



Date Formatting in Dynamic Text

Zephyr Associates, Inc.
P.O. Box 12368
Zephyr Cove, NV 89448

775-588-0654
800-789-5323
Fax 775-588-8423

www.styleadvisor.com

Format() For Dates and Time Periods

We have a separate lesson for formatting numbers in which we go over the syntax for :Format{}. I know that you don't want to hear this, but there is a completely different syntax for :Format when you're working on dates and time periods. Instead of squiggly brackets you use parentheses and the strings inside the parens have to be in quotes. It's like this:

```
:Format("Format String")
```

Date Format Strings

For Date objects the built-in keywords are:

```
"QQ"  
"MM"  
"0MM"  
"Mon"  
"Month"  
"YY"  
"YYYY" or "YEAR"  
"DD"  
"0DD"  
"WEEK"  
"WK"
```

You can add your own text to the date format just by including it in the quotes. For example, if you want commas between the date and the day of the week you can do something like this:

```
<%  
Today():Format("Month DD, WK")  
%>
```

And you'll see a result like "January 30, Wed".

Let's go over each of these keywords one at a time.

"QQ" returns the quarter in the format "Q1" (or "Q2", "Q3", and "Q4").

```
<%  
Today():Format("QQ")  
%>
```

Displays "Q1" because it's January while I write this.

"MM" returns the month formatted as one or two digit numbers from 1-12.

```
<%  
Today() : Format ("MM")  
%>
```

Displays “1” because January is the first month.

“0MM” returns the month formatted the same way as “MM” except that it will pad with zeros so that all months show two digits (01-12).

```
<%  
Today() : Format ("0MM")  
%>
```

Displays “01” because January is the first month and I asked for leading zeros for months less than 10.

“Mon” returns the month as its three letter acronym with the first letter capitalized, from “Jan” to “Dec”.

```
<%  
Today() : Format ("Mon")  
%>
```

Displays “Jan” because that’s how we abbreviate “January”.

“Month” returns the month completely spelled out with an initial capital from “January” to “December”.

```
<%  
Today() : Format ("Month")  
%>
```

Displays “January”.

“YY” returns the last two digits of the year. You have the better part of a thousand years before you have to worry about Y3k problems with your DT, so maybe it’s safe to use this version.

```
<%  
Today() : Format ("YY")  
%>
```

Displays “08” for the current year.

"YYYY" or "YEAR" returns all four digits of the year.

```
<%  
Today() : Format ("YYYY")  
%>
```

Displays "2008" for the current year. So does

```
<%  
Today() : Format ("YEAR")  
%>
```

"DD" returns the day of the month from 1-31.

```
<%  
Today() : Format ("DD")  
%>
```

Displays "30" for the current day because I'm typing this on the thirtieth.

"0DD" returns the day of the month padded with leading zeros for days lower than the tenth of the month from 01-31.

```
<%  
Today() : Format ("0DD")  
%>
```

Displays "30" for the current day because I'm typing this on the thirtieth but on the third it would return "03" rather than "3".

"WEEK" returns the day of the week with initial caps from Monday to Sunday.

```
<%  
Today() : Format ("WEEK")  
%>
```

Displays "Wednesday" for the current day because I'm typing this on Wednesday the thirtieth.

"WK" returns the three letter abbreviation of the day of the week with initial caps from Mon to Sun.

```
<%  
Today() : Format("WK")  
%>
```

Displays "Wed" for the current day because I'm typing this on Wednesday the thirtieth.

You may combine these keywords and whatever separators you like. For example:

```
<%  
Today() : Format("WEEK is the DDth day of Month")  
%>
```

Will return "Wednesday is the 30th day of January". Please note that this code will only work for dates higher than 3 since in English we have special endings for the first three ordinals. You'd need to test for which day of the month it was and create a special case for the first three to get this block to work in all cases.

