenfolge zurück, bei der alle Leerraum-Zeichen am Ende entfernt wurden.	Zeichenfolge
„Space“ auf Seite 94	Gibt eine Zeichenfolge zurück, die aus einer angegebenen Anzahl von Leerzeichen besteht.	Zeichenfolge
„Str“ auf Seite 95	Wandelt eine Zahl in eine Zeichenfolge um. FormCalc formatiert das Ergebnis auf die angegebene Breite und rundet auf die angegebene Zahl von Dezimalstellen.	Zeichenfolge
„Stuff“ auf Seite 96	Fügt eine Zeichenfolge in eine andere Zeichenfolge ein.	Zeichenfolge
„Substr“ auf Seite 97	Extrahiert einen Abschnitt aus einer angegebenen Zeichenfolge.	Zeichenfolge
„Sum“ auf Seite 43	Gibt die Summe der von null verschiedenen Elemente eines gegebenen Satzes von Zahlen zurück.	Arithmetik
„Term“ auf Seite 74	Gibt die Anzahl der Perioden zurück, die erforderlich sind, um mit konstanten periodischen Einzahlungen auf ein verzinstes Konto einen angegebenen Endwert zu erzielen.	Finanz
„Time“ auf Seite 62	Gibt die aktuelle Systemuhrzeit als Anzahl von Millisekunden seit Beginn der Epoche zurück.	Datum und Uhrzeit
„Time2Num“ auf Seite 62	Gibt für eine angegebene Uhrzeit-Zeichenfolge die Anzahl der Millisekunden seit Beginn der Epoche zurück.	Datum und Uhrzeit
„TimeFmt“ auf Seite 63	Gibt ein Uhrzeitformat in einem angegebenen Uhrzeitformat-Stil zurück.	Datum und Uhrzeit

Funktion	Beschreibung	Typ
„UnitType“ auf Seite 82	Gibt die Einheit einer Maßangabe zurück. Eine Maßangabe ist eine Zeichenfolge, die aus einer Zahl und einer nachfolgenden Einheitenbezeichnung besteht.	Sonstiges
„UnitValue“ auf Seite 83	Gibt den Zahlenwert einer Messung mit ihrer zugeordneten Maßangabe nach einer optionalen Einheitenumwandlung zurück.	Sonstiges
„Upper“ auf Seite 98	Wandelt alle Kleinbuchstaben in einer angegebenen Zeichenfolge in Großbuchstaben um.	Zeichenfolge
„Uuid“ auf Seite 98	Gibt eine UUID-Zeichenfolge (Universally Unique Identifier) zurück, die als Kennzeichnungsmethode verwendet werden soll.	Zeichenfolge
„Within“ auf Seite 79	Gibt TRUE (1) zurück, wenn der geprüfte Wert innerhalb eines angegebenen Bereichs liegt, bzw. FALSE (0), wenn dies nicht der Fall ist.	Logisch
„WordNum“ auf Seite 99	Gibt die englischsprachige Textentsprechung einer angegebenen Zahl zurück.	Zeichenfolge

4 Arithmetik-Funktionen

Diese Funktionen führen verschiedene mathematische Operationen aus.

Funktionen

Abs

Gibt den Betragswert eines Zahlenwerts oder Ausdrucks zurück bzw. liefert das Ergebnis null, wenn der Wert oder Ausdruck null ist.

Syntax

`Abs(`*`n1`*`)`

Parameter

Parameter	Beschreibung
`n1`	Ein auszuwertender Zahlenwert oder Ausdruck.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Abs`:

Ausdruck	Rückgabe
`Abs(1.03)`	`1.03`
`Abs(-1.03)`	`1.03`
`Abs(0)`	`0`

Avg

Wertet einen Satz von Zahlenwerten und/oder Ausdrücken aus und gibt den Mittelwert der von null verschiedenen Elemente innerhalb dieses Satzes zurück.

Syntax

`Avg(`*`n1`* `[, `*`n2`* `...])`

Parameter

Parameter	Beschreibung
`n1`	Der erste Zahlenwert oder Ausdruck des Satzes.
`n2` (optional)	Weitere Zahlenwerte oder Ausdrücke.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Avg`:

Ausdruck	Rückgabe
`Avg(0, 32, 16)`	`16`
`Avg(2.5, 17, null)`	`9.75`
`Avg(Price[0], Price[1], Price[2], Price[3])`	Der Mittelwert der vier ersten von null verschiedenen Vorkommen von `Price`.
`Avg(Quantity[*])`	Der Mittelwert aller von null verschiedenen Vorkommen von `Quantity`.

Ceil

Gibt die ganze Zahl zurück, die unmittelbar größer oder gleich einer gegebenen Zahl ist, oder liefert das Ergebnis null, wenn der Parameter null ist.

Syntax

`Ceil(n)`

Parameter

Parameter	Beschreibung
`n`	Ein beliebiger Zahlenwert oder Ausdruck. Die Funktion gibt `0` zurück, wenn `n` kein Zahlenwert oder Ausdruck ist.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Ceil`:

Ausdruck	Rückgabe
`Ceil(2.5875)`	`3`
`Ceil(-5.9)`	`-5`
`Ceil("abc")`	`0`
`Ceil(A)`	100, wenn der Wert von `A` 99.999 beträgt.

Count

Wertet einen Satz von Werten und/oder Ausdrücken aus und gibt die Anzahl der von null verschiedenen Elemente innerhalb dieses Satzes zurück.

Syntax

`Count(n1 [, n2 ...])`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck.
`n2` (optional)	Weitere Zahlenwerte und/oder Ausdrücke.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Count`:

Ausdruck	Rückgabe
`Count("Tony", "Blue", 41)`	`3`
`Count(Customers[*])`	Die Anzahl der von null verschiedenen Vorkommen von `Customers`.
`Count(Coverage[2], "Home", "Auto")`	`3`, sofern das dritte Vorkommen von `Coverage` von null verschieden ist.

Floor

Gibt die größte Ganzzahl zurück, die kleiner oder gleich dem angegebenen Wert ist.

Syntax

`Floor(`*n*`)`

Parameter

Parameter	Beschreibung
`n`	Ein beliebiger Zahlenwert oder Ausdruck.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Floor`:

Ausdruck	Rückgabe
`Floor(21.3409873)`	`21`
`Floor(5.999965342)`	`5`
`Floor(3.2 * 15)`	`48`

Max

Gibt den Maximalwert der von null verschiedenen Elemente im angegebenen Satz von Zahlen zurück.

Syntax

```
Max(n1 [, n2 ...])
```

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck.
`n2` (optional)	Weitere Zahlenwerte und/oder Ausdrücke.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Max`:

Ausdruck	Rückgabe
`Max(234, 15, 107)`	`234`
`Max("abc", 15, "Tony Blue")`	`15`
`Max("abc")`	`0`
`Max(Field1[*], Field2[0])`	Hier werden die von null verschiedenen Vorkommen von `Field1` sowie das erste Vorkommen von `Field2` ausgewertet und der höchste Wert wird zurückgegeben.
`Max(Min(Field1[*], Field2[0]), Field3, Field4)`	Hier werden die von null verschiedenen Vorkommen von `Field1` sowie das erste Vorkommen von `Field2` ausgewertet und der niedrigste Wert wird zurückgegeben. Das Endergebnis ist der aus dem Vergleich dieses Wertes mit den Werten für `Field3` und `Field4` ermittelte Höchstwert. Siehe auch „Min“ auf Seite 40.

Min

Gibt den Minimalwert der von null verschiedenen Elemente im angegebenen Satz von Zahlen zurück.

Syntax

`Min(`*`n1`* `[, `*`n2`* `...])`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck.
`n2` (optional)	Weitere Zahlenwerte und/oder Ausdrücke.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Min`:

Ausdruck	Rückgabe
`Min(234, 15, 107)`	`15`
`Min("abc", 15, "Tony Blue")`	`15`
`Min("abc")`	`0`
`Min(Field1[*], Field2[0])`	Hier werden die von null verschiedenen Vorkommen von `Sales_July` sowie das erste Vorkommen von `Sales_August` ausgewertet und der niedrigste Wert wird zurückgegeben.
`Min(Max(Field1[*], Field2[0]), Field3, Field4)`	Hier werden die von null verschiedenen Vorkommen von `Field1` sowie das erste Vorkommen von `Field2` ausgewertet und der höchste Wert wird zurückgegeben. Das Endergebnis ist der aus dem Vergleich dieses Wertes mit den Werten für `Field3` und `Field4` ermittelte Höchstwert. Siehe auch „Max“ auf Seite 39.

Mod

Gibt den Modulus der Division einer Zahl durch eine andere zurück. Der Modulus ist der Rest, der sich bei der Division des Dividenden durch den Divisor ergibt. Das Vorzeichen des Rests entspricht stets dem Vorzeichen des Dividenden.

Syntax

`Mod(n1, n2)`

Parameter

Parameter	Beschreibung
`n1`	Der Dividend, ein Zahlenwert oder Ausdruck.
`n2`	Der Divisor, ein Zahlenwert oder Ausdruck.

Wenn `n1` und/oder `n2` keine Zahlenwerte oder Ausdrücke sind, gibt die Funktion den Wert 0 zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Mod`:

Ausdruck	Rückgabe
`Mod(64, -3)`	`1`
`Mod(-13,3)`	`-1`
`Mod("abc", 2)`	`0`
`Mod(X[0], Y[9])`	Das erste Vorkommen von `X` wird als Dividend verwendet und das zehnte Vorkommen von `Y` als Divisor.
`Mod(Round(Value[4], 2), Max(Value[*]))`	Das fünfte Vorkommen von `Value`, auf zwei Dezimalstellen gerundet, wird als Dividend verwendet und das größte aller von null verschiedenen Vorkommen von `Value` wird als Divisor verwendet. Siehe auch „Max“ auf Seite 39 und „Round“ auf Seite 42.

Round

Wertet einen angegebenen Zahlenwert oder Ausdruck aus und gibt eine auf die angegebene Anzahl von Dezimalstellen gerundete Zahl zurück.

Syntax

`Round(n1 [, n2])`

Parameter

Parameter	Beschreibung
`n1`	Ein auszuwertender Zahlenwert oder Ausdruck.
`n2` (optional)	Die Anzahl von Dezimalstellen (maximal 12), mit der `n1` auszuwerten ist. Wenn Sie keinen Wert für `n2` angeben oder wenn `n2` ungültig ist, geht die Funktion davon aus, dass die Anzahl der Dezimalstellen 0 ist.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Round`:

Ausdruck	Rückgabe
`Round(12.389764537, 4)`	`12.3898`
`Round(20/3, 2)`	`6.67`
`Round(8.9897, "abc")`	`9`
`Round(FV(400, 0.10/12, 30*12), 2)`	`904195.17`. Hier wird der mit der Funktion `FV` ausgewertete Wert übernommen und auf zwei Dezimalstellen gerundet. Siehe auch „FV“ auf Seite 67.
`Round(Total_Price, 2)`	Rundet den Wert von `Total_Price` auf zwei Dezimalstellen ab.

Sum

Gibt die Summe der von null verschiedenen Elemente eines gegebenen Satzes von Zahlen zurück.

Syntax

`Sum(`*`n1`* `[, `*`n2`* `...])`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck.
`n2` (optional)	Weitere Zahlenwerte und/oder Ausdrücke.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Sum`:

Ausdruck	Rückgabe
`Sum(2, 4, 6, 8)`	`20`
`Sum(-2, 4, -6, 8)`	`4`
`Sum(4, 16, "abc", 19)`	`39`
`Sum(Amount[2], Amount[5])`	Summiert das dritte und sechste Vorkommen von `Amount`.
`Sum(Round(20/3, 2), Max(Amount[*]), Min(Amount[*]))`	Summiert den Wert von 20/3, auf zwei Dezimalstellen gerundet, sowie das größte und das kleinste der von null verschiedenen Vorkommen von `Amount`. Siehe auch „Max“ auf Seite 39, „Min“ auf Seite 40 und „Round“ auf Seite 42.

5 Datums- und Uhrzeitfunktionen

Die Funktionen in diesem Kapitel dienen speziell zur Erstellung und Bearbeitung von Datums- und Uhrzeitwerten.

Funktionen

Datum und Uhrzeit strukturieren

Gebietsschemata

Ein *Gebietsschema* ist ein Standardbegriff, der bei der Entwicklung internationaler Standards zur Identifizierung einer bestimmten Nationalität (Sprache und/oder Land) dient. In FormCalc legt ein *Gebietsschema* die für ein bestimmtes Land oder für eine bestimmte Region gültigen Datums-, Uhrzeit-, Zahlen- und Währungsformate fest, um zu gewährleisten, dass die Endbenutzer die gewohnten Formate verwenden können.

