

Creating HTML Slides

by Grant McLean
grantm@cpan.org



HTML is a great format for presentation slides. All modern desktop systems can view it with standard software; it's a perfect fit for publishing to the web; and search engines love it. However, manually editing HTML pages and managing the links between them is a painful and unpleasant process. That's where Perl can help.

■ Slide composition -----

As Ruby-on-Rails developer David Heinemeier Hansson is fond of saying, "constraints are liberating". When I'm preparing a presentation I find that not having to make decisions about fonts, colours, animation effects, *etc.*, frees me up to concentrate on the real job of entering the bullet points and thinking about the message I want to convey.

To liberate myself from extraneous decision making, I build my presentation slides using a Perl script which imposes certain constraints. Each slide is made up of these objects:

- a title
- an optional decorative image
- a list of content items
- navigation links

Three types of content items I want to support:

- simple bullet text (no nesting)
- code samples (with optional syntax highlighting)
- inline images (typically screenshots)

Figure 1 (next page) shows an example slide that crams in most of these elements.

The system I developed is in many respects a throwaway. I sometimes find when preparing a new presentation that the subject matter, the audience, or even the forum calls for a subtle twist to my standard slide formula. Rather than try to make my code more and more general to cater for all these different possibilities, I simply take a copy of the script and "hack in" whatever changes are required or perhaps even "hack out" some standard feature that's getting in the way.

One Liner

Add a newline to print output with `-l`

```
$ perl -le 'print "Hello World!"  
Hello World  
$
```

and some of my design decisions to give you the background to adapt the script or perhaps develop your own from scratch. This will give you the tools to build a system that meets your needs.

■ What Goes In-----

When I need to create a new presentation, I change into my talks folder; `untar` a skeleton set of files (including the `mkpres.pl` script itself); rename the resulting directory to match the new talk title; and add the new directory to my Subversion repository.

To flesh out the presentation, I change into the new directory; edit the `talk.xml` file; and drop some images into the `html/images` directory. For example, Listing 1 (next page) shows the XML input I used to generate the sample slide illustrated in Figure 1.

Editing raw XML probably sounds like a tedious way to assemble a presentation. Indeed, I originally envisaged building a front-end to hide the XML, using either a GTK interface or a browser-based application. As a stop-gap, I created some macros in my text editor (Vim), shown in Listing 2 (next page), and discovered that the result was much more streamlined than a graphical interface could ever be and I've been happy to stick with that ever since.

The macros work by saving me work as follows:

- **s** Add markup for a new slide leaving the cursor positioned to type the title
- **b** Insert a new line with `<bullet> ... </bullet>` tags, leaving the cursor positioned to type the text
- **c** Add `<code>` and `CDATA` tags which allow code snippets to be pasted in without worrying about normal XML escaping rules
- **r** Add `<screenshot>` tags, leaving the cursor positioned to type the image filename

■ What comes out -----

When I run the `mkpres.pl` script, it generates a series of HTML files. In addition to the main content slides, there's a title slide to act as the front cover. It also creates a table of contents slide to allow me to jump straight to a specific slide—mainly useful when I'm handling questions at the end of a talk.

Each slide file includes navigation links for advancing to