Creating Document-level JavaScripts

- Using Acrobat (Windows) v4.x -

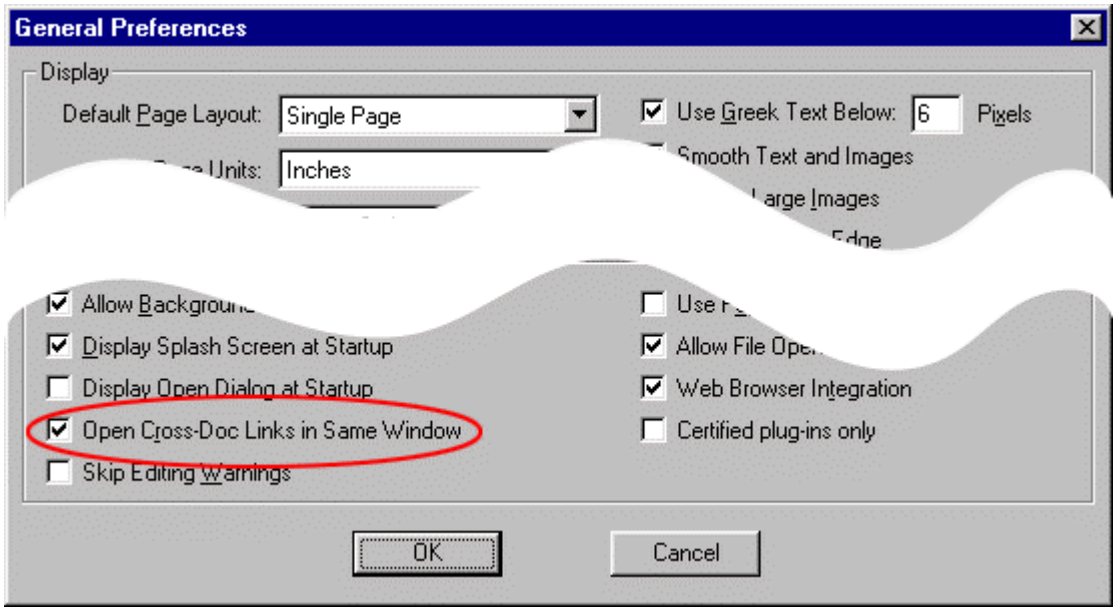
What is this demo for?

This document is an illustrated walk-through for adding document-level JavaScripts for your PDF. For more background information on JavaScript see the section "Working with JavaScript Actions" at pages 422-424 of the Acrobat Guide PDF. You can access the latter via the menu 'Help->Acrobat Guide'' (in the full Acrobat app). In addition, you will find more information about Acrobat JavaScript in the PDF via the menu 'Help->Forms JavaScript Guide', also in the full Acrobat app.

One useful application of document-level JavaScripts is to change Acrobat & Reader preferences at run-time. Normally achieving such changes would require the application to be closed and re-started (especially inconvenient if running Reader from CD).

In this tutorial, as a working example you will add a document level JavaScript to turn off the General Preference option to "Open Cross-Doc Links in Same Window" from its default 'on' state. The property is listed at page 11 of the Forms JavaScript Guide PDF. Note that in the v4.0 copy of that PDF, the property is incorrectly listed as type 'number' when it is a 'boolean' field; this is corrected in the PDF shipped with v4.05.

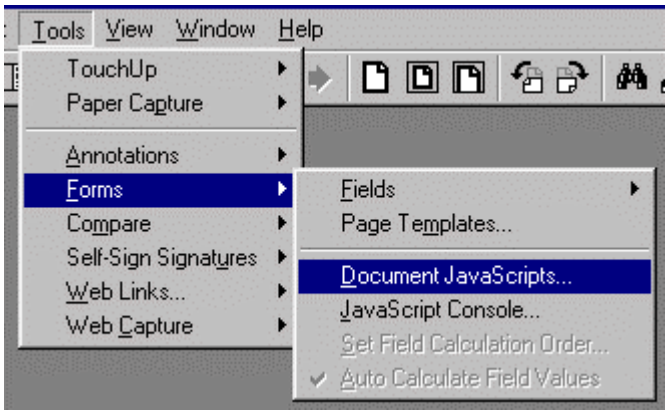
If you open Acrobat's General Preferences (File->Preferences->General) and have default option setting you should see the item circled below is checked. If it is not, and you wish to follow this tutorial, uncheck the option, close Acrobat and re-start it.