Jedes Gebietsschema besteht aus einer eindeutigen Zeichenfolge, der *Gebietsschema-Kennung*. Der Inhalt dieser Zeichenfolgen wird von der Internationalen Standardisierungsorganisation (ISO) Internet Engineering Task Force (IETF) kontrolliert, die eine Arbeitsgruppe der Internet Society ist (www.isoc.org).

Gebietsschema-Kennungen bestehen aus einem Sprachenteil, einem Länderteil oder beidem. Die folgende Tabelle enthält die gültigen Gebietsschemata für diese Version von LiveCycle Designer:

Sprache	Land	ISO-Code
Arabisch	Vereinigte Arabische Emirate	ar_AE
Arabisch	Bahrain	ar_BH

Sprache	Land	ISO-Code
Arabisch	Algerien	ar_DZ
Arabisch	Ägypten	ar_EG
Arabisch	Irak	ar_IQ
Arabisch	Jordanien	ar_JO
Arabisch	Kuwait	ar_KW
Arabisch	Libanon	ar_LB
Arabisch	Libyen	ar_LY
Arabisch	Marokko	ar_MA
Arabisch	Oman	ar_OM
Arabisch	Katar	ar_QA
Arabisch	Saudi-Arabien	ar_SA
Arabisch	Sudan	ar_SD
Arabisch	Syrien	ar_SY
Arabisch	Tunesien	ar_TN
Arabisch	Jemen	ar_YE
Bulgarisch	Bulgarien	bg_BG
Chinesisch	Hongkong	zh_HK
Chinesisch	Volksrepublik China (vereinfacht)	zh_CN
Chinesisch	Taiwan (traditionell)	zh_TW
Dänisch	Dänemark	da_DK
Deutsch	Österreich	de_AT
Deutsch	Deutschland	de_DE
Deutsch	Luxemburg	de_LU
Deutsch	Schweiz	de_CH
Englisch	Australien	en_AU
Englisch	Kanada	en_CA
Englisch	Indien	en_IN
Englisch	Irland	en_IE
Englisch	Neuseeland	en_NZ
Englisch	Südafrika	en_ZA

Sprache	**Land**	**ISO-Code**
Englisch	Großbritannien	en_GB
Englisch	Großbritannien (Europa)	en_GB_EURO
Englisch	Vereinigte Staaten von Amerika	en_US
Estnisch	Estland	et_EE
Finnisch	Finnland	fi_FI
Flämisch	Belgien	nl_BE
Französisch	Belgien	fr_BE
Französisch	Kanada	fr_CA
Französisch	Frankreich	fr_FR
Französisch	Luxemburg	fr_LU
Französisch	Schweiz	fr_CH
Griechisch	Griechenland	el_GR
Hebräisch	Israel	he_IL
Indonesisch	Indonesien	id_ID
Italienisch	Italien	it_IT
Italienisch	Schweiz	it_CH
Japanisch	Japan	ja_JP
Koreanisch	Republik Korea	ko_KR
Koreanisch	Korea (Hanja-Zeichen)	ko_KR_HANI
Kroatisch	Kroatien	hr_HR
Lettisch	Lettland	lv_LV
Litauisch	Litauen	lt_LT
Malaiisch	Malaysia	ms_MY
Niederländisch	Niederlande	nl_NL
Norwegisch (Bokmal)	Norwegen	nb_NO
Norwegisch (Nynorsk)	Norwegen	nn_NO
Polnisch	Polen	pl_PL
Portugiesisch	Brasilien	pt_BR
Portugiesisch	Portugal	pt_PT
Rumänisch	Rumänien	ro_RO
Russisch	Russland	ru_RU

Sprache	**Land**	**ISO-Code**
Schwedisch	Schweden	sv_SE
Serbokroatisch	Bosnien und Herzegowina	sh_BA
Serbokroatisch	Kroatien	sh_HR
Serbokroatisch	Serbien und Montenegro	sh_CS
Slowakisch	Slowakei	sk_SK
Slowenisch	Slowenien	sl_SI
Spanisch	Ecuador	es_EC
Spanisch	El Salvador	es_SV
Spanisch	Guatemala	es_GT
Spanisch	Honduras	es_HN
Spanisch	Nicaragua	es_NI
Spanisch	Panama	es_PA
Spanisch	Paraguay	es_PY
Spanisch	Puerto Rico	es_PR
Spanisch	Uruguay	es_UY
Spanisch	Argentinien	es_AR
Spanisch	Bolivien	es_BO
Spanisch	Chile	es_CL
Spanisch	Kolumbien	es_CO
Spanisch	Costa Rica	es_CR
Spanisch	Dominikanische Republik	es_DO
Spanisch	Mexiko	es_MX
Spanisch	Peru	es_PE
Spanisch	Spanien	es_ES
Spanisch	Venezuela	es_VE
Thailändisch	Thailand	th_TH
Thailändisch	Thailand, traditionell	th_TH_TH
Tschechisch	Tschechische Republik	cs_CAZ
Türkisch	Türkei	tr_TR
Ukrainisch	Ukraine	uk_UA

Sprache	Land	ISO-Code
Ungarisch	Ungarn	hu_HU
Vietnamesisch	Vietnam	vi_VN

In der Regel sind beide Elemente eines Gebietsschemas von Bedeutung. Beispielsweise werden die Namen der Wochentage und Monate in Englisch für Kanada und Großbritannien identisch formatiert, Datumsangaben aber unterschiedlich. Daher ist die Angabe der Sprache Englisch als Gebietsschema nicht ausreichend. Die Angabe nur des Landes als Gebietsschema ist ebenfalls unzureichend. So gibt es für Kanada unterschiedliche Datumsformate für Englisch und Französisch.

Grundsätzlich wird jede Anwendung in einer Umgebung eingesetzt, in der ein Gebietsschema vorhanden ist. Dieses Gebietsschema wird als *Umgebungsgebietsschema* bezeichnet. Unter bestimmten Umständen wird eine Anwendung auf einem System oder in einer Umgebung ausgeführt, in der kein Gebietsschema vorhanden ist. In diesen seltenen Fällen wird das Umgebungsgebietsschema auf den Standardwert Englisch, USA gesetzt (en_US). Dieses Gebietsschema wird als *Standardgebietsschema* bezeichnet.

Epoche

Sowohl den Datums- als auch den Uhrzeitwerten ist ein Ursprung oder eine *Epoche* zugeordnet, d. h. ein Zeitpunkt, an dem die Zeitrechnung begonnen wurde. Alle Datums- und Uhrzeitwerte, die vor dieser Epoche liegen, sind ungültig.

Die Werteinheit für Datumsfunktionen ist die seit Beginn der Epoche verstrichene Anzahl von Tagen. Die Werteinheit für Uhrzeitfunktionen ist die seit Beginn der Epoche verstrichene Anzahl von Millisekunden.

In LiveCycle Designer ist der erste Tag der Epoche für die Datumsfunktionen der 1. Januar 1900 und die erste Millisekunde der Epoche für die Uhrzeitfunktionen ist Mitternacht, 00:00:00, Greenwich Mean Time (GMT). Aus dieser Definition folgt, dass für Benutzer in Zeitzonen östlich der Greenwich Mean Time eventuell negative Uhrzeitwerte zurückgegeben werden.

Datumsformate

Ein *Datumsformat* gibt in Kurzform an, wie das Datum dargestellt wird. Es besteht aus mehreren Interpunktionszeichen und Symbolen, die das Format angeben, in dem das Datum dargestellt werden soll. Die folgende Tabelle zeigt einige Beispiele für Datumsformate:

Datumsformat	Beispiel
MM/DD/YY	11/11/78
DD/MM/YY	25/07/85
MMMM DD, YYYY	März 10, 1964

Das Datumsformat wird von einem ISO-Standard bestimmt. Jedes Land bzw. jede Region legt ihr eigenes Datumsformat fest. Dabei stehen eine kurze, eine mittlere, eine lange und eine vollständige Formatkategorie zur Auswahl. In der folgenden Tabelle sind Beispiele zu den unterschiedlichen Datumsformaten in verschiedenen Gebietsschemata für die einzelnen Kategorien aufgeführt.

Gebietsschema-Kennung und Beschreibung	Datumsformat (Kategorie)	Beispiel
en_GB Englisch (Großbritannien)	DD/MM/YY (kurz)	08/18/92 08/04/05
fr_CA Französisch (Kanada)	YY-MM-DD (mittel)	92-08-18
de_DE Deutsch (Deutschland)	D. MMMM YYYY (lang)	17. Juni 1989
fr_FR Französisch (Frankreich)	EEEE, 'le' D MMMM YYYY (vollständig)	Lundi, le 29 Octobre, 1990

Uhrzeitformate

Ein *Zeitformat* gibt in Kurzform an, wie die Uhrzeit formatiert wird. Es besteht aus Interpunktionszeichen, Literalen und Mustersymbolen. Die folgende Tabelle zeigt einige Beispiele für Uhrzeitformate:

Uhrzeitformat	Beispiel
h:MM A	19:15 Uhr
HH:MM:SS	21:35:26
HH:MM:SS 'o''clock' A Z	14:20:10 o’clock PM EDT

Uhrzeitformate werden von einer ISO-Norm bestimmt. Jedes Land legt die Form seiner Uhrzeitformate in einer Standard- sowie einer kurzen, mittleren, langen und vollständigen Variante fest. Im Gebietsschema wird das Format der Uhrzeiten festgelegt, die den Standards des betreffenden Landes entsprechen.

In der folgenden Tabelle sind einige Beispiele für die unterschiedlichen Datumsformate in verschiedenen Gebietsschemata für die einzelnen Kategorien aufgeführt.

Gebietsschema-Kennung und Beschreibung	Uhrzeitformat (Kategorie)	Beispiel
en_GB Englisch (Großbritannien)	HH:MM (kurz)	14:13
fr_CA Französisch (Kanada)	HH:MM:SS (mittel)	12:15:50
de_DE Deutsch (Deutschland)	HH:MM:SS z (lang)	14:13:13 -0400
fr_FR Französisch (Frankreich)	HH ‘h‘ MM Z (vollständig)	14 h 13 GMT-04:00

Datums- und Uhrzeit-Musterformate

Die folgenden Symbole müssen verwendet werden, um Datums- und Uhrzeitmuster für Datums-/Uhrzeitfelder zu erstellen. Bestimmte Datumssymbole werden nur in den Gebietsschemata für Chinesisch, Japanisch und Koreanisch verwendet. Diese Symbole werden unten auch angegeben.

Hinweis: Komma (,), Bindestrich (-), Doppelpunkt (:), Schrägstrich (/), Punkt (.) und Leerzeichen ( ) werden als Literalwerte behandelt und können an einer beliebigen Stelle in einem Muster vorkommen. Wenn eine Wortfolge in ein Muster aufgenommen werden soll, schließen Sie sie in einfache Anführungszeichen (') ein. Beispielsweise kann `'Your payment is due no later than' MM-DD-YY` als Anzeigemuster festgelegt werden.

Datums-symbol	**Beschreibung**	**Formatierter Wert für das Gebietsschema Englisch (USA). Dabei lautet der dem Gebietsschema entsprechende Eingabewert 1/1/08 (dies bedeutet „January 1, 2008“).**
`D`	1- oder 2-stelliger (1-31) Tag des Monats	1
`DD`	Mit 0 aufgefüllter 2-stelliger (01-31) Tag des Monats	01
`J`	1-, 2- oder 3-stelliger (1-366) Tag des Jahres	1
`JJJ`	Mit 0 aufgefüllter dreistelliger (001-366) Tag des Jahres	001
`M`	Ein- oder zweistelliger (1-12) Monat des Jahres	1
`MM`	Mit 0 aufgefüllter zweistelliger (01-12) Monat des Jahres	01
`MMM`	Abgekürzter Monatsname	Jan
`MMMM`	Vollständiger Monatsname	Januar
`E`	Einstelliger (1-7) Tag der Woche: 1=Sonntag	3 (weil der 1. Januar 2008 auf einen Dienstag fällt)
`EEE`	Abgekürzter Wochentag	Tue (weil der 1. Januar 2008 auf einen Dienstag [engl. „Tuesday“] fällt)
`EEEE`	Vollständiger Wochentag	Tuesday (weil der 1. Januar 2008 auf einen Dienstag [engl. „Tuesday“] fällt)
`YY`	Zweistellige Jahreszahl, wobei Zahlen unter 30 als Jahresangaben nach dem Jahr 2000 und Zahlen ab 30 als Jahresangaben vor dem Jahr 2000 zu verstehen sind. Beispiele: 00=2000, 29=2029, 30=1930 und 99=1999	08
`YYYY`	Vierstellige Jahreszahl	2008

Datums-symbol	**Beschreibung**	**Formatierter Wert für das Gebietsschema Englisch (USA). Dabei lautet der dem Gebietsschema entsprechende Eingabewert 1/1/08 (dies bedeutet „January 1, 2008“).**
G	Epochenname (v. Chr. oder n. Chr.)	AD (engl. für „n. Chr.“)
w	Einstellige (0-5) Wochennummer des Monats: Woche 1 ist die früheste auf einen Samstag endende Sequenz von vier aufeinander folgenden Tagen	1
WW	Zweistellige (01-53) Woche des Jahres gemäß ISO-8601: Woche 1 ist die Woche, die den 4. Januar enthält	01

Es sind verschiedene weitere Datumsmuster für die Darstellung von Daten in den Gebietsschemata für Chinesisch, Japanisch und Koreanisch verfügbar.

