FrameMaker ExtendScript Ideas

By Rick Quatro, President of Carmen Publishing Inc.
rick@frameexpert.com 585-366-4017
http://www.frameexpert.com http://frameautomation.com

What is FrameMaker ExtendScript?

- ExtendScript is a scripting environment built into FrameMaker. It gives you the ability to automate FrameMaker tasks.
- The concept is similar to Visual Basic for Applications (VBA) that is built into Microsoft Office products like Word and Excel.
- ExtendScript is Adobe's version of the JavaScript scripting language. It is extended to include features that go beyond the core JavaScript language.
- ExtendScript is used for automation in other Adobe products, such as RoboHelp, InDesign, Photoshop, Illustrator, and Acrobat.

Why Use ExtendScript?

Scripting is all about automation and making things go faster. There are three major reasons to use automation:

- Automate tedious, repetitive tasks.
- Extend FrameMaker functionality.
- Make impractical tasks practical.

Below is a list of several sample scripts and why they were written. For each script I list the estimated or actual cost to the client and some possible ways that they could be enhanced. While some of these may only be of interest to a narrow audience, they illustrate the broad range of things that can be accomplished with FrameMaker ExtendScript.

ApplyBlankBeforeSection.jsx

Workflows vary so much that I am sometimes surprised at what people find important. FrameMaker has great book features, but one particular client likes to use multiple sections in a single FrameMaker document. Furthermore, they wanted each section to start on a right hand page and any blank pages before sections to be marked with "Intentionally Blank".

It is useful to see how this would be done by hand. First, our Section paragraph format would be set to start at the **Top of Right Page**. Next, we would have a custom master page with the "Intentionally Blank" text on it. To apply this to blank pages at the end of sections, we can use **Format > Page Layout > Master Page Usage**.

Doing it by hand is not awful, but it is tedious and prone to mistakes, so this is a good candidate for a script.

Estimated client cost: $250.

Possible enhancements: The script could be triggered automatically when the document is opened or before it is printed, or at any of a number of other FrameMaker events. Also, the section paragraph format and blank master page could be specified in a settings file instead of being hardcoded into the script.

ApplySubheadRows.jsx

FrameMaker tables are great for organizing all kinds of information. Two useful features are table headings that repeat and table continuation variables. But in certain large tables, you may want to use subhead rows to delimit sections of data within the table. We might think of these as "subhead rows" and we want a way to easily format them differently from the other rows in the table.

This table gives a natural indication of the subhead rows: the left cell of the row has content, but the rest of the cells don't. This gives the script a way to easily find the rows that need special formatting.

This script uses a second script to "reset" the subhead rows whenever it is run. This reset script can be run by itself to restore the tables to their original state.

Estimated client cost: $600.

Possible enhancement: A way to set the subhead rows' specifications in a settings file separate from the script.

DocumentNavigator.jsx

This is an example of adding a productivity enhancement to FrameMaker. We make a simple "Navigator" script for helping us navigate longer FrameMaker documents. In structured FrameMaker you can use the Structure View as a navigation aid. Our script will use a floating dialog box with a list of the document's headings.

To display the headings in the active document, you click the **Refresh** button. I was going to have it do this automatically whenever you switch active documents, but I decided that it was useful to have it done manually. That allows you to switch to another document temporarily, and then use the Navigator to return to the original document. There is also a back button (**<<**) so you can retrace your steps through the document.

Estimated client cost: I wrote this one for myself, but since it has general appeal, it would probably sell for $25-40.

Required enhancement: A way to prompt for the headings that are included in the script. Right now they are hard-coded in the script.

ConfirmElementDeletion.jsx

Here is another script that I never would have thought of. A client has xml that is largely machine-generated and then opened with FrameMaker. The user may make minor edits in FrameMaker but they want to ensure that no elements are accidently deleted from the document. So before an element can be deleted, the user is presented with a Yes/No prompt to make sure they want to do it.

This is an example of a "notification" script that is triggered automatically by a FrameMaker event. These kind of scripts sit in memory and wait for a particular event to happen in FrameMaker before springing into action. There are many FrameMaker events that can be monitored by a script. In this case, the script waits for the user to attempt to delete an element and then displays a confirmation dialog box. If the user clicks No, the event is cancelled and the element is not deleted.

Estimated client cost: $350.

Possible enhancement: A toggle command so that the script could be easily turned on or off.

SaveAsMif_Event.jsx

This is another example of a simple notification script that automatically saves a MIF version of any FrameMaker document you save. This is useful if you work with others that have various versions of FrameMaker.

Estimated client cost: $100.

Possible enhancement: A toggle command so that the script could be easily turned on or off.

SetVariableDefinitions.jsx

This script is useful if you have a lot of user variables that you need to set for a particular project. Using the FrameMaker interface for defining variables can be a bit awkward, so this script allows you to set them in a simple, two-column FrameMaker table.

There are three things that make this particularly useful: First, if any of the variable formats don't exist in the document, the script will create them for you. Second, you can run the script on a document or book. When you run it on a book, it means that you can skip the normal step of importing formats from one document to the rest. Third, because you are not importing formats, the script will leave your system variables alone.

Estimated client cost: $300.

SetDocumentData.jsx

This script is a variation on the previous script for setting variable definitions in a document. Instead of being driven by a FrameMaker table, it is driven by a simple XML file. The user is prompted with a dialog box that lists the variable formats and the default values for definitions.

This script is interesting because the number of fields in the dialog box is controlled by the number of <group> elements in the configuration file. That makes it easy to customize for different projects.

Estimated client cost: $300.

SetAttributeValues.jsx

This script is for setting attribute values on a selected element in a structured FrameMaker document. If you have used the built-in Attribute pod, you know that it is difficult to set values because you can't tab from attribute to attribute.

This script is context-sensitive so it will only show attributes that are available on the selected element.

Estimated client cost: $400.

Possible enhancement: A context-menu command with a shortcut key to make it easy to invoke.

BooksToPdf.jsx

Most of the previous scripts have been focused on smaller tasks. But you can also automate larger workflows as well. This script outputs a set of PDFs from one or more books, showing and hiding conditions as appropriate. The script is driven by a simple XML file that indicates each book's path, the conditions which should be shown, and the output path for the PDF.

The sample book we are using is 544 pages, but all three of the PDFs are created in around 5 minutes.

Estimated client cost: $1,200.

Possible enhancements: A useful function would be the ability to set variable definitions, based on new elements added to the Settings.xml file. The script could be reconfigured so that it could be controlled by a FrameMaker settings document or text file instead of an xml file.

Conclusion

You can see how useful ExtendScript is for FrameMaker users. Even simple scripts can continue to save time every time they are used.

See http://frameautomation.com/?p=416 to learn how to receive the sample scripts in exchange for a small donation to the Ride for Roswell to benefit the Roswell Park Cancer Institute. My son Jason and I will be riding 102 miles on June 27, 2015 to raise money for cancer research.