Evaluator Reference

A simple calculator or a powerful programmable computer

Organizational Development Strategies Incorporated
1850 Sawgrass Pointe Dr
Hayes, VA, 23072
(804) 642-3538

PBlais@ODStrategies.Org

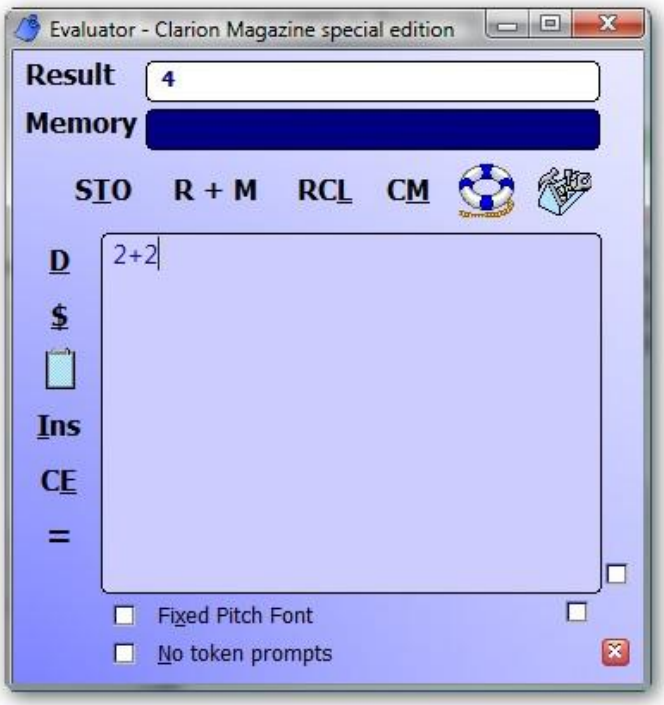
Contents

Basic Evaluator features

A quick reference to basic operations

The Evaluator screen

A traditional calculator has many buttons but Evaluator provides a calculator workspace that you can type your mathematical expression and then lets you calculate the answer as well as edit the formula.



Pressing your= key or clicking the= icon with the mouse will display the result above in the Result line.

You may use all the common math notation symbols combined with numbers:

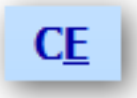
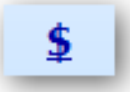
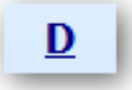
+	Plus key to perform addition
-	Minus key to perform subtraction
*	Asterisk key to perform multiplication
/	forward slash to perform division
^	Carat to perform exponentiation
&	String concatenation
%	Modulus (remainder from a division)
()	left and right parenthesis for grouping

The current Evaluator workspace is limited to formula length of 6,000 characters. For long expressions you may require left and right parenthesis to group the computation

properly. Computing your formula ignores all spaces, tabs, comments, and line returns. This lets you type in a column or type across the entire line in a way that makes it easier for you to read. The workspace resizes to fit your screen, as you require by dragging the edges and corners of the window.

Evaluator buttons

If you type a formula and press the equal key you will see your answer in the Result line. The rest of the buttons on the Evaluator allow more advanced features. The following are the features supported by these buttons.

	**Clear entry** The CE icon when clicked or selected with Alt E clears the current formula pad. Select a second time to clear the result line too.
	**Solution** The equal icon and equal key request the computer to solve your formula. The solution is displayed on the result line.
	**Copy result** The Copy result icon copies the current result line into your system clipboard. You may then paste the result in any application.
	**Dollar format** Dollar format button or Alt D will replace the result line formatted to two decimal places and add a dollar sign.
	**Store** The Store icon or Alt S copies the result line to the memory line.
	**Result plus memory** Result plus memory or Alt R adds the current result line to the memory line.
	**Results Clear** Results clear or Alt L will remove the value from the Results line.
	**Clear memory** Clear memory or Alt M clears the value in the memory line.
	**Format as a date** Format the Result as a calendar date or Alt D(see advanced

	**Insert function** Insert function (Alt I) inserts a built in function to the Evaluator workspace. See advanced features.
	**Help screen** From the help screen you can launch the XPS or PDF formatted help document.
	**Configuration options** Wallpaper, colors and fonts are user defined. A set of 6 predefined themes are available and you may customize those as you require. Configurations are saved between uses (see Configuration options chapter).

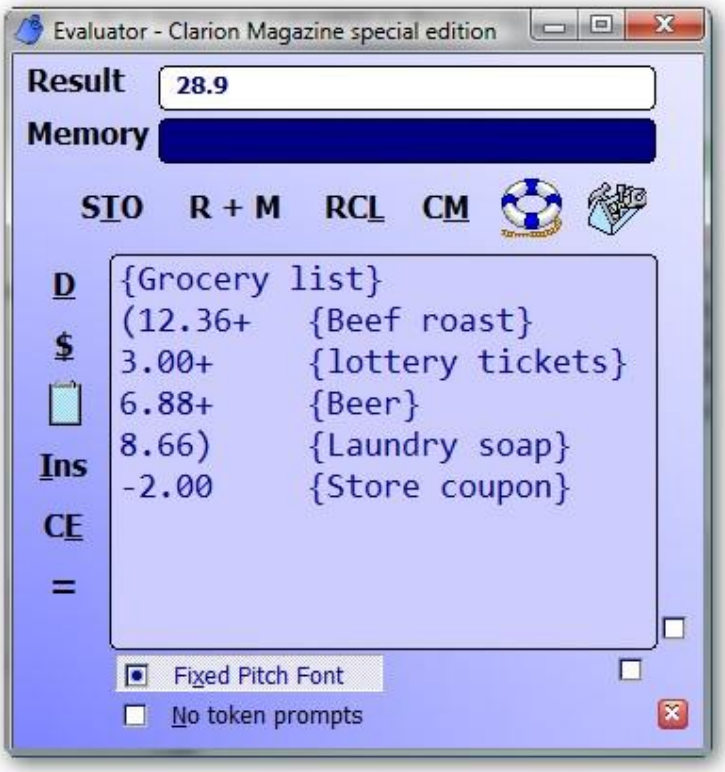
Using the Memory and Result lines

Within any formula you may use the contents of either the Memory or Results lines in a new formula by typing in the variable name of Results or Memory. These are predefined variables for your use. You may even use them to develop recursive formulas such as the following scenario:

Result	100
Memory	1
Evaluator workspace	Memory + Results

Each time the equal key is pressed the formula will compute the result causing the Display to increment the result by one each time the equal key is pressed. The above example will count from 100 by one for each press of the equal key. You may use these variables any place you might use a number.

