INDESIGN magazine

Making an Impact with Motion

Plus:

- Word Play: Type on a Path
- Javascript for InDesign
- Packaging Design

APRIL 2020 / ISSUE #132

JAVASCRIPT FOR INDESIGN

Book Excerpt

BY **Peter Kahrel**

ANYONE WHO WORKS REGULARLY IN InDesign has compiled a wish list of features they'd like to see added to the program. But perhaps the greatest feature of them all already exists—the ability to control InDesign with scripts. That means you—yes, you!—can create the features you want. But first you need to learn how to tell InDesign what you want in a language that it understands, like JavaScript. And the best way to learn it for the purpose of writing InDesign scripts is to read Peter Kahrel's book, *JavaScript for InDesign*. In the following pages, we present an excerpt to get you started, along with an exclusive set of simple one-liner scripts that Peter came up with to give you a quick taste of success.

Introduction

Two things stand between the would-be scripter and an InDesign JavaScript: InDesign's object model and JavaScript. Though both are complex, once a few hurdles are overcome, anyone can start writing scripts fairly quickly. This guide hopes to show that numerous tedious tasks in InDesign can be automated with very simple scripts of sometimes just one or two lines. These simple scripts can pave the way to more complicated scripts. What you need most of all is determination.

To give just one short example, imagine this task: you have a document with dozens of pages, and each page contains one or more text frames and one or more graphics. All these page items are on the same layer, and you decide that the document would be much easier to handle if the graphics were on a separate layer. The task, then, consists of two steps: create a new layer and move all graphics to this layer. Can you imagine doing this manually? Well, the following two-line script (FIGURE 1) does it for you in the blink of an eye.

The first line creates a new layer with the name 'pictures', the second line moves all graphics to that layer. You ask, 'But how do I know that layers are added like that', and 'How do I know that a graphic is in an object 'rectangle'?' Read on—the purpose of this guide is to show how to discover this.

FIGURE 1

```
myLayer = app.activeDocument.layers.add ({name: 'pictures'});
app.activeDocument.rectangles.everyItem().itemLayer = myLayer;
```

Another aim is to show that there are many very tedious and labour-intensive tasks in InDesign which can be solved with surprisingly simple scripts.

This book is intended for people who know InDesign fairly well but do not necessarily know much about scripting/programming. Knowledge of InDesign is necessary; after all, if you don't know InDesign there's not much point trying to script it. Knowledge of a programming language is not necessary (though it helps, of course). I believe that anyone can learn how to write scripts up to a certain level. You don't have to be a mathematician in order to acquire some scripting skills. In fact, many excellent script writers are graphic designers or typesetters or have different backgrounds, as can be seen, for instance, in Adobe's scripting forum. Creating JavaScripts for InDesign is not about computer science: it is about making something work in InDesign, and often that's pretty simple.

In essence, this guide contains three parts. In the first part we deal with Adobe's script editor and the object model, and we show how the object model can be explored using the script editor. The second part outlines the basics of JavaScript. This is not a full JavaScript course but deals with the main elements of the language and gives some examples to get you started. And in the third part we write several scripts. They all essentially

handle text. The first few scripts deal with a number of basic text-scripting techniques. This is followed by some scripts that go into various aspects of find and change.

I first show how this can be scripted merely to automate InDesign's Find/Change dialog, then move on to show how Find can be used to script a flexible kerning editor. We then take a close look at tables. Though InDesign's tables are quite powerful, some features are missing and we'll show how these can be scripted. In the last section we turn to some aspects of text frames.

All scripts in this book have been tried and tested, and should work as advertised in modern InDesign versions (they should in fact work in versions dating back to CS3). Nevertheless, before trying any script, even those that seem simple and innocuous, always make a copy of your document or try the script on a mock document. Never try out a script on an unsaved production document: InDesign's undo works very well, but you don't want to put yourself at its mercy.