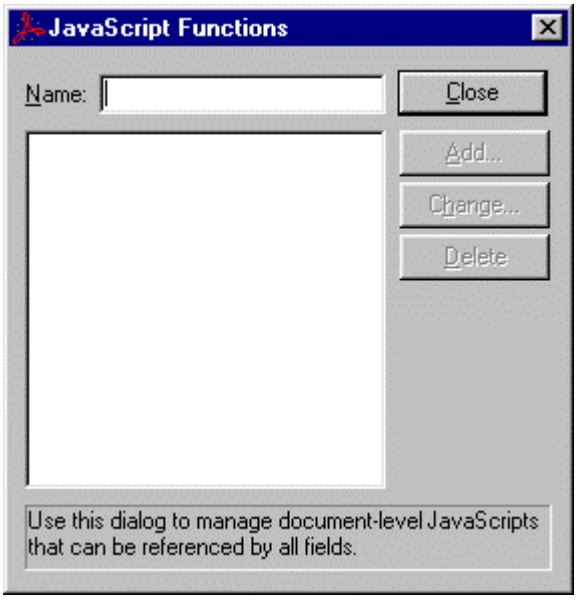
Before starting the tutorial you should either create a new PDF or use an existing one you don't mind altering. To be able to test the demo works, either add a link from your PDF to another PDF on your hard drive or have ready to open an existing PDF with a cross-doc PDF link in it. Lastly, make sure the PDF you will be using for the demo is the only PDF open in Acrobat - close other PDF files if necessary.

Let's get started…

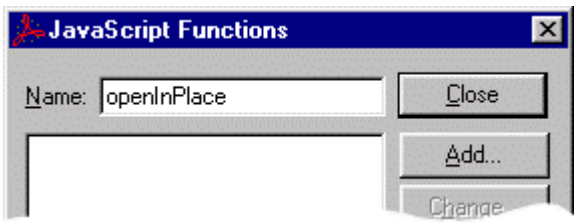
1. To add a document JavaScript, select 'Tools->Forms->Document JavaScripts…' as shown below:



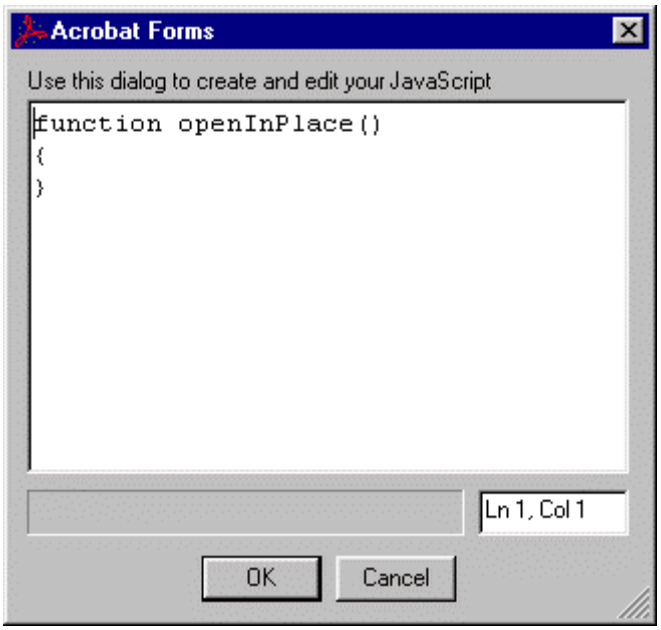
You will now see the 'JavaScript Functions' dialog as shown below. (If you are using a PDF that already has a document-level JavaScript in it, you may see an entry in the main list panel - bottom left below).



2. Call your new script 'openInPlace'' to reflect the JavaScript property you will alter. Type your script's name into the text box at the top of the dialog. Note the 'Add…' button is now enabled.



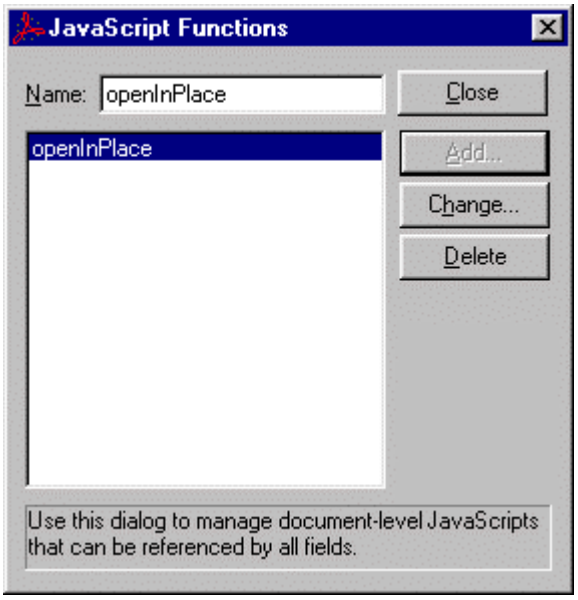
Click the ‘Add…’ button and the ‘Acrobat Forms’ dialog is displayed as below:



3. Note that a default empty function is written into the script box, whose name is the same as you used for the script’s name in the last dialog. For this demo, you don’t need the default function script, so select it all and delete it. Now enter the code shown below, taking care to copy the syntax.

If you are new to JavaScript bear in mind that the language is case-sensitive and a statement separator (;) may be required, as here, to terminate a line. If your script doesn’t work when finished, most likely you’ve used ‘False’ instead of ‘false’ or omitted the semicolon at the end of the line.

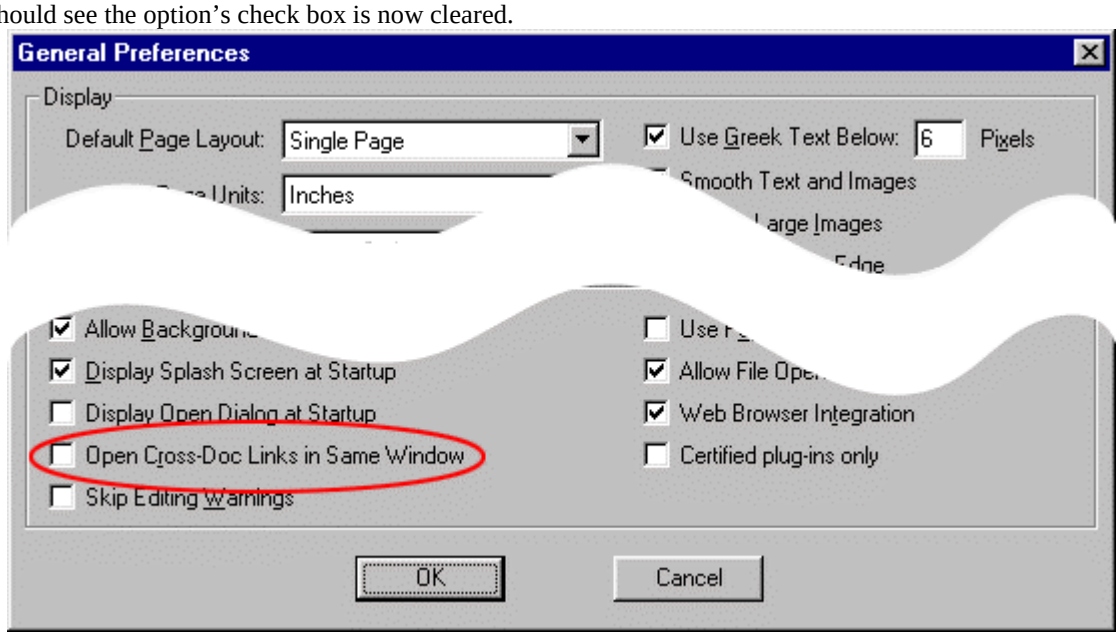
4. When you're done, click the 'OK' button. You will now see the original 'JavaScript Functions' dialog, but notice your new script is included in the script list (lower panel) and the 'Change…' button is now enabled, allowing you access to your script to edit it.



5. Now click the 'Close' button. Although Acrobat doesn't warn you of the fact, your new script is now run. If there is an error in the script you've created, Acrobat will display the Java Console window and (may) give you some indication of the error - though if you are new to JavaScript you'll probably still need to figure out what the error is. If you do get an error, close the console window and follow step #1 to get back to the dialog shown above. Then click the 'Change…' button to edit your script.

6. Assuming you've made no errors, save your PDF. Your script is now saved as part of the PDF file just as it is when you attach a simple script to a form button action. Is a separate .JS script file created? No, not in this demo - the script is incorporated in the PDF file.

7. Now you can prove that your script has worked by re-opening the General Preferences dialog. You should see the option's check box is now cleared.