Adding comments



You may add comments any place inside the formula as long as you always start them with a curly brace and end them with a curly brace. The comment may span multiple lines and neither the curly braces nor anything in between is used for computation. The above is a grocery list with the items identified. Comments are not required but are useful when cutting and pasting the formula.

Basic features summary

Use the formula pad to type any expression then press the = key or icon. To sum a list of number just type them separated by a plus sign then press equal. You may choose to add enter keys or tabs to make it easier to read and proof. If any one number is incorrect just edit the number and reselect the equal key or button. Copy the result to the clipboard to paste it in another application. With these basic features you can perform all the functions on any basic calculator.

Once you have gained comfort with basic operations you can explore the advanced features of built in functions.

Advanced Evaluator features

A reference to advanced operations

Using built in functions

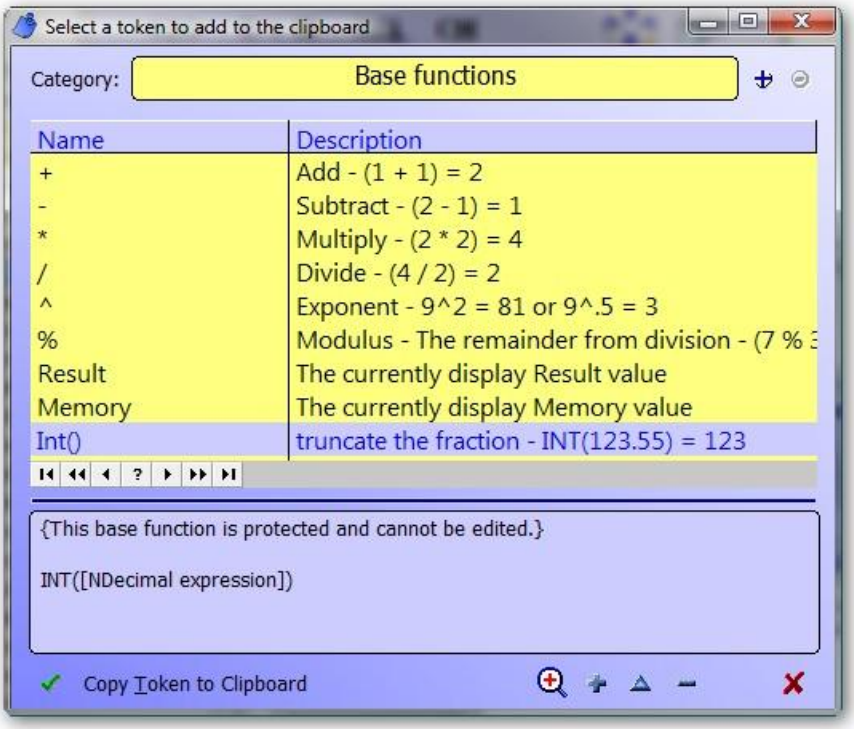
The insert function button permits you more advanced functions within your own formulas you build in the Evaluator workspace. Using advanced features you can nest functions and perform more complex solutions such as trigonometry, calendar mathematics, and string manipulations with less typing. You may also save commonly used forumals for later recall.

Before we can begin the advanced discussion we need to cover some of the basic formula requirements that set the limit of what you can do. Each time you press the solution button (equal key) the formula in the Evaluator workspace solves as one expression that yields only one result. You do not have the ability to write lines of code as you do with a programming language but you will see you can still accomplish a lot of power and write easy to use equations. In this way you are only writing one line of computer code – it can be a big one however!

The Evaluator incorporates a run time evaluation library created by Soft Velocity makers of the Clarion programming family of development tools. All parameters and functions start as text values implicitly converted from string to real numbers, new strings, integers, binary, hexadecimal, and octal formats as demanded by the functions that contain them. All results are string data that can be passed as input to another function. In this way we can mix and nest functions to perform many calculations. It also means the whole complexity of converting data from one type to another is all automatic.

It all begins with the Ins button

By selecting the Insert button (Alt I) a screen of functions appears. Functions are sorted by categories then sorted alphabetically within. They display in color based on



the defined colors for that category. Selection of a row in the table highlights the selected function and displays its formula below the list box. Double clicking or selecting and pressing Enter or Alt T will return you to the workspace and paste the formula into the Evaluator workspace.

Built in parameter prompts

In the above screen you will see the INT() function has been selected on the screen and the formula below shows as:

INT([NDecimal expression])

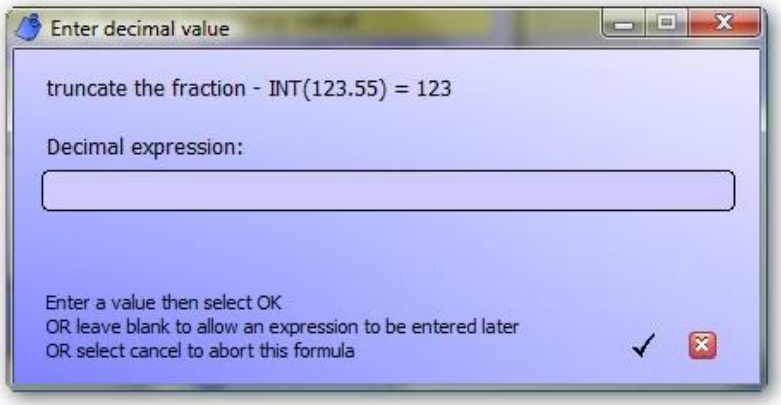
The INT() function accepts any value and truncates the fractional part of the number. The number123.45 is truncated to123. To use this in a formula you could type:

INT(123.45)

The formula you see above includes some extra characters:

[NDecimal expression]

This is the prompt component to the built in function. When you double click the INT()function in the list you will be displayed a second screen.



The Window title is instructing you to enter a decimal value. The very first letter of the text in the parameter determines what type of data the formula requests for this particular parameter. In this case the “N” signifies that the data type is decimal.

The first line of text in the window is the Description attribute of the function. It tells the user what function they are using.

The prompt text above the entry field is directly from the formula prompt text. Text enclosed in square brackets makes the “Decimal expression:” prompt you see in the above screen. The text has a colon added automatically.

The entry field is a free form entry field. You could enter a number or you may need to enter a complex expression that eventually will be used in the INT() function.

If you Click the OK icon (or enter key) without a value entered Evaluator pastes the whole formula into the Evaluator. You could use this feature when building complex formulas using base functions. The second option is to enter a number or an expression. It will substitute the prompt text for your data entry. The third option is to cancel (escape key) and nothing will be inserted into the Evaluator workspace.