Japanische Epochen können anhand verschiedener Symbole dargestellt werden. Die letzten vier Epochensymbole sind alternative Symbole für die Darstellung japanischer Ären.

Datensymbol (Chinesisch, Japanisch, Koreanisch)	**Beschreibung**
DDD	Der ideografisch-numerisch festgelegte Tag des Monats im Gebietsschema.
DDDD	Der nach der Zehnerregel ideografisch-numerisch festgelegte Tag des Monats im Gebietsschema.
YYY	Das ideografisch-numerisch festgelegte Jahr im Gebietsschema.
YYYYY	Das nach der Zehnerregel ideografisch-numerisch festgelegte Jahr im Gebietsschema.
g	Der alternative Epochenname des Gebietsschemas. Für die gegenwärtige japanische Epoche Heisei zeigt dieses Muster das ASCII-Zeichen H (U+48) an.
gg	Der alternative Epochenname des Gebietsschemas. Für die gegenwärtige japanische Epoche zeigt dieses Muster das Kanji an, das dem Unicode-Symbol (U+5E73) entspricht.
ggg	Der alternative Epochenname des Gebietsschemas. Für die gegenwärtige japanische Epoche zeigt dieses Muster die Kanjis an, die den Unicode-Symbolen (U+5E73 U+6210) entsprechen.
g	Der alternative Epochenname des Gebietsschemas. Für die gegenwärtige japanische Epoche zeigt dieses Muster das ASCII-Zeichen H (U+FF28) in voller Breite an.
g g	Der alternative Epochenname des Gebietsschemas. Für die gegenwärtige japanische Epoche zeigt dieses Muster das Kanji an, das dem Unicode-Symbol (U+337B) entspricht.

Zeit-symbol	Beschreibung	Dem Gebietsschema entsprechender Eingabewert	Formatierter Wert für das Gebiets-schema Englisch (USA)
h	Ein- oder zweistellige (1-12) Stunde im 12-Stunden-System (AM/PM)	12:08 AM oder 2:08 PM	12 oder 2
hh	Mit 0 aufgefüllte zweistellige (01-12) Stunde im 12-Stunden-System (AM/PM)	12:08 AM oder 2:08 PM	12 oder 02
k	Ein- oder zweistellige (0-11) Stunde im 12-Stunden-System (AM/PM)	12:08 AM oder 2:08 PM	0 oder 2
kk	Zweistellige (00-11) Stunde im 12-Stunden-System (AM/PM)	12:08 AM oder 2:08 PM	00 oder 02
H	Ein- oder zweistellige (0-23) Stunde des Tages	12:08 AM oder 2:08 PM	0 oder 14
HH	Mit 0 aufgefüllte zweistellige (00-23) Stunde des Tages	12:08 AM oder 2:08 PM	00 oder 14
K	Ein- oder zweistellige (1-24) Stunde des Tages	12:08 AM oder 2:08 PM	24 oder 14
KK	Mit 0 aufgefüllte zweistellige (01-24) Stunde des Tages	12:08 AM oder 2:08 PM	24 oder 14
M	Ein- oder zweistellige (0-59) Minute der Stunde **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stundensymbol verwenden.	2:08 PM	8
MM	Mit 0 aufgefüllte zweistellige (00-59) Minute der Stunde **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stundensymbol verwenden.	2:08 PM	08
S	Ein- oder zweistellige (0-59) Sekunde der Minute **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stunden- und Minutensymbol verwenden.	2:08:09 PM	9
SS	Mit 0 aufgefüllte zweistellige (00-59) Sekunde der Minute **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stunden- und Minutensymbol verwenden.	2:08:09 PM	09

Zeit-symbol	Beschreibung	Dem Gebietsschema entsprechender Eingabewert	Formatierter Wert für das Gebiets-schema Englisch (USA)
FFF	Dreistellige (000-999) Tausendstelsekunde **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stunden-, Minuten- und Sekundensymbol verwenden.	2:08:09 PM	09
A	Der Zeitraum von Mitternacht bis Mittag (AM) bzw. von Mittag bis Mitternacht (PM)	2:08:09 PM	PM
z	Zeitzonenformat nach ISO-8601 (z. B. Z, +0500, -0030, -01, +0100) **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stundensymbol verwenden.	2:08:09 PM	-0400
zz	Alternatives ISO-8601 Zeitzonenformat (z. B. Z, +05:00, -00:30, -01, +01:00) **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stundensymbol verwenden.	2:08:09 PM	-04:00
Z	Abgekürzter Zeitzonenname (z. B. GMT, GMT+05:00, GMT-00:30, EST, PDT) **Hinweis:** Sie müssen dieses Symbol zusammen mit einem Stundensymbol verwenden.	2:08:09 PM	EDT

Reservierte Symbole

Die folgenden Symbole haben spezielle Bedeutungen und können nicht als Literalzeichen im Text verwendet werden.

?	Das Symbol entspricht einem beliebigen Zeichen. Wenn es für die Anzeige zusammengeführt wird, wandelt es sich zu einem Leerzeichen.
*	Das Symbol steht beim Senden für 0 oder für UNICODE-Leerzeichen. Wenn es für die Anzeige zusammengeführt wird, wandelt es sich zu einem Leerzeichen.
+	Das Symbol steht beim Senden für ein oder mehrere UNICODE-Leerzeichen. Wenn es für die Anzeige zusammengeführt wird, wandelt es sich zu einem Leerzeichen.

Date

Gibt das aktuelle Systemdatum als Anzahl von Tagen seit Beginn der Epoche zurück.

Syntax
`Date()`

Parameter

Keine

Beispiele

Die folgende Tabelle zeigt ein Beispiel für die Verwendung der Funktion `Date`:

Ausdruck	Rückgabe
`Date()`	`37875` (die Anzahl der Tage vom Beginn der Epoche bis zum 12. September 2003)

Date2Num

Gibt für eine angegebene Datums-Zeichenfolge die Anzahl von Tagen seit Beginn der Epoche zurück.

Syntax
`Date2Num(`*`d`* `[, `*`f`* `[, `*`k`* `]])`

Parameter

Parameter	Beschreibung
`d`	Eine Datums-Zeichenfolge in dem von `f` gelieferten Format, die auch dem von `k` angegebenen Gebietsschema entspricht.
`f` (optional)	Eine Datumsformat-Zeichenfolge. Wenn `f` nicht angegeben ist, wird das Standard-Datumsformat `MMM D, YYYY` verwendet.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn `k` nicht angegeben (oder ungültig) ist, wird das Umgebungsgebietsschema verwendet.

Die Funktion gibt den Wert `0` zurück, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Das Format des angegebenen Datums entspricht nicht dem in der Funktion angegebenen Format.
- Das in der Funktion angegebene Gebietsschema oder Datumsformat ist ungültig.

Es sind nicht genügend Informationen angegeben, um einen eindeutigen Tag seit Beginn der Epoche zu bestimmen (das heißt, die Datumsangaben fehlen oder sind unvollständig).

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Date2Num`:

Ausdruck	Rückgabe
`Date2Num("Mar 15, 1996")`	`35138`
`Date2Num("1/1/1900", "D/M/YYYY")`	`1`
`Date2Num("03/15/96", "MM/DD/YY")`	`35138`
`Date2Num("Aug 1,1996", "MMM D, YYYY")`	`35277`
`Date2Num("96-08-20", "YY-MM-DD", "fr_FR")`	`35296`
`Date2Num("1/3/00", "D/M/YY") - Date2Num("1/2/00", "D/M/YY")`	`29`

DateFmt

Gibt für einen angegebenen Datumsformat-Stil eine Datumsformat-Zeichenfolge zurück.

Syntax

`DateFmt([`*n* `[, `*k* `]])`

Parameter

Parameter	Beschreibung
`n` (optional)	Eine Ganzzahl, welche den durch das Gebietsschema festgelegten Datumsformat-Stil angibt wie folgt: • 1 (Kurzer Formatstil) • 2 (Mittlerer Formatstil) • 3 (Langer Formatstil) • 4 (Vollständiger Formatstil) Wenn `n` nicht angegeben (oder ungültig) ist, wird der Standardstilwert 0 verwendet.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn `k` nicht angegeben (oder ungültig) ist, wird das Umgebungsgebietsschema verwendet.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `DateFmt`:

Ausdruck	Rückgabe
`DateFmt(1)`	`M/D/YY` (bei Gebietsschema en_US)
`DateFmt(2, "fr_CA")`	`YY-MM-DD`
`DateFmt(3, "de_DE")`	`D. MMMM YYYY`
`DateFmt(4, "fr_FR")`	`EEEE D' MMMM YYYY`

IsoDate2Num

Gibt für eine angegebene gültige Datums-Zeichenfolge die Anzahl der Tage seit Beginn der Epoche zurück.

Syntax

`IsoDate2Num(d)`

Parameter

Parameter	Beschreibung
`d`	Eine gültige Datums-Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `IsoDate2Num`:

Ausdruck	Rückgabe
`IsoDate2Num("1900")`	`1`
`IsoDate2Num("1900-01")`	`1`
`IsoDate2Num("1900-01-01")`	`1`
`IsoDate2Num("19960315T20:20:20")`	`35138`
`IsoDate2Num("2000-03-01") - IsoDate2Num("20000201")`	`29`

IsoTime2Num

Gibt für eine angegebene gültige Uhrzeit-Zeichenfolge die Anzahl der Millisekunden seit Beginn der Epoche zurück.

Syntax

`IsoTime2Num(d)`

Parameter

Parameter	Beschreibung
`d`	Eine gültige Uhrzeit-Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `IsoTime2Num`:

Ausdruck	Rückgabe
`IsoTime2Num("00:00:00Z")`	`1`, für einen Benutzer in der nordamerikanischen Zeitzone Eastern Time (ET).
`IsoTime2Num("13")`	`64800001`, für einen Benutzer in Boston, USA.
`IsoTime2Num("13:13:13")`	`76393001`, für einen Benutzer in Kalifornien.
`IsoTime2Num("19111111T131313+01")`	`43993001`, für einen Benutzer in der nordamerikanischen Zeitzone Eastern Time (ET).

LocalDateFmt

Gibt eine lokalisierte Datumsformat-Zeichenfolge mit dem angegebenen Datumsformat-Stil zurück.

Syntax

`LocalDateFmt([n [, k ]])`

Parameter

Parameter	Beschreibung
`n` (optional)	Eine Ganzzahl, welche den durch das Gebietsschema festgelegten Datumsformat-Stil angibt wie folgt: • 1 (Kurzer Formatstil) • 2 (Mittlerer Formatstil) • 3 (Langer Formatstil) • 4 (Vollständiger Formatstil) Wenn `n` nicht angegeben (oder ungültig) ist, wird der Standardstilwert `0` verwendet.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn `k` nicht angegeben (oder ungültig) ist, wird das Umgebungsgebietsschema verwendet.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Funktion `LocalDateFmt`:

Ausdruck	Rückgabe
`LocalDateFmt(1, "de_DE")`	`tt.MM.uu`
`LocalDateFmt(2, "fr_CA")`	`aa-MM-jj`
`LocalDateFmt(3, "de_CH")`	`t. MMMM jjjj`
`LocalDateFmt(4, "fr_FR")`	`EEEE j MMMM aaaa`

LocalTimeFmt

Gibt eine lokalisierte Uhrzeitformat-Zeichenfolge mit dem angegebenen Uhrzeitformat-Stil zurück.

Syntax

```
LocalTimeFmt([n [, k ]])
```

Parameter

Parameter	Beschreibung
`n` (optional)	Eine Ganzzahl, welche den durch das Gebietsschema festgelegten Uhrzeitformat-Stil angibt wie folgt: • 1 (Kurzer Formatstil) • 2 (Mittlerer Formatstil) • 3 (Langer Formatstil) • 4 (Vollständiger Formatstil) Wenn `n` nicht angegeben (oder ungültig) ist, wird der Standardstilwert `0` verwendet.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn `k` nicht angegeben (oder ungültig) ist, wird das Umgebungsgebietsschema verwendet.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `LocalTimeFmt`:

Ausdruck	Rückgabe
`LocalTimeFmt(1, "de_DE")`	`HH:mm`
`LocalTimeFmt(2, "fr_CA")`	`HH:mm:ss`
`LocalTimeFmt(3, "de_CH")`	`HH:mm:ss z`
`LocalTimeFmt(4, "fr_FR")`	`HH' h 'mm z`

Num2Date

Gibt für eine angegebene Anzahl von Tagen seit Beginn der Epoche eine Datums-Zeichenfolge zurück.

