Light Programming for Technical Writers

Russell Ward

tekom/tcworld 2012 - Wiesbaden, Germany
Agenda

- Establish why programming skills are important
- Dispel any lingering myths that you can not or should not have these skills
- Discuss basic concepts of programming and computer instruction
- Demonstrate specific examples relevant to writers
  - Basic Windows command line scripting
  - Basic web page scripting with HTML and JavaScript
  - Microsoft application macros, including Word, Excel, and Visio
  - Photoshop actions
  - FrameMaker customization with ExtendScript and the FDK
- Personal testimony and final pep talk
- Q&A
Speaker contact information

Russ Ward

- Senior Technical Writer at Spirent Communications in Germantown, MD.
  20324 Seneca Meadows Parkway
  Germantown, MD 20876
  301.444.2489
  russ.ward@spirent.com
  www.spirent.com

- Owner of West Street Consulting, a part-time enterprise specializing in Structured FrameMaker plugins and custom development.
  357 W. North St.
  Carlisle, PA 17013
  717.240.2989
  russ@weststreetconsulting.com
  www.weststreetconsulting.com
About the sample files

This tutorial will involve the demonstration of several sample files, which you can obtain at:

www.weststreetconsulting.com/misc/tekom_2012.zip

These files were designed specifically to give you some working examples of the concepts covered in the tutorial, such that you could use them as a starting point for further study.
Why modest development skills are important

- Primarily, because you have more power to make the computer do what you want. Otherwise, you are completely dependent upon someone else to create new functionality for you.

- Development skills allow you to tackle technical issues related to publishing and information management, things that technical writers frequently face that have little to do with authoring.

- Once you gain some abilities and actually create something that is useful, your enthusiasm for your career will take off.

- You will begin to respect yourself more, as will those who engineer software for a living, although they’ll never admit that they didn’t respect you in the first place.

- You may become a better writer, especially if you are in the field of software user assistance. Nothing will help you understand how software works better than the experience of creating it yourself.

- Bottom line... your value to the marketplace will increase, which corresponds to an increased likelihood of better pay, more leeway with negotiations, and survival of layoffs.
Why you can do it

- Primarily, because you are smart and technically astute. If you are not, you shouldn’t be a technical writer.

- You are in the business of providing clear and logical information, which is often procedural in nature. That is exactly what software development is, just using a different language. Lucky for you, if you can read this, you have a head start because all mainstream languages are based on English.

- You do not need to become a super code wizard to realize significant benefit from software development, especially when you are developing for basic publishing and information management needs. Generally, we are talking about tweaking and streamlining existing tools, not developing all-new tools.

- You may not need to spend a dime on tools or training. Many languages and development tools are free. Additionally, the internet is awash in free tutorials and technical support.

- It is not as hard as you think. It looks incredibly daunting at first and you may need some help to get started, but a little knowledge builds a little confidence, which then snowballs until you aren’t scared anymore.
Caveats

- You may have to invest some extra time, especially if you have a heavy writing workload already. Nothing good comes without sacrifice, though.

- If you do manage to acquire some basic skills, you might be tempted to walk around like you know as much as the engineers. You don’t and will eventually be reminded as such.

- You may decide that you would rather move into engineering and leave technical writing behind.

- Because so few technical writers obtain these skills, other writers may feel threatened by your expertise. This is not your fault, but it can create political problems that you didn’t expect.
**Mindset, planning, and technique**

- Software development is engineering, but so is effective writing. To build a document or a piece of software, you have to get the pieces together, then arrange them in the proper order so they make sense.

- Always start with a high-level perspective. Define the basic flow and embellish your plan from there. Just like writing a human procedure, try to work up a clear picture of the general steps that will be required before you start.

- Prioritize carefully while developing. Software development is notorious for scope creep. You may spend forever trying to get one little thing to work, then run out of time to make the most important things work.

- Remember the internet if you get stuck! If you can’t figure something out, chances are somebody before you was in the same boat and asked the question on some forum, where some guru gave the answer. There is no such thing as a dumb question to Google!
Just like information management, use every technique for code reuse that makes sense. If you’ve never developed code before, you may not know of such tools, but will eventually become familiar with some.

Just like writing, be prepared to make revisions, at all phases of the process.

Don’t expect that you have to know all the code syntax that the project will require, before you start. Nearly all development exercises are also times to learn new things about some language.

Finally, if your software will have end users other than yourself, listen to their feedback and especially DON’T GET MAD IF THEY CRITICIZE YOUR WORK! If they say something is junk, they might be right. Remember all those comments you’ve made about some user interface in the past? Welcome to the receiving end!
Windows command line and batch files

- Windows has a command line interface that supports several basic, but potentially useful, commands. These commands can be run directly from a console (command prompt) window or collected into a file and run as a batch.
  - A typical path to launch a console window is **Start > All Programs > Accessories > Command Prompt**
  - Batch files are in simple text format and normally named with a *.bat* extension, which most systems will recognize and execute as such

- Perhaps the most useful commands are related to file management, such as the copy commands that can be used to create backups.