I'm game, let's try it:

```
<%  
Tod = Today() : Format("DD")  
  
if Substring(Tod, -1) == 1 and (Tod < 10 or Tod > 20) then  
  Ord = "st"  
elseif Substring(Tod, -1) == 2 and (Tod < 10 or Tod > 20) then  
  Ord = "nd"  
elseif Substring(Tod, -1) == 3 and (Tod < 10 or Tod > 20) then  
  Ord = "rd"  
else  
  Ord = "th"  
end  
  
Today() : Format("WEEK is the DD" .. Ord .. " day of Month")  
%>
```

How about that? See what's happening there? First we set a variable called "Tod" to the day of the month. Then we check to see if the current day of the month is 1 or 21 or 31. If so we set a variable called "Ord" (short for ordinal) to the string "st" as in "1st". If it's not, we check to see if "Tod" is 2 or 22 and if so set "Ord" to "nd" as in "2nd". If it's not 2 we check for 3 or 23 and set "Ord" to "rd" as in "3rd". If it isn't any of those three we set "Ord" to "th". Then we concatenate the value of "Ord" into our format string where we originally had "th". Substring(Tod, -1) returns the last character in "Tod". In this case we're looking for 1, 2, or 3 as the last character, but we need to exclude 11, 12, or 13 to adjust for the fact that all the teens take "th" as their ordinal ending.

Time Period Keywords

Formatting Time Period objects uses the same syntax (thank goodness) but a different set of keywords. For review, a Time Period object is a length of time, rather than a single date. `AnalysisLength()` returns a Time Period. Here are the keywords useful for formatting Time Periods:

```
"PCount"  
"Periods"  
"Period"  
"YCount"  
"Years"  
"QCount"  
"Quarters"  
"MCount"  
"Months"  
"DCount"  
"Days"
```

"PCount" returns the number of periods in the time period. Which period is used depends on the periodicity of the analysis.

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("PCount"))  
%>
```

Would return something like: "119" but won't tell you 119 whats. Not very useful if you don't know what units are being displayed.

"Periods" returns the name of the period that "PCount" counts. It will use the same capitalization as you use in the word "Periods". It will also pluralize the period name grammatically. Period names will be one of "Day", "Month", "Quarter", "Year" and their plurals.

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("PCount Periods"))  
%>
```

Will return something like "119 Months" if your periodicity is monthly.

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("Pcount periods"))  
%>
```

Will return something like "119 months". Notice that the word "months" is lower case because "periods" was not capitalized in the format string.

"Period" returns the name of the period that "PCount" counts. It will use the same capitalization as you use in the word "Period". It will not pluralize the period name. Period names will be one of "Day", "Month", "Year".

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("PCount Period"))  
%>
```

Will return something like "119 Month" if your periodicity is monthly.

"YCount" returns the number of years in the time period.

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("YCount"))  
%>
```

Would return something like: "9" but won't tell you 9 whats. Not very useful if you don't know what units are being displayed.

"Years" returns the word "Year" properly pluralized and with the same capitalization as used in the format string. You'll get "Year" if the "YCount" is 1 and "Years" if it's more.

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("YCount years"))  
%>
```

Will return something like "9 years". Notice that the word "years" is lower case because "years" was not capitalized in the format string.

We'll skip examples of the remaining strings. They work just like the ones already described, for quarters, months, and days.

You can include several counts in the same formatted time period and anything that isn't one of the keywords listed will just appear in the string. You could say something like this:

```
<%  
Duration = AnalysisLength()  
Display(Duration:Format("YCount years and MCount months"))  
%>
```

For a result that looks like "9 years and 11 months".

It may take some playing to get these formatted time periods to come out neatly. What if the time period above was an even number of years with no months remaining? Then you'd get a result that looks like "9 years and 0 months". That's fairly silly looking. You could brush up on your string handling and search the results for " 0 months" before displaying it and leave it off if you find it, like this:

```
<%  
Duration = AnalysisLength()  
Replace(Duration:Format("YCount years and MCount months"), "  
0 months", "")  
%>
```

What this does is replace the string " 0 months" with an empty string (""). If the string doesn't exist in the formatted time period, nothing happens to it. Notice that I included the blank space before the "0".

Okay, just to make things more complicated, you can actually include three separate format strings, one for each periodicity (daily, monthly, quarterly in that order). DT will use the appropriate format string for the periodicity of the analysis.

Default Date Format Variables

We know that whole “providing three separate format strings” is complicated so we’ve also provided you with a set of variables with which you can change the default date formats, either for a single block of code or for an entire workbook:

```
dailyDateFormat = "Week, Month DD, Year"  
monthlyDateFormat = "Month Year"  
quarterlyDateFormat = "Month Year"
```

I’ve included the default settings for these variables above.

