



WebTricks Custom T-SQL Functions Date Functions

**Bringing the power and ease of ColdFusion's
built-in functions to SQL Server**

Version 1.0

User/Developer Guide

Table of Contents

About WebTricks Custom T-SQL Functions	3
T-SQL	3
About Custom/User-Defined T-SQL Functions	3
WebTricks Functions	3
System Requirements.....	4
Compatibility	4
Terms of Use.....	4
About WebTricks.com.....	4
Using Custom Functions.....	4
Creating Functions	4
Calling Functions	4
Date Functions	5
CreateDate.....	5
CreateDateTime	6
CreateTime.....	6
DateCompare	7
DateFormat	8
DayOfWeek	8
DayOfWeekAsString	9
DayOfYear	10
DaysInMonth	10
DaysInYear	11
FirstDayOfMonth.....	11
Hour	12
IsLeapYear	12
Minute	13
MonthAsString.....	14
ParseDateTime	14
Quarter	15
Second.....	16
TimeFormat.....	16
Week	17

About WebTricks Custom T-SQL Functions

This document is to familiarize you with the details of the WebTricks Custom T-SQL functions.

T-SQL

T-SQL (or Transact SQL) is the “flavor” or SQL used by Microsoft SQL Server. T-SQL is an extended version of standard SQL that contains many function calls that will only work with Microsoft SQL Server.

About Custom/User-Defined T-SQL Functions

In addition to the functions that come with Microsoft SQL Server, custom – or user-defined – SQL functions can be created to extend functionality, much the same way custom tags and user-defined functions do in ColdFusion.

WebTricks Functions

While T-SQL has many built-in functions, it does not provide the same formatting capabilities of ColdFusion’s built-in functions with which many ColdFusion developers have become familiar. Unless you have an advanced knowledge of T-SQL, it may be difficult to replicate the desired functionality.

Familiar Function Names

The WebTricks Custom T-SQL functions not only replicate the functionality of ColdFusion, they also replicate the function name and usage of ColdFusion functions, making it easy to remember how to call the functions.

When To Use

While extremely useful, custom T-SQL functions do take more processing time than built-in function calls and can slow down your query processing if not used wisely. The best time to use custom T-SQL functions are as follows:

- The query results will be passed directly to an external resource (such as a Flash application or via a Web Server) and you cannot control formatting on the receiving end
- The query results need to be processed as is – without the use of additional ColdFusion formatting
- The function will aid in processing a trigger or stored procedure

Learning Tools

Many of the WebTricks Custom T-SQL functions contain advanced T-SQL code processing and they make an excellent learning tool when creating your own user-defined functions and stored procedures.

System Requirements

The custom functions will work on any Microsoft SQL Server 2000 database. The functions can be called directly in the database, or can be used from any programming language that makes a call to a Microsoft SQL Server 2000 database.

Compatibility

The functions have been tested with Microsoft SQL Server 2000 and were compared to ColdFusion functions on ColdFusion MX 7.

Terms of Use

You are permitted to modify or customize your copy of the functions to your liking; however you may not redistribute to other parties. You are not permitted to redistribute the functions unless explicitly authorized and licensed by WebTricks/Limited Reality LLC. You are not permitted to re-post or circulate the source code, or any derivative works, on the Internet, newsgroups, in emails or in any other manner.

About WebTricks.com

WebTricks.com is a tips and tricks site for web developers specializing in ColdFusion, JavaScript and SQL. It has been a popular ColdFusion resource since 1997. Please visit us at <http://www.webtricks.com>.

Using Custom Functions

Creating Functions

Each custom function contained in this pack has its own file named *function_name.sql*. The file can be opened within SQL Server's Query Analyzer and executed to create the function.

If at any time the script is modified and you wish to update existing function you can either DROP/delete the function or modify the script to use the ALTER statement as opposed to the CREATE statement.

Calling Functions

In T-SQL, custom functions must be called using a two-part name:

```
user_name.function_name()
```

Typically, "dbo" can be used as the *user_name* value when calling the function.