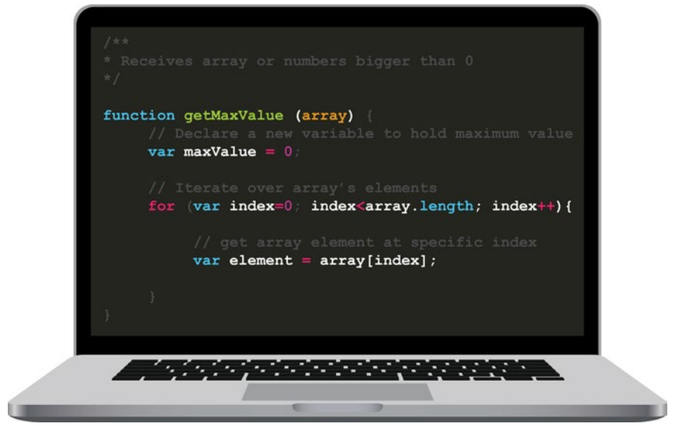
JavaScript/ExtendScript

InDesign and other scriptable Adobe applications use a variant of JavaScript called 'ExtendScript'. It's called ExtendScript because it's JavaScript extended with file-handling and XML-processing. ExtendScript is a by now quite old version of JavaScript, lacking various interesting features that were added in recent years, but it's more than capable. InDesign JavaScript scripts are cross-platform: they can be used on both the Mac and Windows.

The Script Editor

JavaScript files are plain text files. Though they can be created in any text editor, it's advisable to use Adobe's script editor. It's called the ExtendScript Toolkit (ESTK) and it can be downloaded from the Creative Cloud app. The advantage, as we shall see shortly, is that you can run a script straight from the ESTK: there's no need first to save the script, then to run it from within InDesign.

However, the ESTK is beset by problems on the Mac. There have been various problems in recent years which could make working with it problematic. In 2019, Apple's operating-system upgrade, Catalina, dropped



support for 32-bit applications. And since the ESTK is a 32-bit application it joined the ranks of programs that can no longer be used on the Mac. This led many Mac script writers to abandon the ESTK entirely and to rely on text editors.

Things look better on the Windows side. Even so, the ESTK is an aging application that hasn't seen any development for many years. It still works remarkably well for such an old application but its age is showing.

SIMPLE SCRIPTS TO GET YOU STARTED

Experimenting with these short scripts will give you a boost of confidence as you start to explore the world of scripting InDesign.

1. How many paragraph styles are there in this document?

```
alert (app.activeDocument.allParagraphStyles.length);
```

To create variants to show the number of other kinds of styles, replace allParagraphStyles with allCharacterStyles, allObjectStyles, and so on.

You can, in fact, count a lot of things, including tables:

```
alert (app.activeDocument.stories.everyItem().tables.
length);
```

2. Suppose you have a book with lots of documents, and all those documents currently start at page 1. To set all documents to continue numbering (i.e. select Automatic Page Numbering in the Numbering & Sections Options dialog box), open them all, run this script, and then close them all:

```
app.documents.everyItem().sections[0].continueNumbering
= true;
```

3. Close all open documents without saving:

```
app.documents.everyItem().close (SaveOptions.NO);
```

4. Save all open documents, then close them:

```
app.documents.everyItem().save();
```

```
app.documents.everyItem().close (SaveOptions.NO);
```

5. Delete an index and remove all topics:

```
app.documents[0].indexes[0].topics.everyItem().remove();
```

6. In InDesign's Find/Change window you can constrain the scope of a search or a replacement to the document, the selected story, a selection, or from the insertion point to the end of the story. You can't target tables and footnotes specifically, but with a script it's simple to do so. First, enter your find and change terms in the window, then run this script to make the replacement in tables only:

```
app.activeDocument.stories.everyItem().tables.
everyItem().changeGrep();
```

Run this one to make the replacement in footnotes only:

```
app.activeDocument.stories.everyItem().
footnotes.everyItem().texts[0].changeGrep();
```

To deal with the ESTK problems on the Mac, Adobe created an ExtendScript plug-in for Visual Studio Code. This is a popular Microsoft code editor for Windows and Mac, which can be downloaded for free here. The ExtendScript plug-in can be downloaded from here. The page with this download link includes directions on how to set up the plug-in and how to use it.