Other functions may have as many prompts as many as required for the formula. When the formula gets complicated, prompting for them makes it easy for anyone to perform complex computations.

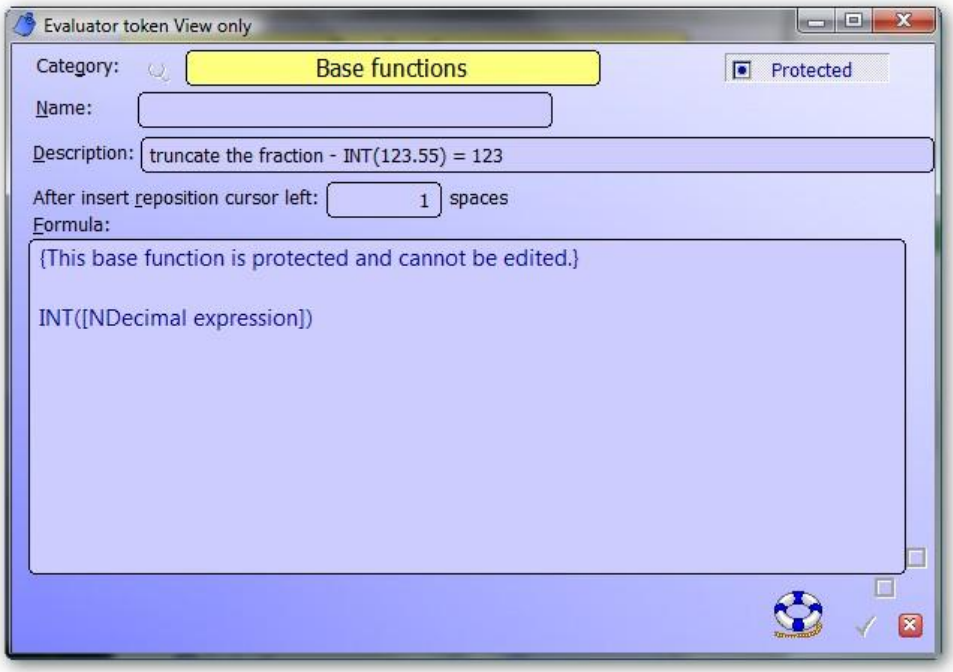
Turning of the prompts

For the simple base functions like INT() you may quickly grow tired of the prompts. You have two choices. The first option is to just type the function in the Evaluator workspace manually and not use the Insert function button OR click on the “No token prompts” button on the main Evaluator screen.

You may then manually type the contents for the function into the Evaluator workspace without a prompt. The Evaluator creates protected functions the first time the program runs. Protected functions may not be changed and display: “{This base function cannot be edited.}”. All functions you create can be change and will always prompt for the defined parameters.

Create and edit built in functions

The add, change, and delete buttons at the bottom of the screen provide access to the attributes of the built in functions and / or let you make new functions. Most of the functions installed are protected and cannot be modified. Formulas you create can be added to the list.



Function attributes

A function is comprised of the following parts:

Name — A short description of the function. This appears in the left most column of the display list. Use this when you make a new function so it is easier to find.

Description — This is a longer description with an example if possible to act as documentation for the user. It displays at the top of any user prompt fields you create.

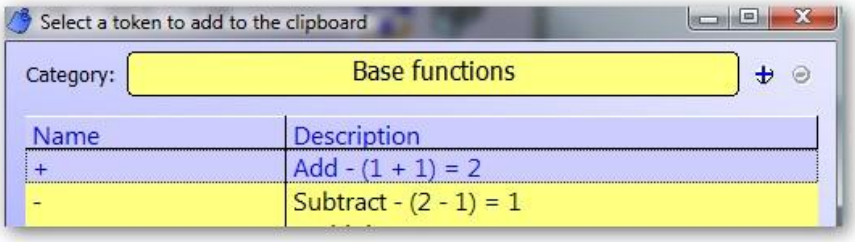
Formula — The actual formula inserted into the Evaluator pad. This is the actual code that is used and may or may not contain user prompts. Extra spaces and enter keys can be added to make the formula easier to read. Extra characters are inserted into the Evaluator workspace. Comments are not inserted into the workspace. Spaces used inside the square brackets will remain to make the prompt read correctly. Your formula can be up to 6000 characters long including any spaces, comments, or prompts.

Cursor reposition — The number of spaces to move the cursor to the left after insertion allows the user to enter the function parameters by positioning the cursor where the first parameter belongs. For example, the INT() built in command has a value of 1 so the cursor will locate between the left and right parenthesis. This is where you would add a number or an expression. Leaving this value blank or zero will locate the cursor at the end of the command after insertion. This simple feature makes using a formula much smoother.

Category — The category makes it easier to find functions. You may create new categories for your own functions. The category name and description are only for organization and documentation. Creating your own functions and categories you perform regularly saves you time and improves accuracy. You may choose to change the existing category colors but you may not delete the base categories or edit a built in functions when it shows the protected icon in the upper right corner. Making your own versions of these same functions is allowed. Combining functions into larger functions can be a great time saver too.

Defining and using function categories

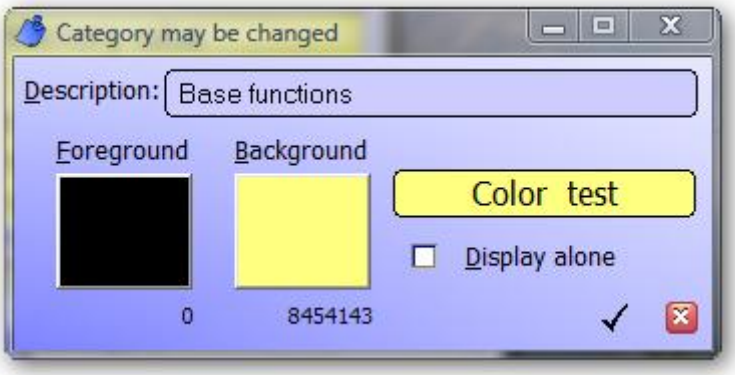
From the Evaluator workspace selecting the Ins icon allows you to pick functions that will be inserted for you. The browse list box can hold a large number of functions. To make it easier to find a function you can narrow the list by clicking on the “plus icon” or selecting “Alt plus”.



This will reveal a list of all the defined categories.

Selecting a single category will limit the browse list to only those categories. The "—no category —--" is for all the functions that don't have a category. By default all the functions that come with Evaluator has one of the above categories. You may change or add categories as you like but you may not delete a category if that category is in use.