Syntax

`Num2Date(n [,f [, k ]])`

Parameter

Parameter	Beschreibung
`n`	Eine Ganzzahl, welche die Anzahl von Tagen angibt. Wenn `n` ungültig ist, gibt die Funktion einen Fehler zurück.
`f` (optional)	Eine Datumsformat-Zeichenfolge. Wenn Sie keinen Wert für `f` angeben, verwendet die Funktion das Standard-Datumsformat `MMM D, YYYY`.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn Sie keinen Wert für `k` angeben oder wenn `k` ungültig ist, verwendet die Funktion das Umgebungsgebietsschema.

Die Funktion gibt den Wert `0` zurück, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Das Format des angegebenen Datums entspricht nicht dem in der Funktion angegebenen Format.
- Das in der Funktion angegebene Gebietsschema oder Datumsformat ist ungültig.

Es sind nicht genügend Informationen vorhanden, um einen eindeutigen Tag seit Beginn der Epoche zu bestimmen (das heißt, die Datumsangaben fehlen oder sind unvollständig).

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Num2Date`:

Ausdruck	Rückgabe
`Num2Date(1, "DD/MM/YYYY")`	`01/01/1900`
`Num2Date(35139, "DD-MMM-YYYY", "de_DE")`	`16-Mrz-1996`
`Num2Date(Date2Num("Mar 15, 2000") - Date2Num("98-03-15", "YY-MM-DD", "fr_CA"))`	`Jan 1, 1902`

Num2GMTime

Gibt für eine angegebene Anzahl von Millisekunden seit Beginn der Epoche eine GMT-Uhrzeit-Zeichenfolge zurück.

Syntax

`Num2GMTime(n [,f [, k ]])`

Parameter

Parameter	Beschreibung
`n`	Eine Ganzzahl, die eine Anzahl von Millisekunden angibt. Wenn `n` ungültig ist, gibt die Funktion einen Fehler zurück.
`f` (optional)	Eine Uhrzeitformat-Zeichenfolge. Wenn Sie keinen Wert für `f` angeben, verwendet die Funktion das Standard-Uhrzeitformat `H:MM:SS A`.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn Sie keinen Wert für `k` angeben oder wenn `k` ungültig ist, verwendet die Funktion das Umgebungsgebietsschema.

Die Funktion gibt den Wert `0` zurück, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Das Format der angegebenen Uhrzeit entspricht nicht dem in der Funktion angegebenen Format.
- Das in der Funktion angegebene Gebietsschema oder Uhrzeitformat ist ungültig.

Es sind nicht genügend Informationen vorhanden, um eine eindeutige Uhrzeit seit Beginn der Epoche zu bestimmen (das heißt, die Zeitangaben fehlen oder sind unvollständig).

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Num2GMTime`:

Ausdruck	Rückgabe
`Num2GMTime(1, "HH:MM:SS")`	`00:00:00`
`Num2GMTime(65593001, "HH:MM:SS Z")`	`18:13:13 GMT`
`Num2GMTime(43993001, TimeFmt(4, "de_DE"), "de_DE")`	`12.13 Uhr GMT`

Num2Time

Gibt für eine angegebene Anzahl von Millisekunden seit Beginn der Epoche eine Uhrzeit-Zeichenfolge zurück.

Syntax

`Num2Time(n [,f [, k ]])`

Parameter

Parameter	Beschreibung
`n`	Eine Ganzzahl, die eine Anzahl von Millisekunden angibt. Wenn `n` ungültig ist, gibt die Funktion einen Fehler zurück.
`f` (optional)	Eine Uhrzeitformat-Zeichenfolge. Wenn Sie keinen Wert für `f` angeben, verwendet die Funktion das Standard-Uhrzeitformat `H:MM:SS A`.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn Sie keinen Wert für `k` angeben oder wenn `k` ungültig ist, verwendet die Funktion das Umgebungsgebietsschema.

Die Funktion gibt den Wert `0` zurück, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Das Format der angegebenen Uhrzeit entspricht nicht dem in der Funktion angegebenen Format.
- Das in der Funktion angegebene Gebietsschema oder Uhrzeitformat ist ungültig.

Es sind nicht genügend Informationen vorhanden, um eine eindeutige Uhrzeit seit Beginn der Epoche zu bestimmen (das heißt, die Zeitangaben fehlen oder sind unvollständig).

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Num2Time`:

Ausdruck	Rückgabe
`Num2Time(1, "HH:MM:SS")`	`00:00:00` in Greenwich und `09:00:00` in Tokio.
`Num2Time(65593001, "HH:MM:SS Z")`	`13:13:13` EST in Boston.
`Num2Time(65593001, "HH:MM:SS Z", "de_DE")`	`13:13:13 GMT-05:00` für einen deutschschweizerischen Benutzer in Boston.
`Num2Time(43993001, TimeFmt(4, "de_DE"), "de_DE")`	`13.13 Uhr GMT+01:00` für einen Benutzer in Zürich.
`Num2Time(43993001, "HH:MM:SSzz")`	`13:13+01:00` für einen Benutzer in Zürich.

Time

Gibt die aktuelle Systemuhrzeit als Anzahl von Millisekunden seit Beginn der Epoche zurück.

Syntax

```
Time()
```

Parameter

Keine

Beispiele

Die folgende Tabelle zeigt ein Beispiel für die Verwendung der Funktion `Time`:

Ausdruck	Rückgabe
`Time()`	`71533235` um genau 3:52:15 PM am 15. September 2003 für einen Benutzer in der Zeitzone Eastern Time (ET).

Time2Num

Gibt für eine angegebene Uhrzeit-Zeichenfolge die Anzahl der Millisekunden seit Beginn der Epoche zurück.

Syntax

```
Time2Num(d [, f [, k ]])
```

Parameter

Parameter	Beschreibung
`d`	Eine Datums-Zeichenfolge in dem von `f` gelieferten Format, die auch dem von `k` angegebenen Gebietsschema entspricht.
`f` (optional)	Eine Uhrzeitformat-Zeichenfolge. Wenn Sie keinen Wert für `f` angeben, verwendet die Funktion das Standard-Uhrzeitformat `H:MM:SS A`.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn Sie keinen Wert für `k` angeben oder wenn `k` ungültig ist, verwendet die Funktion das Umgebungsgebietsschema.

Die Funktion gibt den Wert `0` zurück, wenn mindestens eine der folgenden Bedingungen erfüllt ist:

- Das Format der angegebenen Uhrzeit entspricht nicht dem in der Funktion angegebenen Format.
- Das in der Funktion angegebene Gebietsschema oder Uhrzeitformat ist ungültig.

Es sind nicht genügend Informationen vorhanden, um eine eindeutige Uhrzeit seit Beginn der Epoche zu bestimmen (das heißt, die Zeitangaben fehlen oder sind unvollständig).

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Time2Num`:

Ausdruck	Rückgabe
`Time2Num("00:00:00 GMT", "HH:MM:SS Z")`	`1`
`Time2Num("1:13:13 PM")`	`76393001` für einen Benutzer in Kalifornien während der Winterzeit (Pacific Standard Time) und `76033001`, wenn für denselben Benutzer die Sommerzeit (Pacific Daylight Savings Time) gilt.
`Time2Num("13:13:13", "HH:MM:SS") - Time2Num("13:13:13 GMT", "HH:MM:SS Z")) / (60 * 60 * 1000)`	`8` für einen Benutzer in Vancouver und `5` für einen Benutzer in Ottawa während der Winterzeit. Während der Sommerzeit werden die Werte `7` bzw. `4` zurückgegeben.
`Time2Num("13:13:13 GMT", "HH:MM:SS Z", "fr_FR")`	`47593001`

TimeFmt

Gibt ein Uhrzeitformat in einem angegebenen Uhrzeitformat-Stil zurück.

Syntax

`TimeFmt([`*n* `[, `*k* `]])`

Parameter

Parameter	Beschreibung
`n` (optional)	Eine Ganzzahl, welche den durch das Gebietsschema festgelegten Uhrzeitformat-Stil angibt wie folgt: • 1 (Kurzer Formatstil) • 2 (Mittlerer Formatstil) • 3 (Langer Formatstil) • 4 (Vollständiger Formatstil) Wenn Sie keinen Wert für `n` angeben oder wenn `n` ungültig ist, verwendet die Funktion den Standardstilwert.
`k` (optional)	Eine Zeichenfolge für die Gebietsschema-Kennung, welche den Benennungsstandards für Gebietsschemata entspricht. Wenn `k` nicht angegeben (oder ungültig) ist, wird das Umgebungsgebietsschema verwendet.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `TimeFmt`:

Ausdruck	Rückgabe
`TimeFmt(1)`	`h:MM A` (if en_US locale is set)
`TimeFmt(2, "fr_CA")`	`HH:MM:SS`
`TimeFmt(3, "fr_FR")`	`HH:MM:SS Z`
`TimeFmt(4, "de_DE")`	`H.MM' Uhr 'Z`

6 Finanzfunktionen

Diese Funktionen führen eine Vielzahl von Zins-, Kapital- und Auswertungsberechnungen für den Finanzsektor durch.

Funktionen

Apr

Gibt die jährliche Gesamtbelastung für einen Kredit zurück.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`Apr(n1, n2, n3)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Kapitalbetrag des Kredits angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Betrag pro Zahlung für den Kredit angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Anzahl der Zahlungsperioden während der Laufzeit des Kredits angibt.

Wenn ein Parameter null ist, gibt die Funktion `null` zurück. Wenn ein Parameter negativ oder 0 ist, gibt die Funktion einen Fehler zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Apr`:

Ausdruck	Rückgabe
`Apr(35000, 269.50, 360)`	`0.08515404566` für einen Kredit in Höhe von 35.000 $, der 30 Jahre lang in Monatsraten von 269,50 $ zurückgezahlt wird.
`Apr(210000 * 0.75, 850 + 110, 25 * 26)`	`0.07161332404`
`Apr(-20000, 250, 120)`	Fehler
`Apr(P_Value, Payment, Time)`	In diesem Beispiel werden Variablen statt konkreter Zahlenwerte oder Ausdrücke verwendet.

CTerm

Gibt die Anzahl der Perioden zurück, die erforderlich sind, damit eine Anlage mit einer festen Verzinsung mit Zinseszins auf einen Endwert anwächst.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`CTerm(`*`n1`*`, `*`n2`*`, `*`n3`*`)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Zinssatz pro Periode angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Endwert der Anlage angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Höhe des Anfangskapitals angibt.

Wenn ein Parameter null ist, gibt die Funktion null zurück. Wenn ein Parameter negativ oder 0 ist, gibt die Funktion einen Fehler zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `CTerm`:

Ausdruck	Rückgabe
`CTerm(0.02, 1000, 100)`	`116.2767474515`
`CTerm(0.10, 500000, 12000)`	`39.13224648502`
`CTerm(0.0275 + 0.0025, 1000000, 55000 * 0.10)`	`176.02226044975`
`CTerm(Int_Rate, Target_Amount, P_Value)`	In diesem Beispiel werden Variablen statt konkreter Zahlenwerte oder Ausdrücke verwendet.

FV

Gibt den Endwert von Anlagebeträgen zurück, die regelmäßig und in gleich bleibender Höhe bei einem konstanten Zinssatz eingezahlt werden.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`FV(n1, n2, n3)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Betrag pro Einzahlung angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher die Zinsen pro Anlageperiode angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Gesamtzahl der Einzahlungsperioden angibt.

Die Funktion gibt einen Fehler zurück, wenn eine der folgenden Bedingungen erfüllt ist:

- `n1` oder `n3` ist negativ oder 0.
- `n2` ist negativ.

Wenn einer der Parameter null ist, gibt die Funktion null zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Funktion `FV`:

Ausdruck	Rückgabe
`FV(400, 0.10 / 12, 30 * 12)`	`904195.16991842445`. Dies ist der Wert einer Anlage mit monatlichen Einzahlungen von 400 $ und einer jährlichen Verzinsung von 10 % nach 30 Jahren.
`FV(1000, 0.075 / 4, 10 * 4)`	`58791.96145535981`. Dies ist der Wert einer Anlage mit monatlichen Einzahlungen von 1.000 $ und einer vierteljährlichen Verzinsung von 7,5 % nach 10 Jahren.
`FV(Payment[0], Int_Rate / 4, Time)`	In diesem Beispiel werden Variablen statt konkreter Zahlenwerte oder Ausdrücke verwendet.

