Formatting and error checking expressions in FileMaker

You may be designing databases, which store FileMaker calculations in fields to evaluate them later. This may be to feed the MBS FileMaker Plugin functions like HotKey.SetEvaluate, MenuItem.SetEvaluate, SyntaxColoring.AddContextMenuCommand or one of the many others which take an expression to evaluate in case something happens.

Format

Now if your database runs on a Mac and MBS FileMaker Plugin is installed, you can use the syntax coloring of our plugin to format the formula, e.g. like this script:

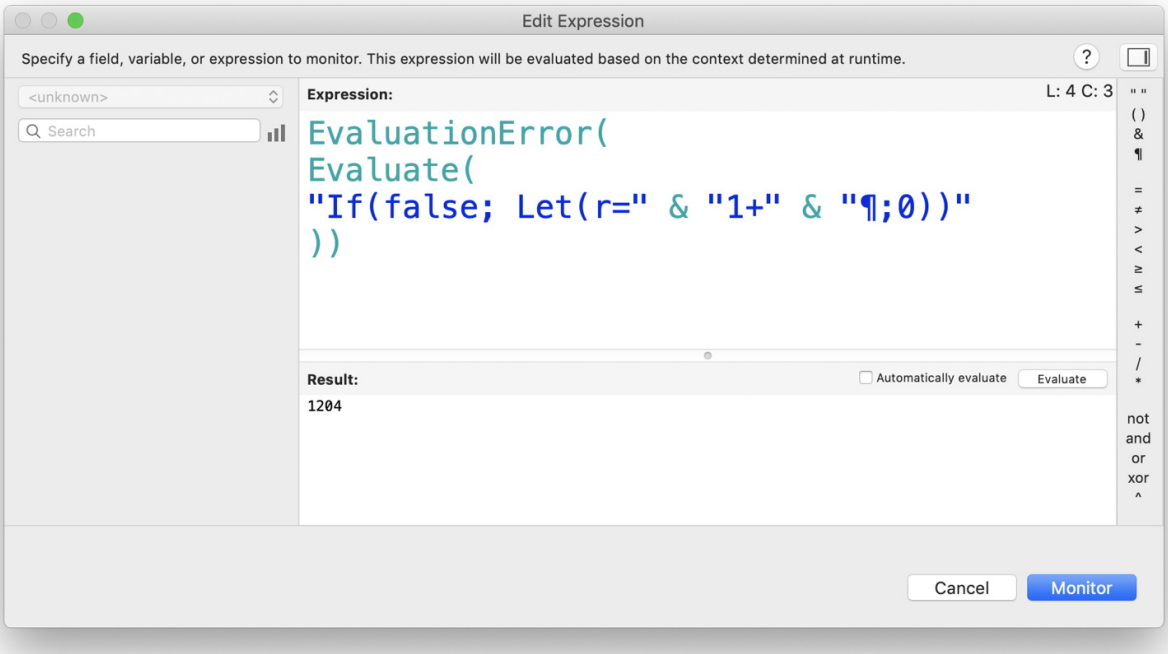
```
If [ MBS( "IsMacOS" ) = 1 ]
    Set Variable [ $r ; Value: MBS( "SyntaxColoring.Format";
EvaluateTest::Expression; 0 ) ]
    Set Field [ EvaluateTest::Expression ; $r ]
End If
```

The check For MacOS is to make sure we call it on MacOS and to check if the plugin is installed at all. If the plugin is missing, this will fail as a condition and then the format function is not called. The SyntaxColoring.Format function uses the syntax coloring definitions the plugin uses for formatting the expressions in the calculation dialog and you can change those. See fmSyntaxColorizer or other databases which provide an interface on the MBS Plugin function SyntaxColoring.AddTag to register those rules.

Check Error

To check for errors you may want to use EvaluationError on the result of Evaluate function. But in the expression you may want to have a Let wrapped in an If(false; ...) expression. The If makes sure the code will go through the parser for the syntax check, but not actually run. The Let with the expression makes sure we wrap your expression properly and you don't mess with the If. In case there are comments in the expression, the ¶ will block any // comment causing an error.

Here we test the If/Let evaluate combination:



Finally we put all in a script to run when expression field is saved and we put the error number into a field in our script:

Set Error Capture [On]
Set Variable [$expression ; Value: "If(false; Let(r=" & EvaluateTest::Expression & "¶;0))"]
Set Variable [$e ; Value: EvaluationError(Evaluate($expression))]
Set Field [EvaluateTest::Error ; $e]

To get a list of the error codes possible, please check the list in the documentation in the 1200 to 1225 range.