The Global Object in Acrobat JavaScript – Samples

For more details see the article "Global JavaScript Object in Acrobat JavaScript" at www.acrobatusers.com.

The examples below demonstrate various techniques for sharing data between different scripting contexts within the Acrobat Scripting environment using the `global` object. All samples, unless noted otherwise, will operate in all versions of Acrobat and Adobe Reader.

A very important point to keep in mind when running these samples in Acrobat 8 or later is that a global variable belongs to the document context where it was created. The variable is not available to other documents and will throw an exception if accessed from another context. Another document can take ownership of a persistent global variable immediately after Acrobat is started up. Or if the variable is reset from an application context, such as a trusted function or the console window.

Sample #1: **Basic** `global` object. The Save button below stores the text entered into the text field as a member of the `global` object. The restore button retrieves that text and puts it back in the same text field.

To test this simple example:

1. Enter text into the field
2. Press Save
3. Delete the text in the field
4. Press Restore

Test Data: Clear

Save Restore

Try different variations of this process to develop an understanding of how the global object works within the different versions and variations of Acrobat/Reader. Close the file, reopen it, and then press Restore. Save the file to a different file name and then try to restore. Close then restart Acrobat, then try to restore. Try this in Reader and different versions of Acrobat. Also try logging into the computer as a different user and then restore the data. Make sure to have the Console Window open at all times for detecting errors as they happen.

Note that in Acrobat 8 and later, the data cannot be accessed by a document that is different from the one that saved it (this was the point of saving to a different name). Also note that the data is not available when Acrobat is closed and re-opened and it is not visible to another user logged into the same system and viewing the same document. Data stored in the global object is only visible to scripts on this PDF, and to the current user.

Sample #2: Persistence of entries in the `global` object. This example is essentially the same as the one above except the saved value is set to Persistent. This means Acrobat saves it to an external file so the value is restored to the `global` object when Acrobat is restarted. Try this example with the same variations that were used in Sample #1. In this sample the data will be restored when Acrobat is closed and reopened. This is the only difference in behavior between the two samples.

Test Data: Clear Reset Global

Save Restore

Sample #3: Practical example of persistence. Scripts have been added to this PDF to restore the last view when the document is reopened. The save script is placed in the WillClose event, and the restore script is in a document level script named RestoreView. The view is saved by converting the doc.viewState object into a string and attaching it to the global object. The view is restored by converting the saved string back into an object. The viewState document property was added to Acrobat in version 7.05.

Sample #4: This is a more **complex persistence** example. It demonstrates an auto-populate feature for a form. The user selects from a list of names, which then fill in a set of fields with the associated data. In this case, the data is for selecting a department within a company, but it could also be for selecting a customer, or any other kind of data used repeatedly on the same form.

The downside to storing this kind of data in a global object is that it cannot be changed from an external source (such as an excel file) and the way it's implemented here the data is only available to a single document. This data could be made available for use on several documents if the scripts that use the global object were moved to a trusted function in a Folder Level script.

The department info fields below are automatically filled from a selection on the "Name" pull down. Since the data is complex, the working version is stored in a temporary object on the global object. However, objects cannot be persisted in the global object. So, whenever the data set is modified it is converted into a string. This string is the value that is persisted. A Document Level script is used to initialize the working version of the data and the dropdown list of departments.

The fields can also be filled out manually. Then the Add/Save button is pressed to either modify an existing entry or add a new entry to the data set.

Department Name:		Add/Modify	Reset Sample
Contact Name:	Email:		

Sample #5: Trusted functions for **auto-populate** example. This is an exact copy of Sample #4 except that the fields below, and the "Sample4" document script, use trusted functions to access the global object. This allows the auto-populate feature to be used on multiple documents. The trusted functions are defined in a JavaScript file Attached to this form. In the File attachments pane you'll find a file named "Sample5.js.txt". Extract this file into the Acrobat JavaScript folder and remove the ".txt" extension. Restart Acrobat to activate the file and use the fields below. More detailed instructions can be found here.

Department Name:		Add/Modify	Reset Sample
Contact Name:	Email:		

Sample #6: **Auto-populate Automation** Script. If you are going to use a Folder Level script, why not just go all the way. In this example, a full automation script is used to provide the exact same auto-populate feature used in Samples #4 and #5. The script is in the "Sample6.js.txt" file attachment. Extract and rename this file as explained in Sample #5. The script adds a tool button to Acrobat (same as icon on the button below) which displays a popup menu with the department name options. The first options adds or modifies the list from the fields on the current document.

Department Name:		Data Auto-Populate
Contact Name:	Email:	