IPmt

Gibt den Zinsbetrag zurück, der in einer bestimmten Zeitspanne für einen Kredit gezahlt wurde.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`IPmt(n1, n2, n3, n4, n5)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Kapitalbetrag des Kredits angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Jahreszinssatz der Anlage angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Höhe der monatlichen Zahlungen angibt.
`n4`	Ein Zahlenwert oder Ausdruck, welcher den ersten Monat angibt, in dem eine Zahlung erfolgt.
`n5`	Ein Zahlenwert oder Ausdruck, der angibt, für wie viele Monate die Berechnung durchgeführt werden soll.

Die Funktion gibt einen Fehler zurück, wenn eine der folgenden Bedingungen erfüllt ist:

- `n1`, `n2` oder `n3` ist negativ oder 0.
- `n4` oder `n5` ist negativ.

Wenn ein Parameter null ist, gibt die Funktion null zurück. Wenn der Zahlungsbetrag (`n3`) kleiner ist als die monatliche Zinsbelastung, gibt die Funktion `0` zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `IPmt`:

Ausdruck	Rückgabe
`IPmt(30000, 0.085, 295.50, 7, 3)`	`624.8839283142`. Dies ist der Zinsbetrag, der für einen Kredit in Höhe von 30.000 $ bei einem Zinssatz von 8,5 % in den drei Monaten zwischen dem siebten und zehnten Monat der Kreditlaufzeit gezahlt wurde.
`IPmt(160000, 0.0475, 980, 24, 12)`	`7103.80833569485`. Die Höhe der im dritten Jahr der Kreditlaufzeit gezahlten Zinsen.
`IPmt(15000, 0.065, 65.50, 15, 1)`	`0`, weil die Höhe der monatlichen Zahlungen geringer ist als die für den Kredit während des Monats anfallenden Zinsen.

NPV

Gibt den Kapitalwert einer Anlage auf der Grundlage eines Diskontsatzes und einer Folge von zukünftigen periodischen Cashflows zurück.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`NPV(n1, n2 [, ...])`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Diskontsatz während einer einzelnen Periode angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Wert eines Cashflows angibt, der am Ende einer Periode erfolgen muss. Es ist wichtig, dass die in `n2` und dahinter angegebenen Werte in der richtigen Reihenfolge aufgeführt sind.

Die Funktion gibt einen Fehler zurück, wenn `n1` negativ oder 0 ist. Wenn einer der Parameter null ist, gibt die Funktion null zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `NPV`:

Ausdruck	Rückgabe
`NPV(0.065, 5000)`	`4694.83568075117`; dies ist der Netto-Barwert einer Anlage mit einer jährlichen Verzinsung von 6,5 %, der Einkünfte von 5.000 $ zufließen werden.
`NPV(0.10, 500, 1500, 4000, 10000)`	`11529.60863329007`; dies ist der Kapitalwert einer Anlage mit einer jährlichen Verzinsung von 10 %, der in den nächsten vier Jahren Einkünfte von jeweils 500 $, 1.500 $, 4.000 $ und 10.000 $ zufließen werden.
`NPV(0.0275 / 12, 50, 60, 40, 100, 25)`	`273.14193838457`; dies ist der Kapitalwert einer Anlage mit einer jährlichen Verzinsung von 2,75 %, der in den nächsten fünf Monaten Einkünfte von jeweils 50 $, 60 $, 40 $, 100 $ und 25 $ zufließen werden.

Pmt

Gibt die Höhe des Rückzahlungsbetrags für einen Kredit bei konstanten Zahlungsbeträgen und konstantem Zinssatz zurück.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`Pmt(n1, n2, n3)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Kapitalbetrag des Kredits angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Zinssatz pro Verzinsungsperiode der Anlage angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Gesamtzahl der Einzahlungsperioden angibt.

Die Funktion gibt einen Fehler zurück, wenn einer der Parameter negativ oder 0 ist. Wenn einer der Parameter null ist, gibt die Funktion null zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Pmt`:

Ausdruck	Rückgabe
`Pmt(150000, 0.0475 / 12, 25 * 12)`	`855.17604207164`; dies ist der monatliche Zahlungsbetrag bei einer Kreditsumme von 150.000 $, die bei einer jährlichen Verzinsung von 4,75 % 25 Jahre lang zurückzuzahlen ist.
`Pmt(25000, 0.085, 12)`	`3403.82145169876`; dies ist der jährliche Zahlungsbetrag bei einer Kreditsumme von 25.000 $, die bei einer jährlichen Verzinsung von 8,5 % 12 Jahre lang zurückzuzahlen ist.

PPmt

Gibt den Tilgungsbetrag zurück, der in einer bestimmten Zeitspanne für einen Kredit gezahlt wurde.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`PPmt(n1, n2, n3, n4, n5)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Kapitalbetrag des Kredits angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den jährlichen Zinssatz angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Höhe der monatlichen Zahlungen angibt.
`n4`	Ein Zahlenwert oder Ausdruck, welcher den ersten Monat angibt, in dem eine Zahlung erfolgt.
`n5`	Ein Zahlenwert oder Ausdruck, der angibt, für wie viele Monate die Berechnung durchgeführt werden soll.

Die Funktion gibt einen Fehler zurück, wenn eine der folgenden Bedingungen erfüllt ist:

- `n1`, `n2` oder `n3` ist negativ oder 0.
- `n4` oder `n5` ist negativ.

Wenn ein Parameter null ist, gibt die Funktion null zurück. Wenn der Zahlungsbetrag (`n3`) kleiner ist als die monatliche Zinsbelastung, gibt die Funktion `0` zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `PPmt`:

Ausdruck	Rückgabe
`PPmt(30000, 0.085, 295.50, 7, 3)`	`261.6160716858`, der Tilgungsbetrag, der für einen Kredit in Höhe von 30.000 $ bei einem Zinssatz von 8,5 % in den drei Monaten zwischen dem siebten und zehnten Monat der Kreditlaufzeit gezahlt wurde.
`PPmt(160000, 0.0475, 980, 24, 12)`	`4656.19166430515`, die Höhe des im dritten Jahr der Kreditlaufzeit getilgten Kreditbetrags.
`PPmt(15000, 0.065, 65.50, 15, 1)`	`0`, weil die monatliche Zahlung kleiner ist als die während des Monats anfallenden Zinsen, daher wird nichts vom Kapitalbetrag getilgt.

PV

Gibt den Gegenwartswert einer Anlage mit regelmäßigen konstanten Einzahlungen und konstantem Zinssatz zurück.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`PV(n1, n2, n3)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Betrag pro Einzahlung angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher die Zinsen pro Anlageperiode angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Gesamtzahl der Einzahlungsperioden angibt.

Die Funktion gibt einen Fehler zurück, wenn `n1` oder `n3` negativ oder 0 ist. Wenn einer der Parameter null ist, gibt die Funktion null zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `PV`:

Ausdruck	Rückgabe
`PV(400, 0.10 / 12, 30 * 12)`	`45580.32799074439`. Dies ist der Wert einer Anlage mit monatlichen Einzahlungen von 400 $ und einer jährlichen Verzinsung von 10 % nach 30 Jahren.
`PV(1000, 0.075 / 4, 10 * 4)`	`58791.96145535981`. Dies ist der Wert einer Anlage mit monatlichen Einzahlungen von 1.000 $ und einer vierteljährlichen Verzinsung von 7,5 % nach zehn Jahren.
`PV(Payment[0], Int_Rate / 4, Time)`	In diesem Beispiel werden Variablen statt konkreter Zahlenwerte oder Ausdrücke verwendet.

Rate

Gibt den Zinssatz pro Verzinsungsperiode zurück, der benötigt wird, damit eine Anlage mit Zinseszins in einem gegebenen Zeitraum von einem gegebenen Gegenwartswert auf einen Endwert anwächst.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`Rate(n1, n2, n3)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher den Endwert der Anlage angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Gegenwartswert der Anlage angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher die Gesamtzahl der Anlageperioden angibt.

Die Funktion gibt einen Fehler zurück, wenn einer der Parameter negativ oder 0 ist. Wenn einer der Parameter null ist, gibt die Funktion null zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Rate`:

Ausdruck	Rückgabe
`Rate(12000, 8000, 5)`	`0.0844717712` (oder 8,45 %); dies ist der Zinssatz, der pro Periode benötigt wird, damit ein Gegenwartswert von 8.000 $ in fünf Perioden auf 12.000 $ anwächst.
`Rate(10000, 0.25 * 5000, 4 * 12)`	`0.04427378243` (oder 4,43 %); dies ist der Zinssatz, der pro Monat benötigt wird, damit der Gegenwartswert in vier Jahren auf 10.000 $ anwächst.
`Rate(Target_Value, Pres_Value[*], Term * 12)`	In diesem Beispiel werden Variablen statt konkreter Zahlenwerte oder Ausdrücke verwendet.

Term

Gibt die Anzahl der Perioden zurück, die erforderlich sind, um mit konstanten periodischen Einzahlungen auf ein verzinstes Konto einen angegebenen Endwert zu erzielen.

Hinweis: Die Methoden für die Zinsberechnung sind von Land zu Land unterschiedlich. Diese Funktion berechnet einen Zinssatz anhand der in den USA üblichen Standards.

Syntax

`Term(n1, n2, n3)`

Parameter

Parameter	Beschreibung
`n1`	Ein Zahlenwert oder Ausdruck, welcher die Höhe der Zahlung am Ende jeder Periode angibt.
`n2`	Ein Zahlenwert oder Ausdruck, welcher den Zinssatz pro Verzinsungsperiode der Anlage angibt.
`n3`	Ein Zahlenwert oder Ausdruck, welcher den Endwert der Anlage angibt.

Die Funktion gibt einen Fehler zurück, wenn einer der Parameter negativ oder 0 ist. Wenn einer der Parameter null ist, gibt die Funktion null zurück.

Hinweis: FormCalc folgt bei der Handhabung von numerischen Fließkommawerten dem internationalen Standard IEEE-754. Weitere Informationen finden Sie unter „Zahlenliterale“ auf Seite 8.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Term`:

Ausdruck	Rückgabe
`Term(475, .05, 1500)`	`3.00477517728` (oder ungefähr 3); dies ist die Anzahl der Perioden, die benötigt werden, um Einzahlungen in Höhe von 475 $ bei einem Zinssatz von 5 % pro Periode auf 1.500 $ anwachsen zu lassen.
`Term(2500, 0.0275 + 0.0025, 5000)`	`1.97128786369`; dies ist die Anzahl der Perioden, die benötigt werden, um Einzahlungen in Höhe von 2.500 $ bei einem Zinssatz von 3 % pro Periode auf 5.000 $ anwachsen zu lassen.
`Rate(Inv_Value[0], Int_Rate + 0.0050, Target_Value)`	In diesem Beispiel werden Variablen statt konkreter Zahlenwerte oder Ausdrücke verwendet. In diesem Fall wird das erste Vorkommen der Variablen `Inv_Value` als Einzahlungsbetrag verwendet. Die Variable `Int_Rate` wird um einen halben Prozentpunkt erhöht und als Zinssatz eingesetzt und die Variable `Target_Value` ist der Endwert der Anlage.

7 Logikfunktionen

Diese Funktionen eignen sich zum Testen und/oder Analysieren von Informationen, um ein wahres (TRUE) oder falsches (FALSE) Ergebnis zu erhalten.

Funktionen

Choose

Wählt einen Wert aus einem gegebenen Satz von Parametern.

Syntax

`Choose(n, s1 [, s2 ...])`

Parameter

Parameter	Beschreibung
`n`	Die Position des auszuwählenden Wertes innerhalb des Satzes. Wenn dieser Wert keine Ganzzahl ist, rundet die Funktion `n` auf den nächsten ganzzahligen Wert ab. Die Funktion gibt eine leere Zeichenfolge zurück, wenn eine der folgenden Bedingungen erfüllt ist: • `n` ist kleiner als 1. • `n` ist größer als die Zahl der Elemente im Satz. Wenn `n` null ist, gibt die Funktion null zurück.
`s1`	Der erste Wert in dem Wertesatz.
`s2` (optional)	Weitere Werte im Satz.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Choose`:

Ausdruck	Rückgabe
`Choose(3, "Taxes", "Price", "Person", "Teller")`	`Person`
`Choose(2, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)`	`9`
`Choose(Item_Num[0], Items[*])`	Gibt den Wert innerhalb des Satzes `Items` zurück, welcher sich an der durch das erste Vorkommen von `Item_Num` bestimmten Position befindet.
`Choose(20/3, "A", "B", "C", "D", "E", "F", "G", "H")`	`F`

Exists

Bestimmt, ob der angegebene Parameter eine Referenz-Syntax zu einem vorhandenen Objekt ist.