If you're on Windows I would advise you to use Adobe's ESTK script editor: it's simpler to install and use and it has its own object-model viewer (see details in the Introduction). On the Mac, do try the ESTK and if it works for you, stay with it while you can. Naturally, when the macOS upgrade makes working with the ESTK impossible you'll have to switch to Visual Studio.

The ExtendScript Toolkit

After installing the ESTK, open it. FIGURE 2 shows its default layout. I always have only the edit window and the console open, that leaves more space for script text.

Start a new file (Ctrl+N or **File > New JavaScript**). At the top left there's a dropdown with application names, in which ExtendScript Toolkit CC is the selected application every time you start the ESTK. To link the ESTK to InDesign, select your version from that dropdown (all Adobe applications that can be scripted with ExtendScript are there). Let's now write and run our first script.

Hello World!

Make sure InDesign is open. Then start a new script with a new editor window open, and enter `alert ('Hello world!');` (FIGURE 3). You can now run the script straight away against InDesign: choose **Debug > Run**

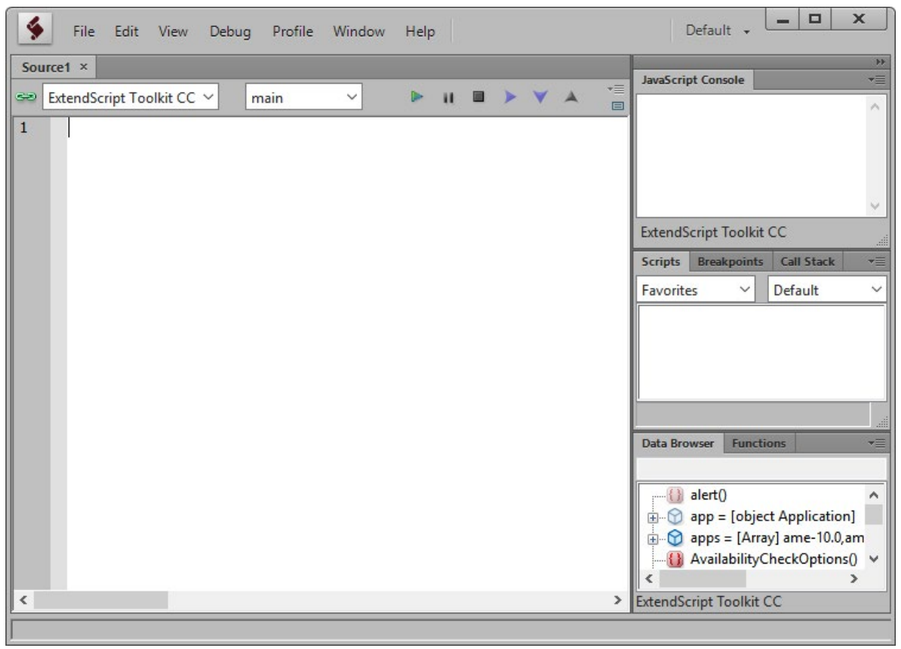


FIGURE 2

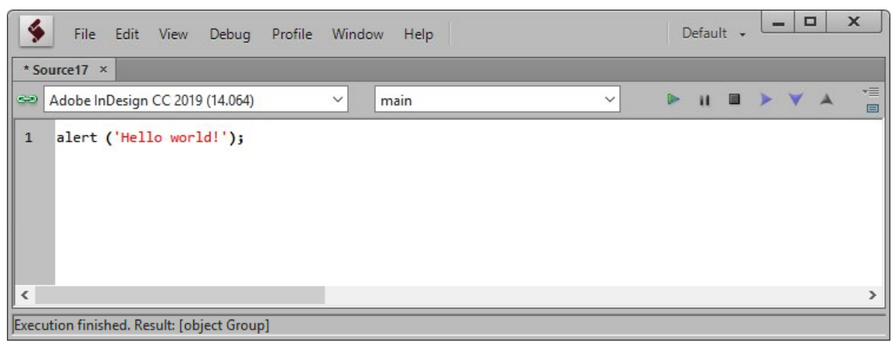


FIGURE 3

(or press F5) to run the script. A small window is displayed (FIGURE 4).