Adding and changing categories



The base functions category ca not have its name changed but the colors may be changed to suit any theme you choose. Clicking the desired foreground or background sample will allow you to select a different color. The Color test sample will display how the choices will appear above the browse as each function is selected.

Display Alone means that the functions in this category will only be displayed when that particular category has been selected as a filter. By default all categories are displayed when no category has been selected. Changing all categories to use this Display alone feature would require each category to be used as a filter in order to see those functions. The feature can be used to organize and make your display simpler to use.

Adding comments

You may add comments as you like to the function or formulas you write. Comments display in the browse and in the edit screen. Comments do not insert into the Evaluator work space. Prior to inserting the function into the Evaluator workspaces, all comments, line feeds, and carriage return characters are removed. When you write complex functions you can store documentation inside the function but not clutter the Evaluator workspace when using it. Base functions that ship with Evaluator are protected and not editable. You also see the protected icon at the top of the edit screen.

Making user prompts

The format for a user prompt is shown below:

[<type of data><The prompt that the user sees>]

The first and last characters must be square brackets. You may not use square brackets as part of the prompt or for any other purpose.

<type of data> The first letter in the user prompt will change the window title that pops up to ask for user prompted data. Using an unrecognized type will ignore the type and let you enter anything. Using the wrong type won't restrict what you can enter but will confuse a user of your function. The following are valid types and may be upper or lower case:

- N Numeric data that can have a decimal point
- I Integers (whole numbers) with no decimal point
- A An angle that can be a decimal value in degrees
- S String data that can include letters or number

<The prompt that the user sees> Starting with the second character and up to but not including the closing square bracket is prompt displayed to the user of your base function. These prompts are substituted after the user supplies the data for that prompt in all instances.

When a user prompt that repeated in more than one place in the formula the prompt only is presented once. You must make sure that these prompts are completely identical because they are case sensitive.

When prompting for a pure text string such as the following function:

UPPER([SEnter your name])

If the user enters:

John Smith

The function will fail because the UPPER() base command requires a text string for input or an expression. The error is because the text "John" is not seen as text but as a variable or function. The error displayed in the result will be "! BIND has not been called for John". If you expect only pure text and never want a text expression it is better to define your function as follows:

UPPER('[SEnter your name]')

In this revision we added single quotes on either side of the user prompt so whatever the user enters will be treated as pure text. If you expect to enter other functions as an answer to the prompt such as:

SUB('John Smith',1,4)

This would yield the following pasted in the Evaluator

UPPER('SUB('John Smith',1,4)')

The correct way it should look is like this:

```
UPPER(SUB('John Smith',1,4))
```

You only need to be concerned with this issue when you are working with text. The last example following this chapter will provide more details about using text and string data in the Evaluator. Working with quotes is the important issue when you use the Evaluator to compute strings. For numbers only it's never an issue.

Tips for making new functions

Create a manual example in the Evaluator itself to test your ideas. When the example works, copy and paste it into a formula and add the short name and description. Then test it again to make sure it still works. As a last step replace all the fixed values with user prompts and test it again.

All text and string data must be enclosed with single straight quotes (the apostrophe key). If a quote itself is part of the data you must precede the quote with another quote. Working with quotes can be a bit confusing so you need to test each step as you go along.

All trigonometric functions SIN(), COS, and TAN() operate in radians not degrees and ASIN(), ACOS, and ATAN() return radians not degrees. Use the conversion variable DEG2RAD and RAD2DEG to switch between degrees and radians by using the appropriate conversion and multiplying the degrees or radians. Several of the included area formulas already installed in Evaluator actually require radians. Keeping the two straight will make sure your results are accurate. See these examples to learn how to use them.

Inserting functions summary

The ability to insert built in functions opens the door to more complex solutions with less typing. By using the prompted parameters anyone can insert the proper data where it belongs. Once you have inserted functions you can still manually add and edit more functions.

A more detailed example

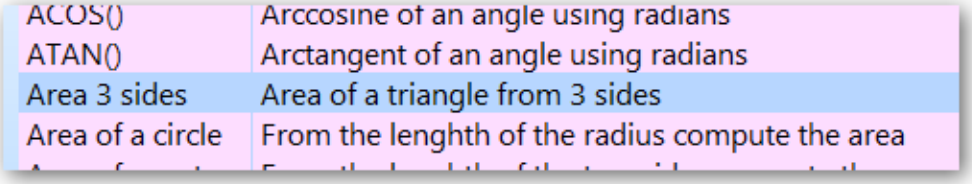
A formula for computing the area of a triangle from the length of 3 sides

One of the more complex functions included with the Evaluator is the formula for computing the area of any triangle. The formula is derived from Heron's formula combined with the Law of Cosines. You enter only the lengths of the three sides in any order for the area formula to be written to your Evaluator work space. The last step is to press the equal key or button. You could also type a plus sign and repat the function to add up many triangles. This is where the power of functions can tackle larger problems. Let's take a closer look at this function included in the Trigonometry category.

Step 1 From a blank Evaluator select the Ins button (or Alt I keys).

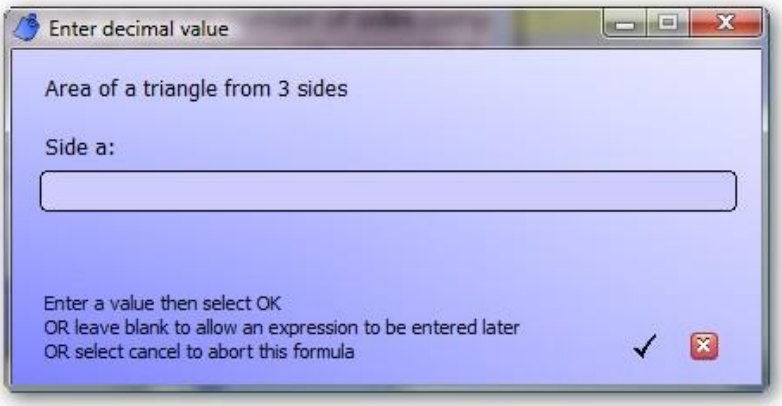
Step 2 From the list of functions scroll down and select:

Double click it, or click then select the green checkmark or click and press the Enter key. Any of these three methods will select the function.

