Data structures in FileMaker with MBS Plugins

MBS Plugin offers a couple of data structures which make scripting in FileMaker more like a programming language. And I know a few of you are keen on writing long scripts with using various plugin provided functions:

Variables

For years we provide global variables. They are independent of the open file and persists until the current application quits. You can use them everywhere the plugin runs, even on the server. You can store values there from one file and query later from other file. On a server you can set something in a script called by one client and later query in a script called by another client via Perform Script On Server. Of course Server Script Engine process uses a different set of variables than the web direct process.

So you call MBS("FM.VariableSet" ; "myVar"; "Hello") in a script and later query it back using MBS("FM.VariableGet" ; "myVar"). The plugin preserves data types, so a container should survive as well as dates and numbers without converting to text. You can also query list of variables, clear them or check if one is defined.

QuickList

In FileMaker you may be familiar with lists. You can put texts together with new line character to a list. But when using lists you pass around a big block of text. All functions acting on the list must parse the list, do their work and build the big text again.

To speed this up, we created QuickList functions. They allow you to create a list in memory, where the plugin can random access entries and add new values efficiently. It is a list of text and we usually parse it only once, e.g. by passing existing list to QuickList.New function. You can use other functions to add entries via lists, SQL query or values. You can sort, remove duplicates, reverse order, serialize or match with regular expression.

By just skipping the parse/output part, the QuickList functions are usually much faster than our similar List functions, which do use same functions for the core work.

Dictionary

The dictionary functions provide a hash map, so you can store values based on keys. Values are stored in memory and kept with their original data type. Lookup are usually faster than database lookups, as everything is in memory and using binary search to find items.

You can create a dictionary with passing key and value pairs and add entries via SQL requests. We can serialize the dictionary to store it, output to JSON or XML.