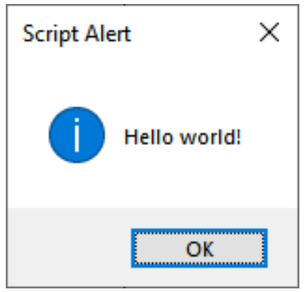


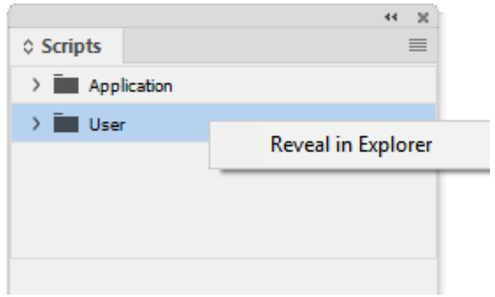
FIGURE 4

Saving scripts

Scripts should be saved in InDesign's Script Panel folder, which is in the Scripts folder. The easiest way to find that folder is as follows: open the Scripts panel in InDesign (**Window > Utilities > Scripts**), right-click the User folder, then select 'Reveal in Explorer' (Windows) or 'Reveal in Finder' (Mac). This opens your scripts folder (FIGURE 5).

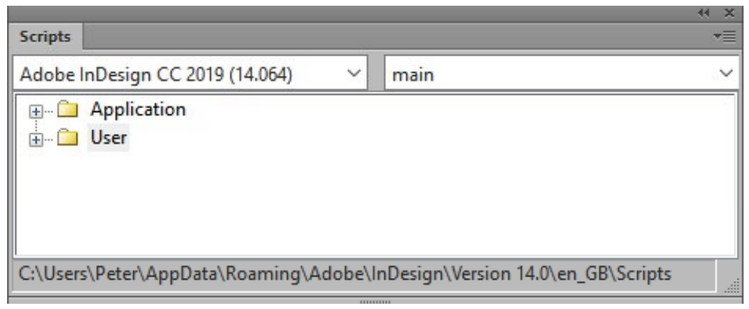
Another possibility: open the ESTK's Scripts panel (**Window > Scripts**), select your application in the dropdown, and click 'User'. The path to InDesign's Script folder is shown at the bottom of the window (FIGURE 6).

FIGURE 5



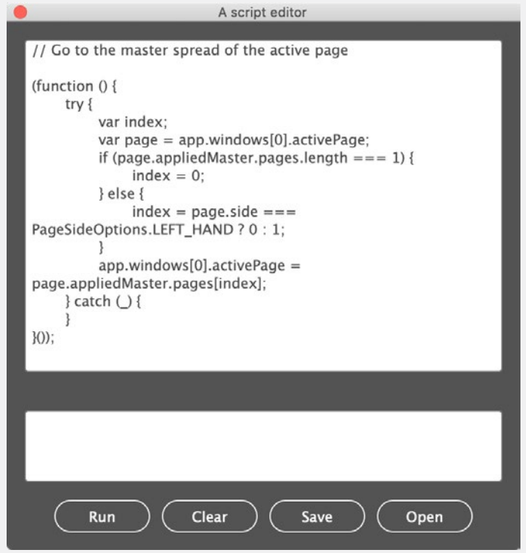
On the Mac, to avoid problems with read and write permissions, it's better to keep your scripts in the user folder.

FIGURE 6



ALTERNATIVE TO ESTK: A SIMPLE SCRIPT RUNNER

Intimidated by the ExtendScript Toolkit (ESTK)? **Download this script** and run it to open a simple script runner in a resizable window:



You can enter any lines of script code and click **Run** to execute the code. Clicking **Clear** clears the output panel at the bottom, **Save** lets you save the code into your scripts folder, and **Open** lets you open a script from your scripts folder.

For more details on the ESTK, go to **Help > JavaScript ToolsGuide CC**. The PDF file that opens contains a chapter on the ESTK with useful information.

Peter Kahrel *is a Scripting Engineer at Typefi Systems and the author of* GREP in InDesign*, as well as books on scripting and automating InDesign published by O'Reilly Media.*