Like ColdFusion, arguments passed into a function must be done within the function call's parenthesis. Since T-SQL is a typed language, the same rules that apply to parameter values in regular SQL queries apply to function arguments:

```
dbo.function_name('string_value_date', date_value, numeric_value, 'string_value')
```

Date Functions

The following functions are contained within the Date Functions pack:

- CreateDate
- CreateDateTime
- CreateTime
- DateCompare
- DateFormat
- DayOfWeek
- DayOfWeekAsString
- DayOfYear
- DaysInMonth
- DaysInYear
- FirstDayOfMonth
- Hour
- IsLeapYear
- Minute
- MonthAsString
- ParseDateTime
- Quarter
- Second
- TimeFormat
- Week

CreateDate

Purpose

Used to create a date from a supplied month, day and year

Usage

CreateDate(*numeric_value_year*, *numeric_value_month*, *numeric_value_day*)

Returns

Date Value

Date created if values create valid date, otherwise returns null

Example

Year, Month, Day	ColdFusion	SQL
2005, 6, 7	{ts '2005-06-07 00:00:00'}	2005-06-07 00:00:00.0
2005, 7, 8	{ts '2005-07-08 00:00:00'}	2005-07-08 00:00:00.0
2005, 8, 9	{ts '2005-08-09 00:00:00'}	2005-08-09 00:00:00.0
2005, 9, 10	{ts '2005-09-10 00:00:00'}	2005-09-10 00:00:00.0

Year, Month, Day	ColdFusion	SQL
2005, 10, 11	{ts '2005-10-11 00:00:00'}	2005-10-11 00:00:00.0
2005, 11, 12	{ts '2005-11-12 00:00:00'}	2005-11-12 00:00:00.0
2005, 12, 13	{ts '2005-12-13 00:00:00'}	2005-12-13 00:00:00.0

CreateDateTime

Purpose

Used to create a date/time from a supplied year, month, day, hour, minute and second

Usage

CreateDateTime(*numeric_value_year*, *numeric_value_month*, *numeric_value_day*, *numeric_value_hour*, *numeric_value_minute*, *numeric_value_second*)

Returns

Date Value

Date/time created if values create valid date/time, otherwise returns null

Example

Year, Month, Day, Hour, Minute, Second	ColdFusion	SQL
2005, 6, 7, 8, 10, 18	{ts '2005-06-07 08:10:18'}	2005-06-07 08:10:18.0
2005, 7, 8, 9, 11, 21	{ts '2005-07-08 09:11:21'}	2005-07-08 09:11:21.0
2005, 8, 9, 10, 12, 24	{ts '2005-08-09 10:12:24'}	2005-08-09 10:12:24.0
2005, 9, 10, 11, 13, 27	{ts '2005-09-10 11:13:27'}	2005-09-10 11:13:27.0
2005, 10, 11, 12, 14, 30	{ts '2005-10-11 12:14:30'}	2005-10-11 12:14:30.0
2005, 11, 12, 13, 15, 33	{ts '2005-11-12 13:15:33'}	2005-11-12 13:15:33.0
2005, 12, 13, 14, 16, 36	{ts '2005-12-13 14:16:36'}	2005-12-13 14:16:36.0

CreateTime

Purpose

Used to create a time from a supplied hour, minute and second, the date value is set to 12/30/1899

Usage

CreateTime(*numeric_value_hour*, *numeric_value_minute*, *numeric_value_second*)

Returns

Date Value

Date/time created if values create valid date/time, otherwise returns null

Example

Hour, Minute, Second	ColdFusion	SQL
8, 10, 18	{ts '1899-12-30 08:10:18'}	1899-12-30 08:10:18.0
9, 11, 21	{ts '1899-12-30 09:11:21'}	1899-12-30 09:11:21.0
10, 12, 24	{ts '1899-12-30 10:12:24'}	1899-12-30 10:12:24.0
11, 13, 27	{ts '1899-12-30 11:13:27'}	1899-12-30 11:13:27.0
12, 14, 30	{ts '1899-12-30 12:14:30'}	1899-12-30 12:14:30.0
13, 15, 33	{ts '1899-12-30 13:15:33'}	1899-12-30 13:15:33.0
14, 16, 36	{ts '1899-12-30 14:16:36'}	1899-12-30 14:16:36.0

DateCompare

Purpose

Used to determine which date is greater than the other

Usage

DateCompare(*date_value1*, *date_value2*)

Returns

Integer Value

0 if dates are equal

-1 if *date_value1* is before *date_value2*

1 if *date_value1* is after *date_value2*

Example

Date/Time 1	Date/Time 2	ColdFusion	SQL
1/1/2005	1/1/2005	0	0
1/1/2005 08:00	1/1/2005 08:00	0	0
1/1/2005 08:00	1/1/2005 08:05	-1	-1
1/1/2005 08:05	1/1/2005 08:00	1	1