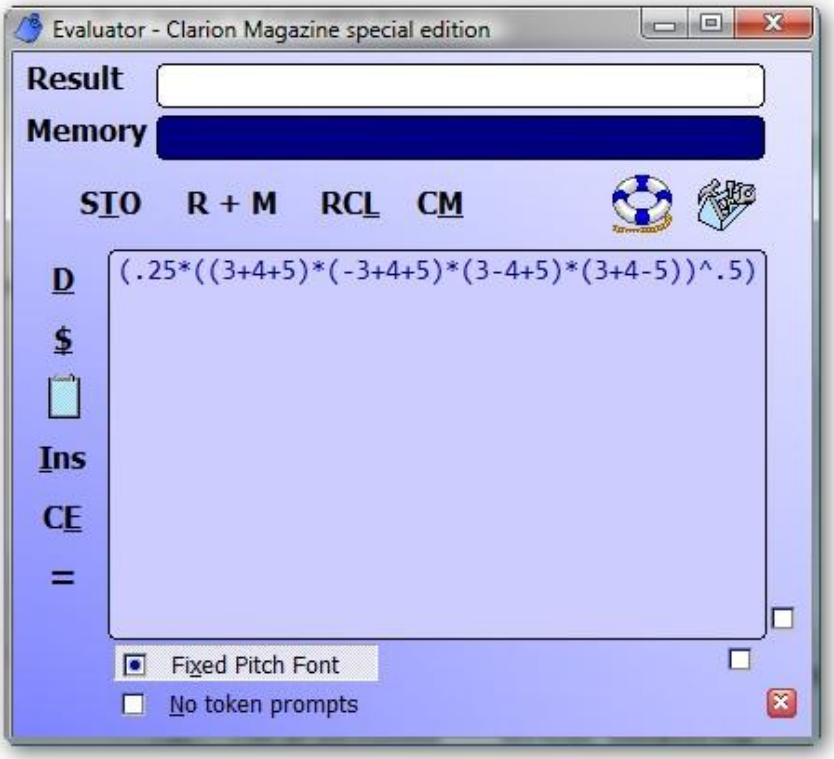


Step 3 A little background on this function will help you understand the rest of the example. To compute the area of a triangle you need to know the length three sides of triangle. You could type in three random numbers but you may find out that sometimes it won't compute properly. In order for the triangle to be a real triangle the sides must connect at the three vertices. If any side is longer than the sum of the lengths of the other two it cannot be a triangle connected at 3 vertices. Our function has to assume that you are not going to enter invalid data. As long as the data entered is real then our formula will work. Invalid data will always yield zero.

The formula will now prompt us to enter the length of the three sides.



For each side a, b, and c we are prompted to enter a length. For our example I'll use the values 3, 4, and 5. These values will define a proper triangle and will let us complete our example. I will also show an alternate way to solve this triangle and prove our formula really works.



Step 4 After we enter the last side we return to the Evaluator workspace our formula appears.

The Evaluator prompts for three side lengths but inserts each side length four times each. Your typing efficiency is now 4 times greater using the Evaluator.

Step 6 The last step is to press the equal button or the equal hot key to display the result. The result is 6 of course. We know that a triangle with sides 3, 4, 5 units long form a perfect right triangle and the area of a right triangle is equal to the two short side multiplied together and divided by 2. So the check for this example is (3*4)/2. If you press the CE button (Alt E) you can enter the check formula as well and get the same result of 6. Using both methods we can now be sure our formula works properly.

Step 7

We have now used a formula one time and that demonstrates the usefulness of our triangle function but we can go farther. Suppose we need to compute the total area of 25 triangular areas so we can estimate the amount of paint we need to buy. In our above example enter a + key in the Evaluator workspace after the previous formula and insert another triangle. You can then compute the area of 2 triangles added together. Continue adding more triangles and in this way sum many triangles. You also have the added benefit of copying your formula and pasting it in a document to save it for later. Add comments to the formula about which triangle is for what and now you have the documented numbers with the formula that computes a proper total.

Other Evaluator base functions

Dates and text strings (open a Evaluator and follow along)

Math functions are a handy feature of the Evaluator but we can do other operations as well. The Evaluator can operate on other types of data and we will now examine a second type of calculation that you might actually use in Evaluator.

Some background information

All projects start with some idea of a required task and the rules to reach a correct solution. Our next example is a real problem I had while creating a software application. I wanted to generate a default title to be used when creating log entries in a data record. I also wanted it to be user defined not something I hard coded. Users always know what they don't like even when they can't tell you what they want. This user defined feature allows the user to create an expression just like an Evaluator formula that is executed each time a log entry is added to a specific record. The product even includes a copy of Evaluator so they can write the expression they want. What follows here is the documentation about the tool used to create that expression. I personally use the expression for my weekly timesheet records. What follows is really a case where: The programmer wrote a program that needed a user defined expression. He wrote another program to write expressions that are define to the first program so he can bill clients that buy programs. These are the serious issues when writing user defined features in software. We are now prepared for the problem.

The mission for this default title started with a need to create a weekly time log I always create on Monday for the week ending on Sunday the previous day. It seemed to me there should be a way to do it automatically so they would all look uniform and always have a Sunday date.

The tools in the box

When you explore the Evaluator tool box you find text handling tools as well as some calendar handling tools. With this in mind I was ready to make the formula expression that could make exactly what I needed. The design was to make the following string:

```
Week ending MM/DD/YYYY
```

I could have just typed the above into the default description for the logs and that is exactly what they would look like. This is close but I wanted the expression to substitute the MM, DD, and YYYY for the real month, day.

Working with dates

Evaluator has a built in function you can use in your formula to compute calendar dates. It's called:

```
TODAY()
```

The TODAY() function returns the current date stored in your computer. In addition the function CLOCK() will return time. If you try it in your Evaluator this is what you would find as the result on the date this documentation was first written - September 26,2007. Your date won't be the same unless your computer is set to the wrong date since I know I wrote this months before you installed the software.

75512

Interesting, but you may be thinking “How does that help me and what does 75512 mean?” Computers don’t always make sense but they do have a standard way of dealing with dates that is quite accurate and useful. To understand what 75512 means I’m going to tell you that a computer date of 4 means January 1, 1801 and it was a Thursday. From here we can compute all the rest of our days longer than any of us will survive.

To work our solution we need to know not only the date but the day of the week as well. One of the interesting functions is the Modulus function. It is written like the following:

75512 % 7

The answer you can compute in your Evaluator workspace by typing it in right now. You will see the correct result is 3. When you divide this number by 7 the remainder is 3. Modulus throws away the answer to the division but returns the remainder instead. In our daily lives we don’t often need the remainder without the result but in computing we commonly do. Any number divided by 7 will always have a modulus of between zero and six. 7 divided by 7 has a remainder of zero and 3 divided by 7 has a remainder (a Modulus) of 3. When combined with the TODAY() function it tells us the one thing we really need – the day of the week. If the modulus of the date is zero we know the day of the week is Sunday and if the modulus is 6 we know it is Saturday and the days in between follow in order.

