Formula Functions Quick Reference

Download the PDF of this article.

In this Article

Related Articles

Introduction

This page is meant to serve as a quick reference guide for all of FormAssembly's formula functions.

If you are unfamiliar with FormAssembly formulas, please visit our Smart Processing with Formulas page for an introduction. For practical formula examples, please see our Formulas for Common Uses page.

You can also view an explanation for each formula function by hovering over an individual function in FormAssembly's field editor.

Functions

The engine supports most functions found in MS Excel. The function must be spelled in uppercase and start with the `@` character.

Please note: Formulas cannot be used in connector repeating sections. Javascript calculations cannot be used in connectors. HTML cannot be used in formulas.

Here is a complete list of all FormAssembly functions, broken down by category:

Logic

@COMPUTE	`@COMPUTE(expression)` Performs arithmetic calculations on form fields. Example:

	`@COMPUTE(%%tfa_1%%+%%tfa_2%%)`

@AND	`@AND(condition 1,condition 2)`
	
	Returns TRUE if both conditions are true, FALSE otherwise (logical AND).
	
	Example:
	`@IF(@AND(%%tfa_1%%>5,%%tfa_2%%>10),"OK","NOT OK")`

@FALSE	`@FALSE()`
	
	Returns the logical value FALSE.
	
	You may receive an error in your formula editor that this argument requires two arguments. It works exactly like @TRUE and does not require two arguments. This is being looked at to be fixed.
	
	Example:
	`@IF(@ISNUMBER(%%tfa_3%%)=@FALSE(),”It is not a number”,”It is a number”)`
	
	Looks at the field and posts if it is a number or not a number.

@IF	`@IF(condition,when_true,when_false)`
	
	Performs a logical test and returns either the second parameter (*if true*) or the third parameter (*if false*).
	
	Example:
	`@IF(%%tfa_1%%>5,"GOOD","NOT ENOUGH")`

@IFERROR	`@IFERROR(value,value_if_error)`
	
	Returns a value you specify if a formula evaluates to an error; otherwise, returns the result of the formula. Use the IFERROR function to trap and handle errors in a formula.
	
	Example:
	`@IFERROR(@DATEDIF(%%tfa_1%%,%%tfa_2%%,"D"),@DATEDIF(%%tfa_2%%,%%tfa_2%%,"D"))`
	
	Compare two dates to receive a difference. This formula compares first date to the second date and if there is an error, it swaps them and compares second to first.

@NOT	`@NOT(condition)`

	Returns TRUE if the condition is false, FALSE otherwise (logical NOT). Example: `@IF(@NOT(%%tfa_1%%>5),"OK","NOT OK")`

@OR	`@OR(condition 1,condition 2)`
	
	Returns TRUE if at least one condition is true, FALSE if all conditions are false (logical OR).
	
	Example:
	`@IF(@OR(%%tfa_1%%>5,%%tfa_2%%>10),when_true,when_false)`

@TRUE	`@TRUE()` Returns the logical value TRUE.

Text and Data

@ADDSLASHES	`@ADDSLASHES(%%tfa_XX%%)`
	
	Returns the string with backslashes before characters that need to be escaped. These characters are single quote ('), double quote ("), backslash (\) and NULL characters.
	
	Example:
	`@ADDSLASHES(%%tfa_1%%)`

@CHAR	`@CHAR(number)`
	
	Returns the character specified by a number. Number is a number between 1 and 255 specifying which character you want.
	
	Example:
	`@CHAR(65)`

@CODE	`@CODE(text)` Returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer.

	Example: `@CODE(%%tfa_1%%)`

@CONCATENATE	`@CONCATENATE(string,string,...)`
	
	Joins 2 or more strings together. Can also be used to send strings and field aliases together.
	
	Example:
	`@CONCATENATE(%%tfa_1%%," ",%%tfa_2%%)`

@CONTAINS	`@CONTAINS(text,field)`
	
	Returns true if the text is inside the field. Otherwise, returns false.
	