DateFormat

Purpose

Used to return a date as string formatted by a supplied mask

Usage

DateFormat(*date_value*, *string_value_mask*)

Allowable mask values:

- d - Day of month with no leading zero
- dd - Day of month with leading zero
- ddd - Day of week string as three letter abbreviation
- dddd - Full day of week string
- m - Month with no leading zeros
- mm - Month with leading zeros
- mmm - Month string as three letter abbreviation
- mmmm - Full month string
- yy - Year as last two digits with leading zero
- yyyy - Full 4 digit year
- short - m/d/y
- medium - mmm d, yyyy
- long - mmmm d, yyyy
- full - dddd, mmmm, d, yyyy

Returns

String Value

Formatted date

Example

Date	Mask	ColdFusion	SQL
02/28/2005	short	2/28/05	2/28/05
02/28/2005	medium	Feb 28, 2005	Feb 28, 2005
02/28/2005	long	February 28, 2005	February 28, 2005
02/28/2005	full	Monday, February 28, 2005	Monday, February 28, 2005
02/28/2005	mm/dd/yy	02/28/05	02/28/05
02/28/2005	m/d/yyyy	2/28/2005	2/28/2005

DayOfWeek

Purpose

Used to determine the ordinal value of the day of the week of a date

Usage

DayOfWeek(*date_value*)

Returns

Integer Value

1 - 7 representing the day of the week
(Sunday - Saturday respectively)

Example

DayOfWeek:

Date	ColdFusion	SQL
6/7/2005	3	3
7/8/2005	6	6
8/9/2005	3	3
9/10/2005	7	7
10/11/2005	3	3
11/12/2005	7	7
12/13/2005	3	3

DayOfWeekAsString

Purpose

Used to determine the day of the week, as a string, for a day value (1-7)

Usage

DayOfWeekAsString(*numeric_value*)

Returns

String Value

Day of the week if value supplied is 1-7, otherwise, null

Example

Weekday	ColdFusion	SQL
1	Sunday	Sunday
2	Monday	Monday
3	Tuesday	Tuesday
4	Wednesday	Wednesday
5	Thursday	Thursday
6	Friday	Friday

Weekday	ColdFusion	SQL
7	Saturday	Saturday

DayOfYear

Purpose

Used to determine the ordinal value of the day of the year of a date

Usage

DayOfYear(*date_value*)

Returns

Integer Value

1 - 366 representing the day of the year

Example

Date	ColdFusion	SQL
6/7/2005	158	158
7/8/2005	189	189
8/9/2005	221	221
9/10/2005	253	253
10/11/2005	284	284
11/12/2005	316	316
12/13/2005	347	347

DaysInMonth

Purpose

Used to determine the number of days in a particular month

Usage

DaysInMonth(*date_value*)

Returns

Integer Value

Number of days in the month

Example

Date	ColdFusion	SQL
6/7/2005	30	30

Date	ColdFusion	SQL
7/8/2005	31	31
8/9/2005	31	31
9/10/2005	30	30
10/11/2005	31	31
11/12/2005	30	30
12/13/2005	31	31

DaysInYear

Purpose

Used to determine the number of days in a particular year

Usage

DaysInYear(*date_value*)

Returns

Integer Value

Number of days in the year

Example

Date	ColdFusion	SQL
1/1/2000	366	366
1/1/2001	365	365
1/1/2002	365	365
1/1/2003	365	365
1/1/2004	366	366
1/1/2005	365	365
1/1/2006	365	365

FirstDayOfMonth

Purpose

Used to determine the ordinal value of the first day of the month of the supplied date

Usage

FirstDayOfMonth(*date_value*)

Returns

Integer Value

Value representing the ordinal of the first day of the month of the date

Example

Date	ColdFusion	SQL
6/7/2005	152	152
7/8/2005	182	182
8/9/2005	213	213
9/10/2005	244	244
10/11/2005	274	274
11/12/2005	305	305
12/13/2005	335	335

Hour

Purpose

Used to determine the hour of a date

Usage

Hour(*date_value*)

Returns

Integer Value

0 - 23 representing the hour of the date

Example

Time	ColdFusion	SQL
08:10:18	8	8
09:11:21	9	9
10:12:24	10	10
11:13:27	11	11
12:14:30	12	12
13:15:33	13	13
14:16:36	14	14

IsLeapYear

Purpose

Used to determine whether or not the supplied year is a leap year

Usage

IsLeapYear(*int_value*)