- Advantages of command line / batch file scripting include:
  - Ease of use
  - No special editors or compilers necessary
  - Simplicity of syntax

- Disadvantages include:
  - Limited functionality
  - Rudimentary development and testing environment
Windows batch files - Copying files

- If you have many files to copy on a regular basis, there is no simpler option than a short batch file with one or more copy commands, for example:

  ```
  copy C:\my_documents\*.doc R:\some_backup_folder\*
  ```

- Batch files are easily scheduled for regular runs with the Windows scheduler. When given a *.bat extension, they automatically execute like a program.
Sample batch files - Demo

- **back_up_files.bat**
  
  Runs a simple file copy action, one that might be useful for making backups.

- **find_text_-_simple.bat**
  
  Searches the html directory in the sample files for the text “spirent” and writes the results to a file called `find_results.txt` in the same directory as the batch file.

- **find_text_-_with_choices.bat**
  
  Similar to `find_text_-_simple.bat`, except it allows you to customize the search string, through user input.

- **_visual_effects.bat**
  
  An extra that can be called from `back_up_files.bat` (or any other file) to do some visual effects.

**Note that on a normal system, batch files can be launched with a double-click from an explorer window!**
Web page scripting (client-side)

- Increasingly, browsers support a strong set of client-side scripting features; that is, code that is executed by the browser on the client machine.
- The most common language is JavaScript.
- Code is placed at strategic locations in the HTML markup using `<script>` elements, for example:
  ```html
  <script type="text/javascript">
  /* some code here */
  </script>
  ```
- Javascript can be placed in the HTML `<head>` where it is executed upon page load. Or, it can be placed in selective places within the `<body>` where it may execute on page load or not, depending on the setup.
Web page scripting strengths/weaknesses

Strengths include:

• Strong control over the presentation of a web page, making it very good for visual customization and dynamic interaction with the user.

• Generally free and easy to use, requiring no compilers or special software. An HTML file is in simple text format and the browser interprets script code automatically upon page load.

• Good support for technologies that are relevant to technical writers, such as CSS and XML/XSLT.

Weaknesses include:

• Because the browser is the processor, generally all output is in the form of browser display. Security limitations normally prevent any sort of heavy processing that involves the file system, especially with writing data.

• While you can develop with a simple text editor, the process can be error-prone especially with respect to debugging code.

• Different browsers require different syntax in some cases.
Sample HTML/JavaScript file - Demo

- dropdown.htm
  
  Contains a dropdown list and a button that change the font color of a test paragraph. It is intended to demonstrate basic techniques for responding to user actions.
Customizing Microsoft Office applications

- All primary Microsoft Office applications have a rich macro environment with which you can do virtually anything, such as:
  - Automated format and text manipulation
  - Database, CMS, and other external system integration
  - Custom GUI and interface development, including user controls embedded directly within documents

- Macros use the Microsoft Visual Basic for Applications language (VBA), a close relative to Visual Basic.

- Office applications include an embedded macro development environment which provides standard conveniences such as auto-completion, fast syntax checking, and highly-granular context-sensitive help.

- Office applications also include a macro “recorder,” which automatically writes rough VBA code as you perform manual tasks within the interface. The recorder receives criticism from “real” developers due to the rudimentary (and sometimes non-functional) code that it writes, but it can be handy for the novice to get started.
Customizing Microsoft Office (cont’d)

- The greatest strength of the Office macro environment is its sheer capability, especially when you are working within a Microsoft framework. It is virtually limitless in its potential for automation and application integration.

- For amateur developers, the scope of functionality and the depth of the VBA language can be daunting. You do not need to be a VBA wizard to get significant value from macro automation, but it can take some time to get situated with the subset of knowledge that a particular task requires.

- Neither these slides, the presentation, or the sample files are intended as a turnkey tutorial for learning VBA and macro development. It takes time and motivation to get familiar with the pieces involved.
Accessing Office macros and the editor

- Macros live within a document or a template. In Word 2010 (with factory default settings) you can access the **Macro** dialog box from the **View** ribbon by selecting **Macros > View Macros**.

- To run a macro on the currently-active document, select the macro and click **Run**. To open a macro in the VBA editor, click **Edit**.
Sample Office files with macros - Demo

- **field_test_reference_-_graphic_configuration.docm**
  (Word 2010) Contains sample macros to automatically configure graphics.

- **orphaned_bullets_-_style_guide_application.docm**
  (Word 2010) Contains a sample macro that automatically applies paragraph formatting to conform to a hypothetical style guide.

- **simple_macros.xlsm**
  (Excel 2010) Contains some simple macros to demonstrate basic functionality.

- **defect_list_-_spreadsheet_formatting.xlsm**
  (Excel 2010) Contains a macro that dynamically reads the contents of a “defects list” table and does a set of formatting tasks.