	Example:
	`@CONTAINS("@gmail.com",%%tfa_1%%)`

@FIND	`@FIND(text1,text2,start_position)`
	
	Returns the location of a substring in a string (case-sensitive). Returns#VALUE! if string not found.
	
	Example (Returns the domain part of an email address):
	`@MID(%%tfa_email%%,@COMPUTE(@FIND("@",%%tfa_email%%)+1),`
	`@COMPUTE(@FIND(".",%%tfa_email%%)-@FIND("@",%%tfa_email%%)-1))`

@LEFT	`@LEFT(text,number_of_characters)`
	
	Extracts a number of characters from a string, starting from the left.
	
	Example:
	`@LEFT(%%tfa_1%%,5)`

@LEN	`@LEN(text)` Returns the number of characters in a string.
@LOWER	`@LOWER(text)`

	Converts all uppercase letters in a text string to lowercase.
@MID	`@MID(text,start_position,number_of_characters)` Extracts a number of characters starting at any position.
@PROPER	`@PROPER(text)` Capitalizes the first letter in a text string and any other letters in text that follow any character other than a letter. Converts all other letters to lowercase letters.
@REPT	`@REPT(text,number_of_times)` Repeats text a given number of times. Use REPT to return a number of instances of a text string.

@RIGHT	`@RIGHT(text,number_of_characters)`
	
	Extracts a number of characters from a string, starting from the right.
	
	Example:
	`@RIGHT(%%tfa_1%%,5)`

@SEARCH	`@SEARCH(find_text,within_text,start_num)` Returns the number of the character at which a specific character or text string is first found, beginning with start_num (case-insensitive). Start_num is 1 by default.
@SUBSTITUTE	`@SUBSTITUTE(base_string,match,match_replacement)` Replaces all occurrences of *match* with *match_replacement* in *base_string*.

@TRIM	`@TRIM(text)`
	
	Strip whitespace from the beginning and end of a string.
	
	Example:
	`@TRIM(%%tfa_1%%)`

@UPPER	`@UPPER(text)` Converts text to uppercase. Example:

	`@UPPER(%%tfa_1%%)`

@URLENCODE	`@URLENCODE(query_string_param)`
	
	Returns the RFC3986-encoded version of the string passed in.
	
	Example:
	`http://www.tfaforms.com/123?tfa_Name=@URLENCODE("Mike Johnson") will output`
	`http://www.tfaforms.com/123?tfa_Name=Mike%20Johnson`

@URLDECODE	`@URLDECODE(query_string_param)`
	
	Decodes RFC3986-encoded version of the string passed in.
	
	Example:
	`http://www.tfaforms.com/123?tfa_Name=@URLDECODE("Mike%20Johnson") will output`
	`http://www.tfaforms.com/123?tfa_Name=Mike Johnson`

@YMDTODAY	`@YMDTODAY(offset,format)` Returns the current date (plus optional offset) in ISO8601/Salesforce Date format. Offset is date offset, e.g. "+1 days". Can be null. More information available here. Output will include time and be in the following format: 2017-09-05-04:00
@YMDNOW	`@YMDTODAY(offset,format)` Returns the current date and time (plus optional offset) in ISO8601/Salesforce DateTime format. Can be set to Salesforce Date format with YMDNOW("","Y-m-d"). More information available here. Output will include the time and be in the following format: 2017-09-05T17:16:30-04:00

Date and Time

@DATEDIF	`@DATEDIF(start_date,end_date,unit)` Calculates the number of days, months, or years between two dates. Unit - the type of information that you want returned: "Y" - the number of complete years, "M" - months, "D" - days, "YD" - the difference between the days of start_date and end_date (the years of the dates are ignored), "MD" - the difference between the days in start_date and end_date (the months and years of the dates are ignored). Dates may be entered as text strings within quotation marks ("2001/1/30"), as serial numbers (36921 =

	January 30, 2001), or as the results of other formulas or functions (for example, DATEVALUE("2001/1/30")).