Returns

Bit Value

0 if not a leap year

1 if a leap year

Differences from ColdFusion

Returns 0/1 as opposed to yes/no

Example

Year	ColdFusion	SQL
2000	YES	1
2001	NO	0
2002	NO	0
2003	NO	0
2004	YES	1
2005	NO	0
2006	NO	0

Minute

Purpose

Used to determine the minute of a date

Usage

Minute(*date_value*)

Returns

Integer Value

0 - 59 representing the minute of the date

Example

Time	ColdFusion	SQL
08:10:18	10	10
09:11:21	11	11
10:12:24	12	12
11:13:27	13	13
12:14:30	14	14

Time	ColdFusion	SQL
13:15:33	15	15
14:16:36	16	16

MonthAsString

Purpose

Used to return the name of the month, as passed in as 1-12

Usage

MonthAsString(*int_value*)

Returns

String Value

 Name of the month

Example

Weekday	ColdFusion	SQL
1	January	January
2	February	February
3	March	March
4	April	April
5	May	May
6	June	June
7	July	July
8	August	August
9	September	September
10	October	October
11	November	November
12	December	December

ParseDateTime

Purpose

Used to return a date value from a date formatted as a string

Usage

ParseDateTime(*string_value*)

Returns

Date Value

Raw date if the string was a valid date

Null if the string was not a valid date

Differences from ColdFusion

There is no conversion attribute

Example

Date/Time	ColdFusion	SQL
1/1/2005 13:25	{ts '2005-01-01 13:25:00'}	2005-01-01 13:25:00.0
January 1, 2003 6:00PM	{ts '2003-01-01 18:00:00'}	2003-01-01 18:00:00.0
5/16/73	{ts '1973-05-16 00:00:00'}	1973-05-16 00:00:00.0
8:00 AM	{ts '1899-12-30 08:00:00'}	1900-01-01 08:00:00.0

Quarter

Purpose

Used to determine the quarter of a date

Usage

Quarter(*date_value*)

Returns

Integer Value

1 - 4 representing the quarter of the date

Example

Date	ColdFusion	SQL
6/7/2005	2	2
7/8/2005	3	3
8/9/2005	3	3
9/10/2005	3	3
10/11/2005	4	4
11/12/2005	4	4
12/13/2005	4	4

Second

Purpose

Used to determine the second of a date

Usage

Second(*date_value*)

Returns

Integer Value

0 - 59 representing the second of the date

Example

Time	ColdFusion	SQL
08:10:18	18	18
09:11:21	21	21
10:12:24	24	24
11:13:27	27	27
12:14:30	30	30
13:15:33	33	33
14:16:36	36	36

TimeFormat

Purpose

Used to return a date as string formatted by a supplied mask

Usage

DateFormat(*date_value*, *string_value_mask*)

Allowable mask values:

- h - Hour (12 hour clock) with no leading zeros
- hh - Hour (12 hour clock) with leading zeros
- H - Hour (24 hour clock) with no leading zeros
- HH - Hour (24 hour clock) with leading zeros
- m - Minutes with no leading zeros
- mm - Minutes with leading zeros
- s - Seconds with no leading zeros
- ss - Seconds with leading zeros
- l - Milliseconds, three digits
- t - Time marker: A or P
- tt - Time marker: AM or PM

- short - h:mm tt
- medium - h:mm:ss tt
- long - h:mm:ss tt
- full - h:mm:ss tt

Returns

String Value

Formatted time

Differences from ColdFusion

The three-letter time zone value returned in long and full is not available

Example

Time	Mask	ColdFusion	SQL
02:12:24	short	2:12 AM	2:12 AM
02:12:24	medium	2:12:24 AM	2:12:24 AM
02:12:24	long	2:12:24 AM EST	2:12:24 AM
02:12:24	full	2:12:24 AM EST	2:12:24 AM
02:12:24	h:m:s	2:12:24	2:12:24
02:12:24	hh:mm:ss tt	02:12:24 AM	02:12:24 AM
02:12:24	HH:MM:SS	02:12:24	02:12:24
02:12:24	hh:mm:ss:l	02:12:24:0	02:12:24:0
02:12:24	H:M:S:l	2:12:24:0	2:12:24:0

Week

Purpose

Used to determine the ordinal value of the week of a date

Usage

Week(*date_value*)

Returns

Integer Value

0 - 53 representing the week of the date

Example

Date	ColdFusion	SQL
03/07/2005	11	11
03/14/2005	12	12

03/21/2005	13	13
03/28/2005	14	14
04/04/2005	15	15
04/11/2005	16	16