Script trigger for moving window

Let us show you how we can use our Window functions in combination with Schedule functions to trigger a script the a window moved in FileMaker. You may know there is a layout size changed trigger, so you notice when a window was resized by the user, but there is no script trigger for moving window built-in. Let's build one.

We build this example script, which starts with setting our parameters on which script to trigger with what parameter and in which file. Then we query the current window to get the reference number. This allows us to later even check the right window, when it is in the background. We query the current position and store it in the tag for the window. The tag is a value you can store to an object in MBS FileMaker Plugin and it allows you to store something related to the window. If you store multiple values, you may use JSON instead. We then schedule a dummy expression to get the reference number for the schedule, since we include it in the expression. Now we can build the expression and we look into that further below. Important is that we replace place holders with the actual values. Then we can put back the expression for the schedule and remember the schedule reference in case you like to stop it by script. Take a look here:

```
# our settings
Set Variable [ $FileName ; Value: Get(FileName) ]
Set Variable [ $ScriptName ; Value: "Moved" ]
Set Variable [ $ScriptParameter ; Value: "" ]
#
# now build it
Set Variable [ $currentWindow ; Value: MBS( "Window.Current" ) ]
Set Variable [ $WindowPos ; Value: MBS( "Window.GetLeft"; $currentWindow ) &
"/" & MBS( "Window.GetTop"; $currentWindow ) ]
# we store the position in the window tag right with the plugin
Set Variable [ $r ; Value: MBS( "Window.SetTag"; $currentWindow;
$WindowPos ) ]
# init schedule
Set Variable [ $ScheduleRef ; Value: MBS( "Schedule.EvaluateAfterDelay"; 1 /*
every second */; "1" /* dummy expression */; ""; ""; 1 /* repeat after every second */
) ]
Set Variable [ $expression ; Value: "Let ( [

currentWindow = $$$currentWindow;
scheduleRef = $$$scheduleRef;
FileName = $$$FileName;
ScriptName = $$$ScriptName;
ScriptParameter = $$$ScriptParameter;
oldWindowPos = MBS( \"Window.GetTag\"; currentWindow );
closed = MBS(\"IsError\");
newWindowPos = MBS( \"Window.GetLeft\"; currentWindow ) & \"/\" &
MBS( \"Window.GetTop\"; currentWindow );

r = If(oldWindowPos ≠ newWindowPos;
		MBS( \"Window.SetTag\"; currentWindow; newWindowPos ) &
		MBS( \"FM.RunScript\"; FileName; ScriptName; ScriptParameter );
		0);
r = If(closed;
		MBS( \"Schedule.Release\"; scheduleRef );
		0)

]; \"\")" ]
Set Variable [ $expression ; Value: Substitute($expression; "$$$ScriptParameter";
Quote($ScriptParameter)) ]
Set Variable [ $expression ; Value: Substitute($expression; "$$$FileName";
Quote($FileName)) ]
Set Variable [ $expression ; Value: Substitute($expression; "$$$ScriptName";
Quote($ScriptName)) ]
Set Variable [ $expression ; Value: Substitute($expression; "$$$scheduleRef";
$scheduleRef) ]
Set Variable [ $expression ; Value: Substitute($expression; "$$$currentWindow";
$currentWindow) ]
Set Variable [ $r ; Value: MBS( "Schedule.SetEvaluate"; $ScheduleRef;
$Expression ) ]
Set Variable [ $$MovedSchedule ; Value: $ScheduleRef ]
```

Let's take a look on the expression below without the extra escaping for the quotes and some coloring:

```
Let ( [

currentWindow = $$$currentWindow;
scheduleRef = $$$scheduleRef;
FileName = $$$FileName;
ScriptName = $$$ScriptName;
ScriptParameter = $$$ScriptParameter;

oldWindowPos = MBS( "Window.GetTag"; currentWindow );
closed = MBS("IsError");
newWindowPos = MBS( "Window.GetLeft"; currentWindow ) & "/" &
MBS( "Window.GetTop"; currentWindow );

r = If(oldWindowPos ≠ newWindowPos;
                MBS( "Window.SetTag"; currentWindow; newWindowPos ) &
                MBS( "FM.RunScript"; FileName; ScriptName; ScriptParameter );
                0);
r = If(closed;
                MBS( "Schedule.Release"; scheduleRef );
                0)

]; "")
```

In the Let statement we first have all values needed in local variables. Then we query the window tag to get the last position. If that fails and we get an error, the window probably got closed and we like to later release the schedule. If we got the position, we query the new position. Then if those don't match, we first store the new position and trigger a script.

Please be aware, that this solution should work in a database with multiple windows since each window gets their own schedule entry. But note that the user may quickly move the window and cause multiple script triggers, which then may get queued if FileMaker is busy.

Let us know if it works.