Don't Pass Commands (to functions and scripts) as Parameters

From a comment I wrote on the Coding Guidelines page before the Proposals section existed:

One guideline I’ve advocated in the past that might fit here is this:

Don’t pass commands (to functions and scripts) as parameters. If you want to be able to call on two different operations, write two different functions or scripts.

For example, the syntax for Tom Robinson’s UUID function is “UUID ( type )”, where “type” specifies whether to use version 1 (timestamp and MAC address) or version 4 (random numbers) of the RFC 4122 standard. I suggest that this would be better written as two functions, “UUID.v1” and “UUID.v4”.

Here’s why. FileMaker’s main selling point to developers is that it’s faster to work with than just about any other development platform out there. There are several ways FileMaker accomplishes this, but I think the most unique is what I call “the palette of available commands.” In an Edit Script window, every script step you can use in a script is listed on the left. In Specify Calculation dialogs, every field, operator, and function you can use in the calculation is listed in the top of the dialog. If you don’t know exactly what you’re looking for, it’s still right there; you don’t have to look it up unless the name of a function isn’t enough to tell you what it does (which is a separate problem).

For every function or script that takes a command as a parameter, the name can’t be enough to tell you what it does; you have to go look up what the enumerated acceptable values are. That’s time I could spend sipping margaritas on the beach. By splitting UUID ( type ) into UUID.v1 and UUID.v4, both options are now visible in the palette of available commands built in to FileMaker.

One possible exception to this might be commands that take boolean values. The meaning of a “suppressDialogs” parameter for a script is reasonably obvious.

What does everyone else think?

This is proposed as a best practice.