- **visio_graphic_export.vsd**
  (Visio 2003) Contains a macro that automatically saves the diagram in PNG and WMF formats.
Photoshop actions

- Photoshop supports a concept of “actions,” which is a pre-recorded set of activities within the interface that can be played back at will.
- While not exactly “programming” in the traditional sense as there is no code development involved, it is a potentially valuable means of automation that is readily accessible to general users.
- Actions are managed through the ACTIONS pod, displayed/hidden by selecting Window > Actions:

```
[Image of Photoshop ACTIONS pod]
```

- The all-important management menu is tucked away in the upper right corner, indicated with a red arrow above.
Sample Photoshop files - Demo

- **raw_screenshot.psd**
  
  (Photoshop CS4) A raw screenshot from a handheld device, viewed in an external VNC viewer. Before publication in the manual, the desire is to extract the actual screen from the VNC window (to make it look as it appears on the device) and add a black border.

- **tcworld_demo_action.atn**
  
  An action file that can be imported to Photoshop CS4 or later. This file contains the action to accomplish the effect described above. In the **ACTIONS** pod, select the menu icon in the upper right corner and then **Load Actions**, then browse to this file. Afterwards, a new “set” called **tcworld_demo** should appear with a single action, **Screenshot prep**. Run this action on the **raw_screenshot.psd** file.
Adobe FrameMaker and ExtendScript

- ExtendScript (ES) is a native scripting language for FrameMaker and many other Adobe products that allows virtually limitless customization and automation of the product.
- ES is part of the product off-the-shelf and does not require additional licensing.
- The language is an extension of JavaScript.
- ES was introduced in FrameMaker 10 and much earlier for other flagship Adobe products.
- For each Adobe product that supports ES, the language varies according to certain product-specific needs. For example, only FrameMaker will have objects, properties, and methods related to books. However, the core of the language is a common library of powerful features that apply across all products.
FrameMaker ES - Strengths/weaknesses

- **Strengths include:**
  - A very powerful and comprehensive set of features for automating and customizing the product.
  - A simple language syntax and runtime parsing; that is, you can write scripts and test with FrameMaker running.
  - A native script development and debugging environment that installs with FrameMaker.

- **Weaknesses include:**
  - Immature documentation. Because the architecture is modeled after the FrameMaker Developers Kit (FDK), persons with FDK experience may have less trouble, but newcomers will struggle with the existing literature.
  - Bugs. Early users of ES have identified several non-critical but still annoying bugs which hopefully will be reduced as the interface matures.
  - Performance. The speed of processing actually is very good, but cannot compare to compiled code such as an FDK plugin.
Scripts are typically authored in the ExtendScript Toolkit editor, launched by selecting File > Script > New Script:

Note that this editor is common to all Adobe products. The product selection in the main toolbar indicates the product to which you are currently writing and running scripts.
The simplest way to run a script is to open it in the toolkit editor and click the run button:

Or, run it through the SCRIPTS LIBRARY pod (File > Script > Catalog).

You can also do the following, all of which involve varying levels of complexity:

- Configure a script to run automatically upon FrameMaker startup
- Set a script to run during a particular user event, such as clicking the mouse, updating a book, etc.
- Create dialog boxes with buttons, etc. that launch scripts
- Create menu commands that launch scripts
FrameMaker ES - Sample script files

- **sample_functions.jsx**
  Contains a series of functions that can be called to do various things. Descriptions of those functions are contained within.

- **graphic_utilities_palette.jsx**
  Contains a function that, when called, produces a modeless dialog box with several controls (such as buttons) for graphic manipulation. When these controls are operated, the underlying code calls functions from `sample_functions.jsx` to do the work. It is intended to show a rudimentary example of creating a dialog box and responding to user actions with it.

- **set_up_menus.jsx**
  When run, this script creates a new menu with a single command that invokes the dialog box from `graphic_utilities_palette.jsx`. It is intended to show how to add your own menus and commands to FrameMaker and how to make them do something.
Adobe FrameMaker Development Kit (FDK)

- The FDK is a set of libraries and source files that allow you to develop custom plugins for FrameMaker.
- The language is C and requires the Microsoft Visual C++ development environment to author and compile.
- It is generally very robust and extremely capable.
- The processing is remarkably fast, able to parse long documents and perform thousands/millions of internal calculations in a flash.
- It is available for no extra charge.
- It is considerably more difficult to master and does not classify well as “light” programming, so the attention focused during this presentation will be high-level and cursory.
- For modest automation and customization tasks, ExtendScript may replace the FDK as the more popular option.
- See the Adobe website for more information and FDK downloads.
Summary and final points

• Software development skills, even modest skills, will set you far apart from your peers and significantly increase your value to the marketplace.

• In many cases, it costs no money to obtain the tools required to develop software. Additionally, the internet is loaded with free tutorials and technical support.

• If anybody says you can’t do it, they are probably lying. If you are technically astute, as most technical writers should be, you can do it.

• If anybody says it is easy, they are lying. You have to really want it and be prepared to work for it. There are reasons why so few people, especially technical writers, acquire these skills.

• You’ll never regret a second that you spend pursuing this type of knowledge, regardless of how far you take it.
QUESTIONS