Syntax

`Exists(v)`

Parameter

Parameter	Beschreibung
`v`	Ein gültiger Referenz-Syntaxausdruck. Wenn `v` keine Referenz-Syntax ist, gibt die Funktion FALSE (`0`) zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Exists`:

Ausdruck	Rückgabe
`Exists(Item)`	TRUE (`1`), wenn das Objekt `Item` existiert, andernfalls FALSE (`0`).
`Exists("hello world")`	FALSE (`0`). Die Zeichenfolge ist keine Referenz-Syntax.
`Exists(Invoice.Border.Edge[1].Color)`	TRUE (`1`), wenn das Objekt `Invoice` existiert und die Eigenschaft `Border` aufweist, die wiederum mindestens eine Eigenschaft `Edge` hat, die wiederum die Eigenschaft `Color` besitzt. Andernfalls gibt die Funktion FALSE (`0`) zurück.

HasValue

Ermittelt, ob der angegebene Parameter eine Referenz-Syntax mit einem von null verschiedenen, nicht leeren oder vom Leerzeichen verschiedenen Wert ist.

Syntax

`HasValue(v)`

Parameter

Parameter	Beschreibung
`v`	Ein gültiger Referenz-Syntaxausdruck. Wenn `v` keine Referenz-Syntax ist, gibt die Funktion FALSE (`0`) zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `HasValue`:

Ausdruck	Rückgabe
`HasValue(2)`	TRUE (`1`)
`HasValue(" ")`	FALSE (`0`)
`HasValue(Amount[*])`	Fehler
`HasValue(Amount[0])`	Wertet das erste Vorkommen von `Amount` aus und gibt TRUE (`1`) zurück, wenn dieser Wert weder null, noch leer, noch das Leerzeichen ist.

Oneof

Bestimmt, ob der angegebene Wert zu einem Satz gehört.

Syntax

`Oneof(`*`s1`*`, `*`s2`*` [, `*`s3`*` ...])`

Parameter

Parameter	Beschreibung
`s1`	Die Position des auszuwählenden Wertes innerhalb des Satzes. Wenn dieser Wert keine Ganzzahl ist, rundet die Funktion `s1` auf den nächsten ganzzahligen Wert ab.
`s2`	Der erste Wert in dem Wertesatz.
`s3` (optional)	Weitere Werte im Satz.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Oneof`:

Ausdruck	Rückgabe
`Oneof(3, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)`	TRUE (`1`)
`Oneof("John", "Bill", "Gary", "Joan", "John", "Lisa")`	TRUE (`1`)
`Oneof(3, 1, 25)`	FALSE (`0`)
`Oneof("loan", Fields[*])`	Prüft, ob eines der Vorkommen von `Fields` den Wert `loan` besitzt.

Within

Bestimmt, ob der angegebene Wert innerhalb eines angegebenen Bereichs liegt.

Syntax

`Within(`*`s1`*`, `*`s2`*`, `*`s3`*`)`

Parameter

Parameter	Beschreibung
`s1`	Der zu prüfende Wert. Wenn `s1` eine Zahl ist, wird eine numerische Sortierreihenfolge verwendet. Wenn `s1` keine Zahl ist, wird beim Sortierungsvergleich die für das aktuelle Gebietsschema angegebene Sortierreihenfolge verwendet. Weitere Informationen finden Sie unter „Gebietsschemata“ auf Seite 44. Wenn `s1` null ist, gibt die Funktion null zurück.
`s2`	Die untere Grenze des Prüfbereichs.
`s3`	Die obere Grenze des Prüfbereichs.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Within`:

Ausdruck	Rückgabe
`Within("C", "A", "D")`	TRUE (`1`)
`Within(1.5, 0, 2)`	TRUE (`1`)
`Within(-1, 0, 2)`	FALSE (`0`)
`Within($, 1, 10)`	TRUE (`1`), wenn der aktuelle Wert zwischen 1 und 10 liegt.

8 Sonstige Funktionen

Die Funktionen in diesem Kapitel lassen sich in keine bestimmte Funktionskategorie einordnen, können sich aber in zahlreichen Anwendungsbereichen als nützlich erweisen.

Funktionen

Eval

Gibt den Wert einer angegebenen Formularberechnung zurück.

Syntax

`Eval(s)`

Parameter

Parameter	Beschreibung

`s`	Eine gültige Zeichenfolge, die einen Ausdruck oder eine Liste von Ausdrücken darstellt.
	
	Hinweis: Die Funktion `Eval` kann keine benutzerdefinierten Variablen und Funktionen referenzieren. Beispiel:
	
	`var s = "var t = concat(s, ""hello"")"`
	`eval(s)`
	
	In diesem Fall erkennt die Funktion `Eval` `s` nicht und gibt daher einen Fehler zurück. Nachfolgende Funktionen, welche die Variable `s` referenzieren, schlagen ebenfalls fehl.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Eval`:

Ausdruck	Rückgabe
`eval("10*3+5*4")`	`50`
`eval("hello")`	`error`

Null

Gibt den Null-Wert zurück. („Null-Wert“ ist gleichbedeutend mit „kein Wert“.)

Definition

`Null()`

Parameter

Keine

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Null`:

Ausdruck	Rückgabe
`Null()`	`null`
`Null() + 5`	`5`
`Quantity = Null()`	Weist dem Objekt `Quantity` `null` zu.
`Concat("ABC", Null(), "DEF")`	`ABCDEF` Siehe auch „Concat“ auf Seite 85.

Ref

Gibt eine Referenz auf ein vorhandenes Objekt zurück.

Definition

`Ref(v)`

Parameter

Parameter	Beschreibung
`v`	Eine gültige Zeichenfolge, die eine Referenz-Syntax, Eigenschaft, Methode oder Funktion darstellt. **Hinweis:** Wenn der angegebene Parameter den Null-Wert hat, gibt die Funktion die Null-Referenz zurück. Für alle anderen angegebenen Parameter erzeugt die Funktion einen Ausnahmefehler.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Ref`:

Ausdrücke	Rückgabe
`Ref("10*3+5*4")`	`10*3+5*4`
`Ref("hello")`	`hello`

UnitType

Gibt die Einheit einer Maßangabe zurück. Eine Maßangabe ist eine Zeichenfolge, die aus einer Zahl und einer nachfolgenden Einheitenbezeichnung besteht.

Syntax

`UnitType(`*s*`)`

Parameter

Parameter	Beschreibung
`s`	Eine gültige Zeichenfolge (Maßangabe), die aus einem Zahlenwert und einer gültigen Maßeinheit besteht. Die folgenden Maßeinheiten werden erkannt: • in, Inch • mm, Millimeter • cm, Zentimeter • pt, Pica, Punkt • mp, Millipunkt Wen `s` ungültig ist, gibt die Funktion `in` zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `UnitType`:

Ausdruck	Ergebnisse
`UnitType("36 in")`	`in`
`UnitType("2.54centimeters")`	`cm`
`UnitType("picas")`	`pt`
`UnitType("2.cm")`	`cm`
`UnitType("2.zero cm")`	`in`
`UnitType("kilometers")`	`in`
`UnitType(Size[0])`	Gibt die Maßeinheit im ersten Vorkommen von `Size` zurück.

UnitValue

Gibt den Zahlenwert einer Messung mit ihrer zugeordneten Maßangabe nach einer optionalen Einheitenumwandlung zurück. Eine Maßangabe ist eine Zeichenfolge, die aus einer Zahl und einer nachfolgenden gültigen Maßeinheit besteht.

Syntax

`UnitValue(`*`s1`* `[, `*`s2`* `])`

Parameter

Parameter	Beschreibung
`s1`	Eine gültige Zeichenfolge (Maßangabe), die aus einem Zahlenwert und einer gültigen Maßeinheit besteht. Die folgenden Maßeinheiten werden erkannt: • in, Inch • mm, Millimeter • cm, Zentimeter • pt, Pica, Punkt • mp, Millipunkt
`s2` (optional)	Eine Zeichenfolge, die eine gültige Maßeinheit enthält. Die Funktion wandelt die in `s1` angegebene Maßangabe in diese neue Maßeinheit um. Wenn Sie für `s2` keinen Wert angeben, verwendet die Funktion die in `s1` angegebene Maßeinheit. Wenn `s2` ungültig ist, wandelt die Funktion `s1` in Zoll (Inch) um.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `UnitValue`:

Ausdruck	Rückgabe
`UnitValue("2in")`	`2`
`UnitValue("2in", "cm")`	`5.08`
`UnitValue("6", "pt")`	`432`
`UnitValue("A", "cm")`	`0`
`UnitValue(Size[2], "mp")`	Gibt den Messwert des dritten Vorkommens von `Size` in Millipunkt umgerechnet zurück.
`UnitValue("5.08cm", "kilograms")`	`2`

9 Zeichenfolgen-Funktionen

Die Funktionen in diesem Kapitel dienen zur Bearbeitung, Auswertung und Erstellung von Zeichenfolgenwerten.

Funktionen

At

Findet die Anfangs-Zeichenposition einer Zeichenfolge innerhalb einer anderen Zeichenfolge.

Syntax
`At(s1, s2)`

Parameter

Parameter	Beschreibung
`s1`	Die Quell-Zeichenfolge.
`s2`	Die Such-Zeichenfolge. Wenn `s2` kein Bestandteil von `s1` ist, gibt die Funktion `0` zurück. Wen `s2` leer ist, gibt die Funktion `1` zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `At`:

Ausdruck	Rückgabe
`At("ABCDEFGH", "AB")`	`1`
`At("ABCDEFGH", "F")`	`6`
`At(23412931298471, 29)`	`5`, das erste Vorkommen von `29` in der Quell-Zeichenfolge.
`At(Ltrim(Cust_Info[0]), "555")`	Die Position der Zeichenfolge `555` innerhalb des ersten Vorkommens von `Cust_Info`. Siehe auch „Ltrim“ auf Seite 91.

Concat

Gibt die Verkettung von zwei oder mehr Zeichenfolgen zurück.

Syntax
`Concat(s1 [, s2 ...])`

Parameter

Parameter	Beschreibung
`s1`	Die erste Zeichenfolge in dem Satz.
`s2` (optional)	Weitere Zeichenfolgen, die an den Satz angehängt werden sollen.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Concat`:

Ausdruck	Rückgabe
`Concat("ABC", "DEF")`	`ABCDEF`
`Concat("Tony", Space(1), "Blue")`	`Tony Blue` Siehe auch „Space“ auf Seite 94.
`Concat("You owe ", WordNum(1154.67, 2), ".")`	`You owe One Thousand One Hundred Fifty-four Dollars And Sixty-seven Cents.` Siehe auch „WordNum“ auf Seite 99.

Decode

Gibt die dekodierte Version einer angegebenen Zeichenfolge zurück.

Syntax

`Decode(`*`s1`* `[, `*`s2`* `])`

Parameter

Parameter	Beschreibung
`s1`	Die zu dekodierende Zeichenfolge.
`s2` (optional)	Eine Zeichenfolge, die die Art der durchzuführenden Dekodierung angibt. Die folgenden Zeichenfolgen sind gültige Dekodierungs-Zeichenfolgen: • url (URL-Dekodierung) • html (HTML-Dekodierung) • xml (XML-Dekodierung) Wenn Sie keinen Wert für `s2` angeben, führt die Funktion eine URL-Dekodierung durch.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Decode`:

Ausdruck	Rückgabe
`Decode("&AElig;&Aacute;&Acirc;&Aacute; &Acirc;", „html“)`	`ÆÁÂÁÂ`
`Decode("~!@#$%^&*()_+|`{"}[] <>?,./;':", „xml“)`	`~!@#$%^&*()_+|`{""}[]<>?,./;':`

Encode

Gibt die kodierte Version einer angegebenen Zeichenfolge zurück.

Syntax
`Encode(`*`s1`* `[, `*`s2`* `])`

Parameter

Parameter	Beschreibung
`s1`	Die zu kodierende Zeichenfolge.
`s2` (optional)	Eine Zeichenfolge, welche die Art der durchzuführenden Kodierung angibt. Die folgenden Zeichenfolgen sind gültige Kodierungs-Zeichenfolgen: • url (URL-Kodierung) • html (HTML-Kodierung) • xml (XML-Kodierung) Wenn Sie keinen Wert für `s2` angeben, führt die Funktion eine URL-Kodierung durch.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Encode`:

Ausdruck	Rückgabe
`Encode("""hello, world!""", "url")`	`%22hello,%20world!%22`
`Encode("ÁÂÃÄÅÆ", "html")`	`&#xc1;&#Xc2;&#Xc3;&#xc4;&#xc5;&#xc6;`

Format

Formatiert die angegebenen Daten gemäß der angegebenen Musterformat-Zeichenfolge.