Use these variables where the format string would appear in :Format()

For example, let’s see how these work.

```
<%  
BeginDate()  
Display(" | ")  
BeginDate():Format(dailyDateFormat, monthlyDateFormat,  
quarterlyDateFormat)  
%>
```

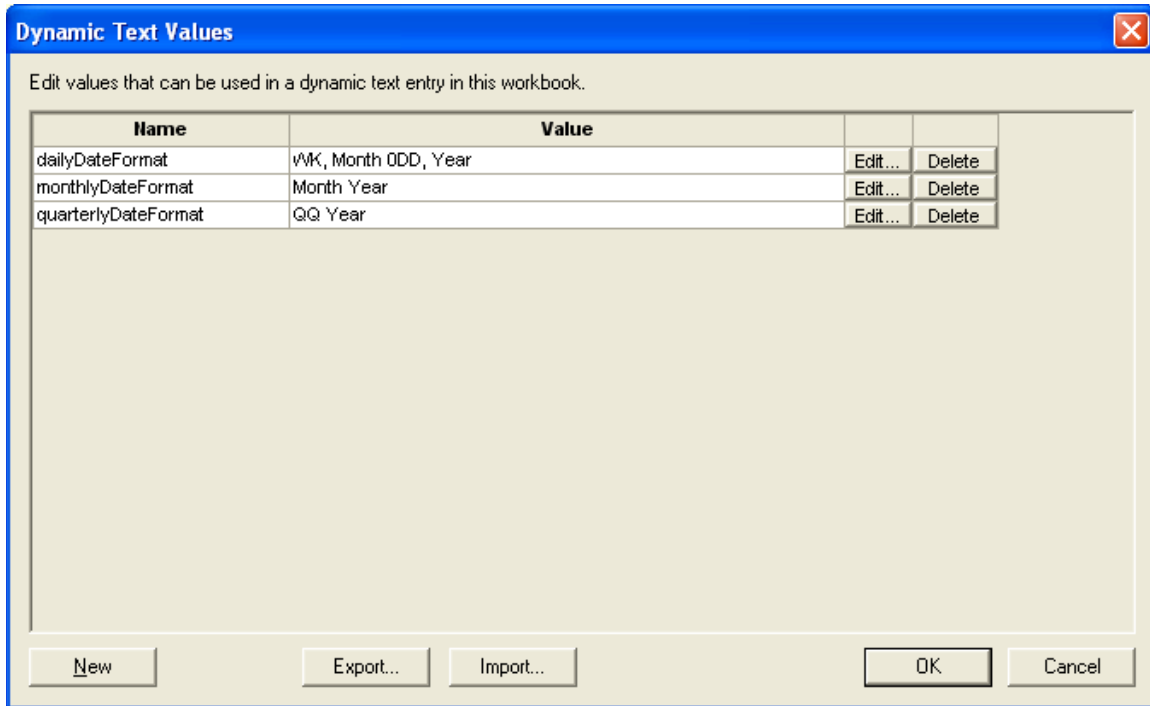
BeginDate() uses the system default date format of “Mon ODD, YYYY”. The second instance of BeginDate() will use the appropriate format string for the periodicity of your analysis. In this case, if the analysis was monthly or quarterly it would use “Month Year”. If the analysis was daily it would use “Week, Month DD, Year”. That is, the above example would show you something like this “Jan 01, 1998” then on a new line “January 1998” for a monthly or quarterly analysis and for daily the second line would be something like “Thursday, January 1, 1998”

If you wanted to change the date format for a block of code and you don’t control the periodicity of the data, you could set these variables at the beginning then use the variable names instead of writing out three format strings whenever you need display a date:

```
<%  
dailyDateFormat = "WK, Month ODD, Year"  
monthlyDateFormat = "Month Year"  
quarterlyDateFormat = "QQ Year"  
  
BeginDate():Format(dailyDateFormat, monthlyDateFormat,  
quarterlyDateFormat)  
%>
```

In this example a daily analysis will show something like “Thu, January 09, 1998”, monthly would show “January 1998” and quarterly would show “Q1 1998”.

If you wanted these new default values to be available everywhere in your workbook you’d need to set them in Dynamic Text Values for Workbook. Select Edit|Dynamic Text Values for Workbook and fill in the dialog like this:



Now you can use these variables instead of spelling out the date format string in all of your DT in this workbook. The above example could be shortened to:

```
<%  
BeginDate():Format(dailyDateFormat, monthlyDateFormat,  
quarterlyDateFormat)  
%>
```

Here’s a trick that’s fairly helpful. If you have the three variables defined as workbook values, you can leave :Format() empty and it’ll use those three format strings by default:

```
<%  
BeginDate():Format()  
%>
```

Pretty easy, huh?