@DATEVALUE	`@DATEVALUE(date_text)`
	
	Returns the serial number of the date represented by date_text. Use DATEVALUE to convert a date represented by text to a serial number.
	
	Example:
	`@DATEVALUE("8/22/2008") returns serial number of the text date = 39682.`

@DAY	`@DAY(serial_number)` Returns the day of a date, represented by a serial number. The day is given as an integer ranging from 1 to 31.
@HOUR	`@HOUR(serial_number)` Returns only the hour of a time value. The hour is given as an integer, ranging from 0 (12:00 A.M.) to 23 (11:00 P.M.).
@LOCALTODAY	`@LOCALTODAY()` Returns the current date according to your language and time-zone settings.
@LOCALNOW	`@LOCALNOW()` Returns the current date and time according to your language and time-zone settings.
@MINUTE	`@MINUTE(serial_number)` Returns the minutes of a time value. The minute is given as an integer, ranging from 0 to 59.
@MONTH	`@MONTH(serial_number)` Returns the month of a date represented by a serial number. The month is given as an integer, ranging from 1 (January) to 12 (December).
@NOW	`@NOW()` Returns the current date as a timestamp. This can be passed to other date functions to extract the day, month or year.
@SECOND	`@SECOND(serial_number)`

	Returns the seconds of a time value. The second is given as an integer in the range 0 (zero) to 59.
@TIME	`@TIME(hour,minute,second)` Returns the decimal number for a particular time, ranging from 0 (zero) to 0.99999999, representing the times from 0:00:00 (12:00:00 AM) to 23:59:59 (11:59:59 P.M.).
@TIMEVALUE	`@TIMEVALUE(time_text)` Returns the decimal number of the time represented by a text string. The decimal number is a value ranging from 0 (zero) to 0.99999999, representing the times from 0:00:00 (12:00:00 A.M.) to 23:59:59 (11:59:59 P.M.).
@TODAY	`@TODAY()` Returns the serial number of the current date. The serial number is the date-time code used for date and time calculations. You can use the TODAY function only as a default value; you cannot use it in a calculated column.

@WEEKDAY	`@WEEKDAY(serial_number,return_type)`
	
	Returns the day of the week corresponding to a date. The day is given as an integer, ranging from 1 (Sunday) to 7 (Saturday), by default. return_type is a number that determines the type of return value: 1 - numbers 1 (Sunday) through 7 (Saturday), 2 - numbers 1 (Monday) through 7 (Sunday), 3 - numbers 0 (Monday) through 6 (Sunday).
	
	Example:
	`@WEEKDAY("2/14/2008",2)`
	`returns day of the week, with numbers 1 (Monday) through 7 (Sunday) = 4.`

@YEAR	`@YEAR(serial_number)` Returns the year value of a date represented by a serial number. The year is given as a 4-digit integer.

Note: FormAssembly formulas honor the same date format parameters as PHP. More information can be found here.

Math and Trig

@ABS	`@ABS(number)` Returns the absolute value of a number.

@CEILING	`@CEILING(number,significance)`
	
	Returns number rounded up, away from zero, to the nearest multiple of significance.
	
	Example:
	`@CEILING(2.5,1)`
	`This will return 3`

@DEGREES	`@DEGREES(angle)` Converts radians into degrees.

@EVEN	`@EVEN(number)`
	
	Returns number rounded up to the nearest even integer.
	
	Example:
	`@EVEN(1.5)`
	`This will return 2`

@FLOOR	`@FLOOR(number,significance)`
	
	Rounds number down, toward zero, to the nearest multiple of significance. Significance is the multiple to which you want to round.
	
	Example:
	`@FLOOR(2.5,1)`
	`This will return 2`

@INT	`@INT(number)`
	
	Rounds a number down to the nearest integer.
	
	Example:
	`@INT(8.9)`
	`This will return 8`

@MROUND	

	`@MROUND(number,multiple)`
	
	Returns a number rounded to the desired multiple.
	