Syntax

`Format(s1, s2 [, s3 ...])`

Parameter

Parameter	Beschreibung
`s1`	Die Musterformat-Zeichenfolge, die ein an das Gebietsschema angepasstes Datums- oder Uhrzeitformat sein kann. Siehe „Gebietsschemata“ auf Seite 44.
`s2`	Die zu formatierenden Quelldaten. Bei Musterformaten für Datumsangaben müssen die Quelldaten entweder eine ISO-konforme Datum-Uhrzeit-Zeichenfolge oder eine ISO-konforme Datums-Zeichenfolge in einem der beiden folgenden Formate sein: • YYYY[MM[DD]] • YYYY[-MM[-DD]] Bei Musterformaten für Uhrzeitangaben müssen die Quelldaten entweder eine ISO-konforme Datum-Uhrzeit-Zeichenfolge oder eine ISO-konforme Uhrzeit-Zeichenfolge in einem der folgenden Formate sein: • HH[MM[SS[.FFF][z]]] • HH[MM[SS[.FFF][+HH[MM]]]] • HH[MM[SS[.FFF][-HH[MM]]]] • HH[:MM[:SS[.FFF][z] • HH[:MM[:SS[.FFF][-HH[:MM]]]] • HH[:MM[:SS[.FFF][+HH[:MM]]]] Bei Datum-Uhrzeit-Musterformaten müssen die Quelldaten eine ISO-konforme Datum-Uhrzeit-Zeichenfolge sein. Bei numerischen Musterformaten müssen die Quelldaten numerisch sein. Bei Text-Musterformaten muss es sich bei den Quelldaten um Text handeln. Bei zusammengesetzten Musterformaten muss die Anzahl der Quelldaten-Argumente mit der Anzahl der Unterelemente im Muster übereinstimmen.
`s3` (optional)	Zusätzliche zu formatierende Quelldaten.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Format`:

Ausdruck	Rückgabe
`Format("MMM D, YYYY", "20020901")`	`Sep 1, 2002`
`Format("$9,999,999.99", 1234567.89)`	`$1,234,567.89` in den USA und `1 234 567,89` Euro in Frankreich.

Left

Extrahiert eine angegebene Anzahl von Zeichen aus einer Zeichenfolge, beginnend beim ersten Zeichen auf der linken Seite.

Syntax

`Left(s, n)`

Parameter

Parameter	Beschreibung
`s`	Die Zeichenfolge, aus der extrahiert werden soll.
`n`	Die Anzahl der zu extrahierenden Zeichen. Wenn die Anzahl der zu extrahierenden Zeichen größer als die Länge der Zeichenfolge ist, gibt die Funktion die komplette Zeichenfolge zurück. Wenn die Anzahl der zu extrahierenden Zeichen 0 oder kleiner ist, gibt die Funktion die leere Zeichenfolge zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Left`:

Ausdruck	Rückgabe
`Left("ABCDEFGH", 3)`	`ABC`
`Left("Tony Blue", 5)`	`"Tony "`
`Left(Telephone[0], 3)`	Die ersten drei Zeichen des ersten Vorkommens von `Telephone`.
`Left(Rtrim(Last_Name), 3)`	Die drei ersten Zeichen von `Last_Name`. Siehe auch „Rtrim“ auf Seite 93.

Len

Gibt die Anzahl der Zeichen in einer angegebenen Zeichenfolge zurück.

Syntax

`Len(s)`

Parameter

Parameter	Beschreibung
`s`	Die zu untersuchende Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Len`:

Ausdruck	Rückgabe
`Len("ABDCEFGH")`	`8`
`Len(4)`	`1`
`Len(Str(4.532, 6, 4))`	`6` Siehe auch „Str“ auf Seite 95.
`Len(Amount[*])`	Die Anzahl der Zeichen im ersten Vorkommen von `Amount`.

Lower

Wandelt alle Großbuchstaben in einer angegebenen Zeichenfolge in Kleinbuchstaben um.

Syntax

`Lower(`*`s`*`, [, `*`k`*` ])`

Parameter

Parameter	Beschreibung
`s`	Die umzuwandelnde Zeichenfolge.
`k` (optional)	Eine Zeichenfolge, die ein gültiges Gebietsschema angibt. Wenn Sie keinen Wert für `k` angeben, verwendet die Funktion das Umgebungsgebietsschema. Siehe auch „Gebietsschemata“ auf Seite 44. **Hinweis:** Diese Funktion wandelt nur die Unicode-Zeichen von U+41 bis U+5A (im ASCII-Zeichensatz) sowie die Zeichen von U+FF21 bis U+FF3A (im vollbreiten Zeichensatz) um.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Lower`:

Ausdruck	Rückgabe
`Lower("ABC")`	`abc`
`Lower("21 Main St.")`	`21 main st.`
`Lower(15)`	`15`
`Lower(Address[0])`	In diesem Beispiel wird das erste Vorkommen von `Address` vollständig in Kleinbuchstaben umgewandelt.

Ltrim

Gibt eine Zeichenfolge zurück, bei der alle Leerraum-Zeichen am Anfang entfernt wurden.

Zu den Leerraum-Zeichen gehören ASCII-Leerzeichen, horizontaler Tabulator, Zeilenvorschub, vertikaler Tabulator, Formularvorschub, Wagenrücklauf sowie die Unicode-Leerzeichen (Unicode-Kategorie Zs).

Syntax

`Ltrim(s)`

Parameter

Parameter	Beschreibung
`s`	Die zu verkürzende Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Ltrim`:

Ausdruck	Rückgabe
`Ltrim("  ABCD")`	`"ABCD"`
`Ltrim(Rtrim("  Tony Blue  "))`	`"Tony Blue"` Siehe auch „Rtrim“ auf Seite 93.
`Ltrim(Address[0])`	Entfernt ggf. den Leerraum am Anfang des ersten Vorkommens von `Address`.

Parse

Analysiert die angegebenen Daten gemäß dem angegebenen Musterformat.

Wenn die Daten erfolgreich analysiert wurden, liefert die Funktion einen der folgenden Werte:

- Datums-Musterformat: Eine ISO-konforme Datums-Zeichenfolge der Form YYYY-MM-DD.
- Uhrzeit-Musterformat: Eine ISO-konforme Uhrzeit-Zeichenfolge der Form HH:MM:SS.
- Datum-Uhrzeit-Musterformat: Eine ISO-konforme Datum-Uhrzeit-Zeichenfolge der Form YYYY-MM-DDTHH:MM:SS.
- Numerisches Musterformat: Eine Zahl.
- Text-Muster: Text.

Syntax

`Parse(s1, s2 )`

Parameter

Parameter	Beschreibung
`s1`	Eine gültige Datum- oder Uhrzeit-Musterformat-Zeichenfolge. Weitere Informationen zu Datums- und Uhrzeitformaten finden Sie unter „Datum und Uhrzeit strukturieren“ auf Seite 44.
`s2`	Die zu analysierenden Zeichenfolgen-Daten.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Parse`:

Ausdruck	Rückgabe
`Parse("MMM D, YYYY", "Sep 1, 2002")`	`2002-09-01`
`Parse("$9,999,999.99", "$1,234,567.89")`	`1234567.89` in den USA

Replace

Ersetzt innerhalb einer angegebenen Zeichenfolge alle Fundstellen einer Zeichenfolge durch eine andere.

Syntax

`Replace(`*`s1`*`, `*`s2`*` [, `*`s3`*` ])`

Parameter

Parameter	Beschreibung
`s1`	Eine Quell-Zeichenfolge.
`s2`	Die zu ersetzende Zeichenfolge.
`s3` (optional)	Die Ersatz-Zeichenfolge. Wenn Sie keinen Wert für `s3` angeben oder wenn `s3` null ist, verwendet die Funktion eine leere Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Replace`:

Ausdruck	Rückgabe
`Replace("Tony Blue", "Tony", "Chris")`	`Chris Blue`
`Replace("ABCDEFGH", "D")`	`ABCEFGH`
`Replace("ABCDEFGH", "d")`	`ABCDEFGH`
`Replace(Comments[0], "recieve", "receive")`	Berichtigt die Schreibung des Wortes `receive` im ersten Vorkommen von `Comments`.

Right

Extrahiert mehrere Zeichen aus einer angegebenen Zeichenfolge, beginnend beim letzten Zeichen auf der rechten Seite.

Syntax

`Right(`*`s`*`, `*`n`*` )`

Parameter

Parameter	Beschreibung
`s`	Die zu extrahierende Zeichenfolge.
`n`	Die Anzahl der zu extrahierenden Zeichen. Wenn `n` größer als die Länge der Zeichenfolge ist, gibt die Funktion die gesamte Zeichenfolge zurück. Wenn `n` gleich oder kleiner 0 ist, gibt die Funktion eine leere Zeichenfolge zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Right`:

Ausdruck	Rückgabe
`Right("ABCDEFGH", 3)`	`FGH`
`Right("Tony Blue", 5)`	`" Blue"`
`Right(Telephone[0], 7)`	Die letzten sieben Zeichen des ersten Vorkommens von `Telephone`.
`Right(Rtrim(CreditCard_Num), 4)`	Die letzten vier Zeichen von `CreditCard_Num`. Siehe auch „Rtrim“ auf Seite 93.

Rtrim

Gibt eine Zeichenfolge zurück, bei der alle Leerraum-Zeichen am Ende entfernt wurden.

Zu den Leerraum-Zeichen gehören ASCII-Leerzeichen, horizontaler Tabulator, Zeilenvorschub, vertikaler Tabulator, Formularvorschub, Wagenrücklauf sowie die Unicode-Leerzeichen (Unicode-Kategorie Zs).

Syntax

`Rtrim(`*`s`*` )`

Parameter

Parameter	Beschreibung
`s`	Die zu verkürzende Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Rtrim`:

Ausdruck	Rückgabe
`Rtrim("ABCD ")`	`"ABCD"`
`Rtrim("Tony Blue ")`	`"Tony Blue"`
`Rtrim(Address[0])`	Entfernt ggf. den Leerraum am Ende des ersten Vorkommens von `Address`.

Space

Gibt eine Zeichenfolge zurück, die aus einer angegebenen Anzahl von Leerzeichen besteht.

Syntax

`Space(`*`n`*` )`

Parameter

Parameter	Beschreibung
`n`	Die Anzahl der Leerzeichen.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Space`:

Ausdruck	Rückgabe
`Space(5)`	`"  "`
`Space(Max(Amount[*]))`	Eine Zeichenfolge, die so viele Leerzeichen enthält, wie es dem größten Vorkommen von `Amount` entspricht. Siehe auch „Max“ auf Seite 39.
`Concat("Tony", Space(1), "Blue")`	`Tony Blue`

Str

Wandelt eine Zahl in eine Zeichenfolge um. FormCalc formatiert das Ergebnis auf die angegebene Breite und rundet auf die angegebene Zahl von Dezimalstellen.

Syntax

`Str(n1 [, n2 [, n3 ]])`

Parameter

Parameter	Beschreibung
`n1`	Die umzuwandelnde Zahl.
`n2` (optional)	Die maximale Breite der Zeichenfolge. Wenn Sie keinen Wert für `n2` angeben, verwendet die Funktion den Wert `10` als Standardbreite. Wenn die resultierende Zeichenfolge länger als `n2` ist, gibt die Funktion eine aus dem Zeichen * (Stern) bestehende Zeichenfolge mit der in `n2` angegebenen Breite zurück.
`n3` (optional)	Die Anzahl der Ziffern, die hinter dem Dezimalpunkt angezeigt werden sollen. Wenn Sie keinen Wert für `n3` angeben, verwendet die Funktion die Standard-genauigkeit 0.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Str`:

Ausdruck	Rückgabe
`Str(2.456)`	`"          2"`
`Str(4.532, 6, 4)`	`4.5320`
`Str(234.458, 4)`	`" 234"`
`Str(31.2345, 4, 2)`	`****`
`Str(Max(Amount[*]), 6, 2)`	Wandelt das größte Vorkommen von `Amount` in eine sechs Zeichen lange Zeichenfolge mit zwei Dezimalstellen um. Siehe auch „Max“ auf Seite 39.

Stuff

Fügt eine Zeichenfolge in eine andere Zeichenfolge ein.

Syntax

`Stuff(s1, n1, n2 [, s2 ])`

Parameter

Parameter	Beschreibung
`s1`	Die Quell-Zeichenfolge.
`n1`	Die Position in `s1`, an welcher die neue Zeichenfolge `s2` eingefügt werden soll. Wenn `n1` kleiner als 1 ist, geht die Funktion von der ersten Zeichenposition aus. Wenn `n1` größer als die Länge von `s1` ist, geht die Funktion von der letzten Zeichenposition aus.
`n2`	Die Anzahl der Zeichen, die aus der Zeichenfolge `s1` gelöscht werden sollen, beginnend an der Zeichenposition `n1`. Wenn `n2` kleiner oder gleich 0 ist, geht die Funktion von 0 Zeichen aus.
`s2` (optional)	Die Zeichenfolge, die in `s1` eingefügt werden soll. Wenn Sie keinen Wert für `s2` angeben, verwendet die Funktion die leere Zeichenfolge.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Stuff`:

Ausdruck	Rückgabe
`Stuff("TonyBlue", 5, 0, " ")`	`Tony Blue`
`Stuff("ABCDEFGH", 4, 2)`	`ABCFGH`
`Stuff(Address[0], Len(Address[0]), 0, "Street")`	Hiermit wird das Wort `Street` am Ende des ersten Vorkommens von `Address` hinzugefügt. Siehe auch „Len“ auf Seite 89.
`Stuff("members-list@myweb.com", 0, 0, "cc:"`	`cc:members-list@myweb.com`

Substr

Extrahiert einen Abschnitt aus einer angegebenen Zeichenfolge.

