Coding guidelines

Make it readable

The coding style guidelines on this site are intended to increase the clarity and readability of your FileMaker code. Careful consideration has been given to the fact that it's all too easy to simply "code-n-go" within FileMaker's calculation dialog box. The information on this site is not here as an imposition. It offers a structure which provides liberation from future mental processing.

As of current, FileMaker does not provide a "code editor" inclusive of beneficial features such as syntax highlighting, code completion, block disclosure (for hiding segments of code) and code checking using a linter. Therefore, it is up to the developer to compose clean code which can easily be read not only by the primary developer but anyone else who may need to make sense of what was done.

The current tools available to FileMaker developers include external editors and snippet or keyword expansion tools.

The following is a very simple example of unreadable vs. readable code. This is a very common example of a FileMaker calculation used to provide user feedback such as a message - either within the user interface or some form of output. The goal is to identify if the current customer is tagged within the category of "prospect" and show a message accordingly.

Unreadable code

```
Let ([ CustomerPhone = Customers::phone ; Categories = List (Customers::Categories) ; Name = Customers::
FirstName & " " & Customers::LastName];
If ( PatternCount ( "¶" & Categories & "¶" ; "¶prospect¶" ) ; "Please contact " & Name & " at " & Customers::
Phone ; "" ))
```

Readable code

```
Let ( [
        ~customerName = Interface » Customers::firstName & " " & Interface » Customers::lastName;
        ~customerPhone = Interface » Customers::phone;
        ~categories = List ( Interface » Customers::categories );
        ~categoryMatch = "prospect"
];
        If ( PatternCount ( ¶& ~categories &¶ ; ¶& ~categoryMatch &¶ ) ;
                "Please contact the " & ~categoryMatch & " " & ~customerName & " at " & ~customerPhone
        )
)
```

Things to note about the code above

- At a glance, the readable code shows you you're dealing with the customer name, phone number and categories.
- You can immediately jump to the output "result" portion of the Let() function and actually "read" the output.
- By putting portions of code on additional lines and indenting, readability increases. The If () function uses the "test" portion on the same line as the function name itself and each of the possible outcomes would appear on their own lines - or multiple if required. In this case, the default is not even required and will only return a result if the test portion evaluates true.
- The code is written in a DRY fashion. In order to use this same code for a "vendor" instead of a "prospect" the only value to change is ~categoryMatch. The output will update accordingly.
- The use of the sigil ~ (tilde) to identify the locally scoped Let() variable makes it highly identifiable within the code. Easily distinguished from TableOccurrence, field names, $localVariables and $$GLOBAL.VARIABLES.
- The space between the semicolon at the end of the If () test portion indicates the continuation of code where the semicolon at the end of the variable declaration ~categories indicates a "break" in visual/mental processing of the code.

All of these types of visual clues contribute to code that is considered "elegant" and easy to read.

Key objectives

- Encapsulate your code — having to change things in many places is a pain
- Write highly cohesive code by using strongly named grouping patterns - reuse is the goal
- Make it DRY — If you see it appear a second time then make it so it will only have to be changed in one place (***D***on't ***R***epeat ***Y***ourself)
- Avoid tight coupling — i.e. write code with fewer dependencies
- Handle all possible errors

Fixing calculation code

If you currently have messy code and want a head start on code cleanup you can use this tool FileMaker Calculation Formatter. While it does not adhere to the standards outlined here. It does a lot of the hard word for you.

Assumptions

It is assumed that, as a serious FileMaker developer, you are using FileMaker Pro Advanced with access to custom functions and other advanced features. Taking full advantage of FileMaker requires this investment.