8. This alone is not proof, as you would see the same dialog state if you had just deselected the option manually via this dialog. Now you'll prove your script actually made this change whilst the app is running. Without closing Acrobat, click your cross-doc link (or open your PDF with the link and click it) and you should now see more than one PDF listed under the Window menu (your listed filenames will differ).

By contrast, with the cross-doc preference option set in the default (as installed) condition the first PDF would have been closed as the second was opened and thus only one file would be listed.

Note: you are not restricted to only having one PDF open whilst adding scripts to your PDFs. The instruction at the start of the tutorial above was only included to make sure you could prove your test script had worked.

9. You've now proved your script. Next time you open the PDF, the script you've just added - and any other document-level scripts - will be run as soon as the PDF loads. Try manually resetting the cross-doc preference to the default 'on' condition and close Acrobat. Re-open the app and check the option is now on again. Now open your demo PDF and re-check the preference dialog. Notice that the cross-doc option has been changed automatically to the 'off' condition.

Well done! You've successfully created a working document level JavaScript.

Anything else?

Certainly. In this tutorial you've simply reset Acrobat's General Preference options. JavaScript is capable of doing much more than you've achieved here. For instance, if you are using Acrobat's forms functionality you may want to set up things such as variables before the user sees the form; using a document-level script can do this for you. As the scripts are run as the file loads any actions/events you have scripted are executed before the user sees your PDF.

To make full use of scripting at this skill level you'll need to learn (some) JavaScript and to read up on how the JavaScript built into Acrobat differs from that supported by web browsers. It is very important to understand that Acrobat's implementation of JavaScript is slightly more restricted than the web form. The Document Object model found in a Web browser has not been translated for Acrobat. Acrobat has the **core**

JavaScript methods and Adobe have added specific JavaScript methods/features for Acrobat. Thus JavaScript in a Browser is only similar in syntax/language structure to Acrobat's version - **not** in functionality/features - they share no functionality. Importantly, though you can run external JavaScripts (.JS files) you have little or no ability to access things outside the PDF environment. There are perfectly good reasons for this but explanation lies outside the scope of this document.

Besides the 'Forms JavaScript Guide', Acrobat installs another PDF about JavaScript - 'JavaScript Language Specification''. This latter PDF is not linked to from Acrobat's Help menu but assuming you've installed Acrobat in the default location, you can find it at:

"C:\Program Files\Adobe\Acrobat 4.0\Help\JSSpec.pdf".

The document is not exactly easy reading and the automatically generated bookmarks are a bit of a mess but there is a wealth of information. As far as I am aware at time of writing (May 2000) there are no JavaScript books based around Acrobat's implementation of JavaScript, and if you are using a general web JavaScript book do be aware of the differences I've mentioned above.

It is worth checking the PlanetPDF site, and its CodeCuts section, as there are a number of useful articles, scripts and links. If you get bogged down there are good, friendly user-to-user forums at AcroBuddies (part of PlanetPDF) and Adobe.

A little knowledge can be a dangerous thing…

Just because you can now change options etc. at runtime do please pause and consider whether such scripting is appropriate. The example used in this demo reflects the many people with CD-based Reader projects who wish to have each link on their project open separately who have asked how to achieve this.

Be aware that whether run from CD or hard drive, all installations of Acrobat (or reader) share a common set of preferences. Bear in mind that if your CD-based project alters a preference, then users with the app on their hard drive will see this reflected when they run the app from the hard drive. So, if you change the application's set-up for to use your PDF(s) unless you reverse the changes when you're done then they will affect use of the user's other PDFs.

There is no 'on exit' type of event in Acrobat's JavaScript, so even if you do cache an existing user preference there is no easy way to ensure you reset to that value on leaving your project. Therefore, be considerate of your users. If you need to do some 'customisation' for your PDFs, think about things like showing a message to alert people to the changes you've made (JavaScript can do this – use app.alert). Alternatively, you could show a warning 'on screen' in the text of you PDF, or if using annotations you can put the information there. The key thing is not to create a problem for your user by the way you solve your own programming difficulties. Do as you would be done by.

That's it!

I hope this has been useful. Compared to something like HTML, JavaScript is harder to get to grips with for the non-programmer. Remember that ideally you should create as much of your PDF, including links and such, in your original source application e.g. word processor, desktop publishing, etc. Think about the structure of your project and the documents within it at an early stage. I hope this tutorial has helped you. If it has please return the favour by using your own expertise to help someone else.

Regards,

Mark Anderson
Portsmouth, UK
mark@yeardley.demon.co.uk
www.yeardley.demon.co.uk

Version: v1.01