Syntax

`Substr(`*`s1`*`, `*`n1`*`, `*`n2`*` )`

Parameter

Parameter	Beschreibung
`s1`	Die Quell-Zeichenfolge.
`n1`	Die Position in der Zeichenfolge `s1`, an welcher die Extraktion beginnen soll. Wenn `n1` kleiner als 1 ist, geht die Funktion von der ersten Zeichenposition aus. Wenn `n1` größer als die Länge von `s1` ist, geht die Funktion von der letzten Zeichenposition aus.
`n2`	Die Anzahl der zu extrahierenden Zeichen. Wenn `n2` kleiner oder gleich 0 ist, gibt FormCalc eine leere Zeichenfolge zurück. Wenn `n1 + n2` größer als die Länge von `s1` ist, gibt die Funktion die Unterzeichenfolge von der Position von `n1` bis zum Ende von `s1` zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Substr`:

Ausdruck	Rückgabe
`Substr("ABCDEFG", 3, 4)`	`CDEF`
`Substr(3214, 2, 1)`	`2`
`Substr(Last_Name[0], 1, 3)`	Liefert die drei ersten Zeichen aus dem ersten Vorkommen von `Last_Name`.
`Substr("ABCDEFG", 5, 0)`	`""`
`Substr("21 Waterloo St.", 4, 5)`	`Water`

Upper

Wandelt alle Kleinbuchstaben in einer angegebenen Zeichenfolge in Großbuchstaben um.

Syntax
`Upper(s [, k ])`

Parameter

Parameter	Beschreibung
`s`	Die umzuwandelnde Zeichenfolge.
`k` (optional)	Eine Zeichenfolge, die ein gültiges Gebietsschema angibt. Wenn Sie keinen Wert für `k` angeben, wird das Umgebungsgebietsschema verwendet. Siehe auch „Gebietsschemata“ auf Seite 44. **Hinweis:** Diese Funktion wandelt nur die Unicode-Zeichen von U+61 bis U+7A (im ASCII-Zeichensatz) sowie die Zeichen von U+FF41 bis U+FF5A (im vollbreiten Zeichensatz) um.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Upper`:

Ausdruck	Rückgabe
`Upper("abc")`	`ABC`
`Upper("21 Main St.")`	`21 MAIN ST.`
`Upper(15)`	`15`
`Upper(Address[0])`	In diesem Beispiel wird das erste Vorkommen von `Address` vollständig in Großbuchstaben umgewandelt.

Uuid

Gibt eine UUID-Zeichenfolge (Universally Unique Identifier) zurück, die als Kennzeichnungsmethode verwendet werden soll.

Syntax
`Uuid([n ])`

Parameter

Parameter	Beschreibung
`n`	Eine Zahl, welche das Format der UUID-Zeichenfolge angibt. Gültige Zahlenwerte sind: • 0 (Standardwert): UUID-Zeichenfolge enthält nur hexadezimale Oktett-Ziffern. • 1: UUID-Zeichenfolge enthält an festgelegten Positionen Bindestriche zur Gliederung der hexadezimalen Oktettfolgen. Wenn Sie keinen Wert für `n` angeben, verwendet die Funktion den Standardwert.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Funktion `Uuid`:

Ausdruck	Rückgabe
`Uuid()`	Ein Wert wie z. B. `3c3400001037be8996c400a0c9c86dd5`
`Uuid(0)`	Ein Wert wie z. B. `3c3400001037be8996c400a0c9c86dd5`
`Uuid(1)`	Ein Wert wie z. B. `1a3ac000-3dde-f352-96c4-00a0c9c86dd5`
`Uuid(7)`	Ein Wert wie z. B. `1a3ac000-3dde-f352-96c4-00a0c9c86dd5`

WordNum

Gibt die englischsprachige Textentsprechung einer angegebenen Zahl zurück.

Syntax

`WordNum(`*`n1`* `[, `*`n2`* `[, `*`k`* `]])`

Parameter

Parameter	Beschreibung
`n1`	Die umzuwandelnde Zahl. Wenn eine der folgenden Aussagen wahr ist, gibt die Funktion als Hinweis auf einen Fehler das Zeichen * (Stern) zurück: • `n1` ist keine Zahl. • Der ganzzahlige Wert von `n1` ist negativ. • Der ganzzahlige Wert von `n1` ist größer als 922.337.203.685.477.550.
`n2` (optional)	Eine Zahl zur Angabe der Formatierungsoption. Gültige Zahlenwerte sind: • 0 (Standardwert): Die Zahl wird in einen Text umgewandelt, welcher die einfache Zahl darstellt. • 1: Die Zahl wird in einen Text umgewandelt, welcher den Geldwert ohne Stellen nach dem Komma darstellt. • 2: Die Zahl wird in einen Text umgewandelt, welcher den Geldwert mit Stellen nach dem Komma darstellt. Wenn Sie keinen Wert für `n2` angeben, verwendet die Funktion den Standardwert (`0`).
`k` (optional)	Eine Zeichenfolge, die ein gültiges Gebietsschema angibt. Wenn Sie keinen Wert für `k` angeben, verwendet die Funktion das Umgebungsgebietsschema. Siehe auch „Gebietsschemata“ auf Seite 44. **Hinweis:** In dieser Version ist es nicht möglich, für diese Funktion eine andere Gebietsschema-Kennung als Englisch anzugeben.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `WordNum`:

Ausdruck	Rückgabe
`WordNum(123.45)`	`One Hundred and Twenty-three Dollars`
`WordNum(123.45, 1)`	`One Hundred and Twenty-three Dollars`
`WordNum(1154.67, 2)`	`One Thousand One Hundred Fifty-four Dollars And Sixty-seven Cents`
`WordNum(43, 2)`	`Forty-three Dollars And Zero Cents`
`WordNum(Amount[0], 2)`	Dieses Beispiel verwendet das erste Vorkommen von `Amount` als umzuwandelnde Zahl.

10 URL-Funktionen

Diese Funktionen dienen zum Austausch von Informationen (Senden und Empfangen), einschließlich Inhaltstypen und Kodierungsdaten, mit beliebigen erreichbaren URL-Adressen.

Funktionen

Get

Lädt den Inhalt der angegebenen URL herunter.

Hinweis: Die Funktion `Get` wird nur ausgeführt, wenn ein Formular zertifiziert ist. Adobe Acrobat® und Adobe Reader® können erst nach dem Initiieren des Ereignisses `initialize` prüfen, ob das Formular zertifiziert ist. Soll die Funktion `Get` vor der Formularwiedergabe bei zertifizierten Formularen eingesetzt werden, verwenden Sie das Ereignis `docReady`.

Syntax

`Get(`*s*`)`

Parameter

Parameter	Beschreibung
`s`	Die URL, von der Daten heruntergeladen werden sollen. Wenn die Funktion von der angegebenen URL keine Daten herunterladen kann, gibt sie einen Fehler zurück.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Get`:

Ausdruck	Rückgabe
`Get("http://www.myweb.com/data/mydata.xml")`	XML-Daten, welche der angegebenen Datei entnommen wurden.
`Get("ftp://ftp.gnu.org/gnu/GPL")`	Der Inhalt der GNU Public License.
`Get("http://intranet?sql=SELECT+*+FROM+projects+FOR+XML+AUTO,+ELEMENTS")`	Das Ergebnis einer SQL-Abfrage der angegebenen Website.

Post

Sendet die angegebenen Daten an die genannte URL.

Hinweis: Die Funktion `Post` wird nur ausgeführt, wenn ein Formular zertifiziert ist. Acrobat und Adobe Reader können erst nach dem Initiieren des Ereignisses `initialize` prüfen, ob das Formular zertifiziert ist. Soll die Funktion `Post` vor der Formularwiedergabe bei zertifizierten Formularen eingesetzt werden, verwenden Sie das Ereignis `docReady`.

Syntax

`Post(`*`s1`*`, `*`s2`*` [, `*`s3`*` [, `*`s4`*` [, `*`s5`*` ]]])`

Parameter

Parameter	Beschreibung
`s1`	Die URL, an die Daten gesendet werden sollen.
`s2`	Die zu sendenden Daten. Wenn die Funktion die Daten nicht senden kann, gibt sie einen Fehler zurück.
`s3` (optional)	Eine Zeichenfolge, welche den Inhaltstyp der zu sendenden Daten enthält. Folgende Inhaltstypen sind zulässig: • application/octet-stream (Standardwert) • text/html • text/xml • text/plain • multipart/form-data • application/x-www-form-urlencoded • Alle sonstigen gültigen MIME-Typen Wenn Sie für `s3` keinen Wert angeben, setzt die Funktion den Inhaltstyp auf den Standardwert. Die Anwendung gewährleistet, dass bei den zu sendenden Daten das richtige Format für den angegebenen Inhaltstyp verwendet wird.

Parameter	Beschreibung
`s4` (optional)	Eine Zeichenfolge mit dem Namen der für die Kodierung der Daten verwendeten Codepage. Folgende Codepage-Namen sind gültig: • UTF-8 (Standardwert) • UTF-16 • ISO-8859-1 • Jede von der Internet Assigned Numbers Authority (IANA) aufgeführte Zeichenkodierung Wenn Sie für `s4` keinen Wert angeben, setzt die Funktion die Codepage auf den Standardwert. Die Anwendung gewährleistet, dass die Kodierung der zu sendenden Daten der angegebenen Codepage entspricht.
`s5` (optional)	Eine Zeichenfolge, die zusätzliche, beim Senden der Daten einzuschließende HTTP-Header enthält. Wenn Sie für `s5` keinen Wert angeben, fügt die Funktion beim Senden keinen zusätzlichen HTTP-Header ein. Wenn Daten an SOAP-Server gesendet werden, ist in der Regel eine SOAPAction-Kopfzeile erforderlich.

Beispiele

Die folgende Tabelle zeigt Beispiele für die Verwendung der Funktion `Post`:

Ausdruck	Rückgabe
`Post("http://tools_build/scripts/jfecho.cgi", "user=joe&passwd=xxxxx&date=27/08/2002", "application/x-www-form-urlencoded")`	Sendet URL-kodierte Anmeldedaten an einen Server und gibt die Bestätigungsseite dieses Servers zurück.
`Post("http://www.nanonull.com/TimeService/TimeService.asmx/getLocalTime", "<?xml version='1.0' encoding='UTF-8'?><soap:Envelope><soap:Body><getLocalTime/></soap:Body></soap:Envelope>", "text/xml", "utf-8", "http://www.Nanonull.com/TimeService/getLocalTime")`	Sendet eine SOAP-Anforderung der Ortszeit an einen Server und erwartet die Rückgabe einer XML-Antwort.

Put

Lädt die angegebenen Daten zu der genannten URL hoch.

Hinweis: Die Funktion `Put` wird nur ausgeführt, wenn ein Formular zertifiziert ist. Acrobat und Adobe Reader können erst nach dem Initiieren des Ereignisses `initialize` prüfen, ob das Formular zertifiziert ist. Soll die Funktion `Put` vor der Formularwiedergabe bei zertifizierten Formularen eingesetzt werden, verwenden Sie das Ereignis `docReady`.

Syntax

`Put(`*`s1`*`, `*`s2`*` [, `*`s3`*` ])`

Parameter

Parameter	Beschreibung
`s1`	Die URL, zu der Daten hochgeladen werden sollen.
`s2`	Die hochzuladenden Daten. Wenn die Funktion die Daten nicht hochladen kann, gibt sie einen Fehler zurück.
`s3` (optional)	Eine Zeichenfolge mit dem Namen der für die Kodierung der Daten verwendeten Codepage. Folgende Codepage-Namen sind gültig: • UTF-8 (Standardwert) • UTF-16 • ISO8859-1 • Jede von der Internet Assigned Numbers Authority (IANA) aufgeführte Zeichenkodierung Wenn Sie für `s3` keinen Wert angeben, setzt die Funktion die Codepage auf den Standardwert. Die Anwendung gewährleistet, dass die Kodierung der hochzuladenden Daten der angegebenen Codepage entspricht.

Beispiele

Die folgende Tabelle zeigt ein Beispiel für die Verwendung der Funktion `Put`:

Ausdruck	Rückgabe
`Put("ftp://www.example.com/pub/fubu.xml", "<?xml version='1.0' encoding='UTF-8'?><msg>hello world!</msg>")`	Nichts, wenn der FTP-Server dem Benutzer die Genehmigung erteilt hat, XML-Daten in die Datei `pub/fubu.xml` hochzuladen. Andernfalls gibt diese Funktion einen Fehler zurück.

Index