	Example:
	`@MROUND(10,3)`
	`This will round 10 to a nearest multiple of 3, which is 9`

@ODD	`@ODD(number)`
	
	Returns number rounded up to the nearest odd integer.
	
	Example:
	`@ODD(1.5)`
	`This will round 1.5 up to the nearest odd integer, 3`

@PI	`@PI()` Returns the number 3.14159265358979, the mathematical constant pi, accurate to 15 digits.

@POWER	`@POWER(number,power)`
	
	Returns the result of a number raised to a power.
	
	Example:
	`@POWER(5,2)`
	`This will return 5 squared = 25`

@PRODUCT	`@PRODUCT(number1,number2,...)` Multiplies all the numbers given as arguments and returns the product.
@QUOTIENT	`@QUOTIENT(numerator,denominator)` Returns the integer portion of a division. Use this function when you want to discard the remainder of a division.
@RADIANS	`@RADIANS(angle)` Converts degrees to radians.
@RAND	

	`@RAND()`
	
	Returns a random number greater than or equal to 0 and less than 1. To generate a random real number between A and B, you can try custom code similar to `RAND()*(b-a)+a`.
	
	Example:
	`@RAND()*100`
	`Returns a random number greater or equal to 0 but less than 100`

@RANDBETWEEN	`@RANDBETWEEN(bottom,top)`
	
	Returns a random number between the numbers you specify.
	
	Example:
	`@RANDBETWEEN(1,100)`
	`Returns a random number between 1 and 100`

@ROUND	`@ROUND(number,decimal_places)` Returns a number rounded to a specified number of decimal places.

@ROUNDDOWN	`@ROUNDDOWN(number,num_digits)`
	
	Rounds a number down, toward zero.
	
	Example:
	`@ROUNDDOWN(3.2,0)`
	`Rounds 3.2 down to zero decimal places = 3`

@ROUNDUP	`@ROUNDUP(number,num_digits)`
	
	Rounds a number up, away from zero.
	
	Example:
	`@ROUNDUP(3.2,0)`
	`Rounds 3.2 up to zero decimal places = 4`

@SIGN	`@SIGN(number)` Determines the sign of a number. Returns 1 if the number is positive, zero (0) if the number is 0, and -1 if the number is negative. Example:

	`@SIGN(10)` `Returns sign of a positive number = 1`
@SQRT	`@SQRT(number)` Returns a positive square root.

@SUM	`@SUM(number1,number2,... )`
	
	Adds all the numbers that you specify as arguments.
	
	Example:
	`@SUM(%%tfa_1%%,%%tfa_2%%)`

@TRUNC	`@TRUNC(number,num_digits )`
	
	Truncates a number to an integer by removing the fractional part of the number. Num_digits is a number specifying the precision of the truncation.
	
	Example:
	`@TRUNC(8.9)`
	`Returns integer part of 8.9 = 8`

Statistical

@MAX	`@MAX(number1,number2,...)` Returns the largest value from the numbers provided.
@MIN	`@MIN(number1,number2,...)` Returns the smallest value from the numbers provided.

Information

@ISBLANK	`@ISBLANK(value)`

	Returns the logical value TRUE if value is empty; otherwise, it returns FALSE.
@ISERR	`@ISERR(value)` Returns the logical value TRUE if value is any error value except #N/A; otherwise, it returns FALSE.
@ISERROR	`@ISERROR(value)` Returns the logical value TRUE if value is any error value, such as #N/A, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!; otherwise, it returns FALSE.
@ISEVEN	`@ISEVEN(number)` Returns TRUE if number is even, or FALSE if number is odd.
@ISNA	`@ISNA(value)` Returns the logical value TRUE if value is #N/A (value not available) error value; otherwise, it returns FALSE.
@ISNUMBER	`@ISNUMBER(value)` Returns the logical value TRUE if value is a number; otherwise, it returns FALSE.
@ISODD	`@ISODD(number)` Returns TRUE if number is odd, or FALSE if number is even.
@NA	`@NA()` Returns the error value #N/A. #N/A is the error value that means "no value is available."