DayName(Today())

Enter the above into your Evaluator and press the equal key. (DayName() is included in a BIND expression). You can type this everyday and it will always tell you the day of the week. It does the modulus function for you and returns back the day of the week as text based on your computers calendar. Your Evaluator now computes text as well as dates! This is not what you normally consider a function of a calculator.

All this exercise is to get to know how dates in the computer work relative to the Evaluator. It means we now have a way to compute the current date and in turn the day of the week, We are in great shape to take on this project.

Working with strings to get the job done

Our mission also requires us to assemble a text string that we will use for our default log description. The way we add text strings is not too unlike adding numbers together. Type the following into your Evaluator workspace:

‘Week ending ‘ & TODAY()

The above is a text “expression”. Actually everything you enter into the Evaluator workspace is an expression. Note that I added a space after the letter g and before the final quote. Copying the text from this document requires you to replace the open and closing quotes with plain straight quotes. It will display an answer in the result line:

Week ending 75512

Your date won’t be the same as above because it is the date when you type not the date when I wrote the example. Now this is getting more like what we want to see but

not quite. Let's recap the process of making strings then move on. The text 'Week ending ' is enclosed in single quotation marks. Notice there is a space after the letter g. The next part is the ampersand symbol. This is called a concatenation operator. It's like a plus sign for joining text expressions together. It joins the results from the TODAY() function we showed earlier with the text string we just demonstrated.

Convert numbers to meaningful text

Just having the correct number is not always the total mission. It is often a requirement that the number look properly in addition to being accurate. The date is the best example since no one knows what 75512 means as far as a date on the calendar is concerned. Fortunately, we have a powerful function called FORMAT(). There are many formats for all sorts of numbers, dates, text and time. We will use just one for this project.

FORMAT() requires just two parameters. The first is the data to format and the second is called a "picture". A picture is the set of instructions for reformatting the data. The FORMAT() function will return formatted data as text the way we want it to look. Let's examine the following format.

```
FORMAT(75512,@D2)
```

If you paste this into the Evaluator you will see it displays the proper date - 9/26/2007. This is what we need to make a date number look like date text instead of number text.

Using all the tools together

Now we are ready to write the expression that will work the way we want. The following is the first version.

```
'Week ending' & FORMAT(TODAY()-1,@D2)
```

If you cut and paste be sure you are using single quote characters. It should now display a result that looks like this (except with today's date not the following old date):

```
Week ending 9/25/2007
```

The above expression takes the current date and subtracts 1 day and formats it as a date and appends it to the string 'Week ending '. Because I said I always make the entry on Monday, this will display the date for Sunday. Sorry, I but forgot to tell you something. I have asked you to write an expression but I failed to provide all the information. Sometimes I don't make the log on Monday. I might not do it until Wednesday. We now have to make another adjustment and for that we need to understand the days of the week from a computing perspective. If you already know your days of the week in order by day you are ready for this part.

If you read back to the early discussion about days of the week, you will recall that when the Date % 7 is equal to zero we know the date is a Sunday. We are now going to add a little trick to make sure I never get a date that isn't a Sunday.

```
'Week ending' & FORMAT(TODAY() - (TODAY()%7) ,@D2)
```

I'll now explain how this works to return the prior Sunday no matter what the date is currently. If the Modulus of any date can only be 0 through 6 and Sunday is always a zero; we can subtract the modulus of the current date from the date to obtain the date

of the prior Sunday unless the current date is a Sunday and we will get the same date. Subtracting the modulus 7 of any date from the date is always going to compute “last” Sunday unless it already is Sunday. That will work for our solution just fine.

Just when we thought we had the mission completed it turns out I forgot something else. I don’t always do the log on Monday or even on Tuesday. Sometimes I even do it early! We need some logic to help get it right. The real complete rule is if I make an entry on Monday, Tuesday or Wednesday then I’m late! If I make it Thursday, Friday or Saturday then I’m early for the week ahead. If I make it on Sunday then that was the correct date. Now no matter what the date is we can now know what to do. We can be sure I can’t change my mind again - really.

We really need some tool we have not discussed yet and it is a tool called CHOOSE(). CHOOSE() solves a single logical expression that determines which of two other expressions then apply. In computer programming it’s called a branch. This one tool alone is all we have to work within our Evaluator toolbox to perform a “logical branch”.

To speed up the example I’m going to tell you that the logical expression we want to use is:

```
TODAY() % 7 < 4
```

The above rule says anytime the date modulus 7 is less than 4 (0,1,2,or 3) then the condition is TRUE otherwise it is FALSE. To us, a modulus value of zero to three means the current date is Sunday through Wednesday otherwise it must be Thursday through Saturday. Let’s see what the new formula looks like:

```
 'Week ending ' & FORMAT(
TODAY()
-  TODAY() % 7
+ CHOOSE( TODAY() %7 <  4 , 0, 7)      {<-the new part is here}
 ,@D2)
```

I have added line breaks and a comment in the formula so it is easier to read. As written this will actually work even with the comment added. Let us walk through the parts of the formula:

‘Week ending ’	is a single text expression we always want no matter what. Notice we include a space after the letter g
FORMAT(	begins the second part of the text we want to make because we are going to compute the date for our text then convert it into a readable date formatted string. What goes inside is what we will compute, but the result of that numeric expression is what the FORMAT() function requires.
,@D2)	concludes the FORMAT() function by specifying the type of formatting we want. In our final expression we want the date displayed in the format mm/dd/yyyy so that format picture we require is@D2.The reference chapter at the end covers all the various format pictures. There are many date formats and maybe you would like to try others.

TODAY() – TODAY() % 7 + CHOOSE(TODAY() % 7< 4, 0, 7) is the computed "numeric expression" that is our date we want to display. Let's take that apart into the sub expressions that make this large expression:

CHOOSE(TODAY() % 7 < 4, 0, 7) The CHOOSE() command is the new numeric expression we added. It will compute the modulus of the current date then compare the result to 4. If it less than 4 the CHOOSE result is zero. If the CHOOSE() evaluated expression is false the result is the numeric expression 7. This form of CHOOSE()returns the first expression value if true and the second expression value if false. See the command reference section for details about CHOOSE(). Choose only operates on a "logical expression" but can return either a text or a numeric result. The result can also be another expression and could include an inner CHOOSE() expression thus letting us perform many branches and sub branches. With practice it opens up a lot of different computations.

Here is a recap of how CHOOSE works for our problem. If we add 0 to the date of "last Sunday" it still is the same date, but if we add 7 it will be Sunday 7 days later. We have the problem or being early or late solved using CHOOSE().

