

Using PerformScript in a custom WebViewer in FileMaker

When you make a custom WebViewer with [MBS FileMaker Plugin](#), you may miss the callbacks with FileMaker.PerformScript there. But we got a way to get them into our WebViewer, too. Let us explain:

First we create a custom WebViewer using [WebView.CreateWithControl](#) function. Passing zero here for the current window and the name of the control for the placeholder on the layout. Then we request WebKit 2 version (macOS only!), so we can use message handlers later.

```
Set Variable [ $$web ; Value: MBS("WebView.CreateWithControl"; 0; "placeholder"; 2) ]
```

Next we install the message handler to receive the script trigger. We use FileMaker as name for the handler. This name must be unique and FileMaker internally uses "fm", so you can install our handler also on a regular WebViewer.

```
Set Variable [ $r ; Value: MBS( "WebView.AddScriptMessageHandler"; $$web; "FileMaker" ) ]
```

Next we evaluate a JavaScript to install it.

```
Set Variable [ $r ; Value: MBS( "WebView.Evaluate"; $$web; Own WebView::JavaScript ) ]
```

The JavaScript looks like this:

```
if (window.FileMaker == null)
{
    window.FileMaker = {};

    window.FileMaker.PerformScriptWithOption = function
(scriptname, parameter, option) {
    if (option == null) {
        option = "0";
    }
    if (parameter == null) {
        parameter = "";
    }

    var message = {};
    message.scriptname = scriptname;
    message.parameter = parameter;
    message.filename = 'Custom WebView.fmp12';
```

```
webkit.messageHandlers.FileMaker.postMessage(message);
    }
    window.FileMaker.PerformScript = function (scriptname,
parameter) {

window.FileMaker.PerformScriptWithOptions(scriptname,
parameter, null);
    }
    v = "okay";
}
else
{
    v = "already installed";
}
}
```

As you see, we install a FileMaker namespace, if it doesn't exist already. Then we define the functions to encode the parameters in an object and pass it to the message handler. The plugin will receive the request and trigger the script. As you see we hard code the name of the file into the JavaScript, but this could also be set dynamically with a JavaScript call later. And we put an answer into a variable, so we get that back from the evaluate call.

Now you can use the trigger in your JavaScript, e.g.

```
FileMaker.PerformScript('test', 'hello');
```

Enjoy custom web viewers with JavaScript callback and adjust this example as needed!