What makes this all work are the rules of precedence (see command reference chapter). The CHOOSE() command needs to be resolved before the rest of the expression can be resolved so the evaluation begins by examining the expression inside the choose command then works its way out through the rest of the function. Using parentheses can help you keep clear in your mind when you write these functions as well as define ambiguous expressions.

This expression can be cut and pasted into the Log record configuration setting. Notice we have no user prompts because we will only need the current date. We can generate our log descriptions week after week on any date and always get the correct date. This expression will resolve leap year too!

Summary

The critical issue in any project is to reduce the problem to its most basic rules after you gather all the information you need. When you understand the mission you can better solve the problem. Writing functions to make computations work faster and accurate are both good reasons to spend time learning the Evaluator features.

We often need to pull several functions together to get not only the correct answer but the correct answer formatted so it looks correct too. You can make your own default Log entry descriptions using the Evaluator to design and test the expression and then paste it in the configuration Log setting so it works for you all the time.

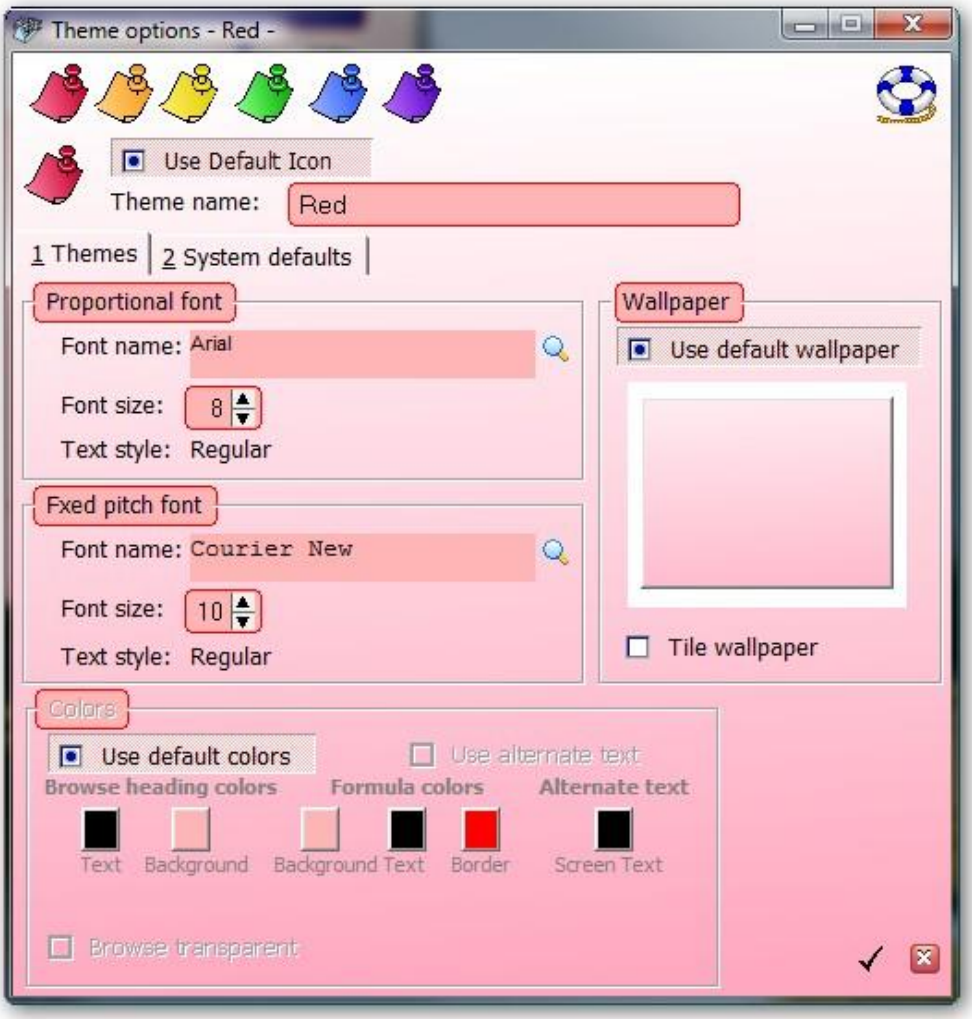
The Evaluator is a fun tool to play with math and other expressions since the tool itself can test your solutions. If you find a useful expression you can save it to the Evaluator for use in the future. Write a good function one time and spend the rest of your days being lazy. It's what we call being a lazy programmer. Sometimes being lazy is a good thing.

User configuration

Making the application look the way you want it to look

User options

The look and feel of the Evaluator can be modified to suit your desktop themes or just how you prefer to work. Different screen resolutions and color schemes require fine tuning. Select the tool box icon with your mouse of press the F2 function key to examine or change the user settings.



The top row of icons represent 6 default themes you can apply. Selecting any of them will change the screen you are looking at to match that theme. Changing a theme or switching to a different theme will be saved for the next time you use Evaluator. The default themes are represented with a corresponding icon in 6 colors. These colors

form the basic idea behind each theme. Each theme can be edited and so you may choose to customize only one or all the themes.

Each theme has a Themes tab and a system default tab. The system defaults apply to all themes and the themes tab are unique for each theme.

Default icon

The icon is displayed in the upper left corner of all the Evaluator windows. If you prefer a different icon uncheck the Use Default Icon check box and click on the icon to open a file selection dialog. You may select any icon you like.

Theme name

The name of the theme is only used on these screens. You may change the name of your theme to any name you prefer.

Proportional font

The Evaluator work space may use a proportional font or a fixed pitch font of your choice. Select the lookup button to pick a font installed on your computer, then adjust the size. When selecting a font you may also choose both a font, style and size. Then adjust the size alone after. The sample text next to the font name displays the selected font name using all the other options you choose.

Fixed pitch font

Proportional fonts are easier to read but fix pitched font line up numbers in columns easier. You may switch between two fonts from the Evaluator window so you may want to have both your favorite types of fonts available. The font name is displayed in the selected font using the colors and other options you select.

Wallpaper

The backgrounds of all the Evaluator screens use a common wallpaper. These may include any .GIF, .BMP, .JPG, or .WFM file you like. By default they are stretched to fit the screen but may be tiled if you prefer. The default wallpaper check box locks the settings. To use a different wallpaper uncheck the Use default wallpaper check box then click the sample image to open a file selection dialog box. Then select the Tile wallpaper check box if you desire that option. The current window will then change to reflect your choice.

Colors

The default check box locks the color settings to the default values. To change the colors uncheck the check box to enable the settings.

Browse heading colors control the headings above the built in functions browse as well as the categories browse. The foreground is the color of the text and the background is the background behind the text. Making one color dark and the other color light will ensure the text is easy to read. Click on the color sample to display a color selector.

Formula colors control the Evaluator workspace and any entry fields. The text and background colors and the border colors adjust to suit the desired theme. Click the color samples to change the colors. The changes will be reflected on the current screen. Adjust them until they look attractive and are easy to read. Having a good contrast between the foreground and background is important. Should your choices become totally incoherent, reselect the Use default colors to start over.

Use alternate text

By default all text not controlled by the formula colors is black. Should you desire to make all text a different color you must enable the use of the alternate text color by first selecting the check box and then select the color. As above try to choose colors that contrast well in order to make the screen readable.

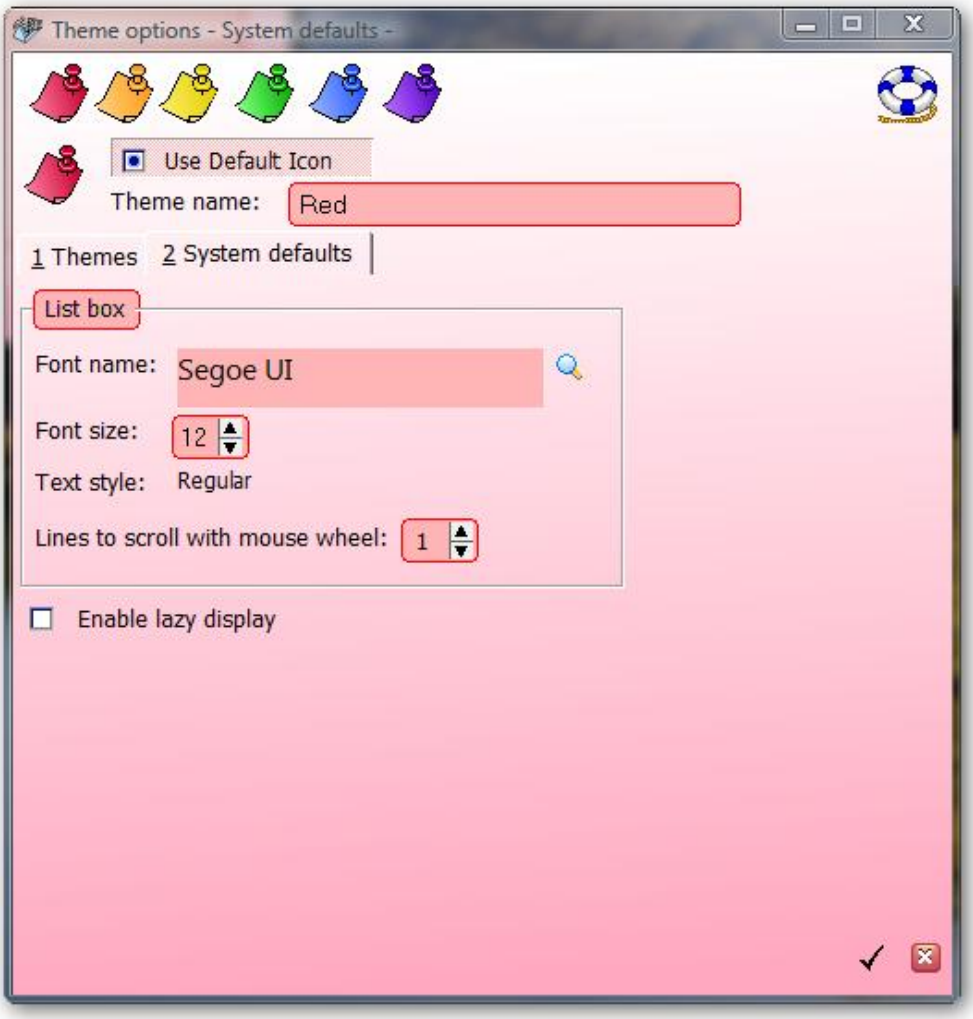
Titles transparent

The check box controls the titles above the browse lists. These titles are the categories that are displayed as you select a function. The default is to display the title with the foreground and back ground color of the category. You may disable the background color here if you desire. None of the background colors in the categories will be applied.

Browse transparent

The space within the list box can be the default color of white or is can be transparent. Functions within the list box are colored by their category colors but blank space is not colored. This will change the left over white space to transparent.

System defaults



System defaults apply to all themes and you may not hsave different settings for these options for each theme. These settings are applied to both the categories and Functions list boxes.

List box font

The list box font may be selected the same way you select the other theme fonts. This font is use inside the list box. You will be limited to a font between 6 points and 14 points. Adjust the size to suit your screen resolution.

Lines to scroll

Some comptuer mice drivers will allow you to control the scrolling within the list box and other drivers will not. Some mice drivers won't let you use this setting and some will.

Enable lazy display

Some video drivers can cause more flicker to appear and this check box tends to reduce that problem. You may try either setting to decide which looks best for your system.

User settings summary

The look of Evaluator is under your control. You can choose from the 6 predefined themes or customize them as you like. In addition to the colors and settings most all the windows can be resized by dragging an edge or a corner. Settings are retained between each time you use Evaluator. The Theme settings are stored in an XML file called s2.xml and the system defaults are stored in an xml file called s1.xml. The screen positions are stored in a file called CWVal.ini. All of the files within Evaluator are stored in the installation directory. Evaluator will not write to any other part of the system or registry.

Uninstalling Evaluator

If you choose to uninstall Evaluator you may delete the entire directory where you have installed the product. Deleting the .ini and .xml only files will return the settings to their default condition on the next launch. Deleting the formulas.tps and categories.tps files will recreate them as installed the next time you run Evaluator.

Comments and support

Send any comments or support questions to Paul Blais using my email address PBlais@ODStratgeies.Org.

Third party products and other credits

Several Clarions third party products were used in the creation of Evaluator.

Snazzy by John Christ is available free from Steve Parkers download web site was used to make the colored browse headers. There is an embed point that will allow you to make the colors definable at run time.

SuperStuff by Boxsoft is available through Mitten Software was used to do all the resize functions and a few other tricks.

xFiles from Capesoft Inc. available on their web site was used for XML settings support. The s1 file loads to a threaded group structure and S2.xml loads to a global threaded queue.

All the other user settings are only pure Clarion code. Using a large XML structure you can externalize a great many settings and provide a very user definable environment.

EVALUATE() is of course a part of Soft Velocity's Clarion development environment. Evaluator was developed in version Clarion 6.3.