September 2005 Vol. 18 No. 1

MUUGlines

The Manitoba UNIX User Group Newsletter

Next Meeting: September 13, 2005: GPS & Mapping with Open Source tools

If you have ever been lost (and who hasn't), you will know just how important it is to know exactly where you are. With a handy little GPS unit, you can know exactly where you are in the world down to a few meters at the push of a button.

Shawn Wallbridge, will show various tools you can use with (or without) a GPS unit. He will cover various desktop mapping software as well as web based mapping applications.

Where to find the Meeting

Meetings are held at the IBM offices at 400 Ellice Ave. (between Edmonton and Kennedy). When you arrive, you will have to sign in at the reception desk, and then wait for someone to take you up (in groups) to the meeting room. Please try to arrive by about 7:15pm, so the meeting can start promptly at 7:30pm. Don't be late or you may not get in.

Limited parking is available for free on the street, or in a lot across Elice from IBM, for $1.00 for the evening. Indoor parking is also available nearby, at Portage Place, for $2.00 for the evening.

Form & Function - Using Forms On the Web

Part 3 - Client-side Scripting

By Gilbert Detillieux, Computer Science, University of Manitoba, June 2005

In the spring of 1996, I wrote parts 1 and 2 of this article, intending to immediately follow-up with this third part. However, I was distracted by the summer's activities, and never got around to it. I had also wanted to learn more about JavaScript, so that I could focus on it in this third part. That also didn't really happen to the extent I had planned. So, as a result, part 3 was quietly forgotten.

So, why, after almost a decade, am I revisiting this, and finally writing part 3? Part of my motivation is that I've seen a lot of bad uses of client-side scripting, that resulted in less-than-friendly, or at least not very intuitive or convenient forms and web pages. I've also seen a few simple things that can go a long way toward making web-based forms easier to work with, and more pleasant. Rather than explore JavaScript at length (as there are a number of tutorials that already do that), I've decided to focus on simple tips you can use to enhance your web-based forms, using client-side scripting.

Before continuing on, however, you may want to review parts 1 (**http://www.muug.mb.ca/tutorials/form1.html**) and 2 (**http://www.muug.mb.ca/tutorials/form2.html**) of this series, which were published in the May and June 1996 issues, respectively, of MUUG Lines, the newsletter of the Manitoba UNIX User Group. In part 1, we looked at how fill-out forms are coded in HTML. In part 2, we looked at the Common Gateway Interface (CGI), which lets you set up scripts to do server-side processing. If you're already comfortable with this, you should have no trouble with this third part. (You should, as well, already be familiar with JavaScript, as it would be used in HTML forms, since the following examples won't explain the code that is used in any detail.)

Non-intrusive Scripting

Rather than dive into examples, I thought I'd start by explaining my philosophy toward client-side scripting. This will help illustrate what I think separates good technique from bad. The overriding theme is that whatever you do in your script should be non-intrusive to the user: the behaviour of the interface should not be drastically different than what the user would experience if there were no client-side scripting at all.

Keep it intuitive: Users typically have sufficient experience with using web-based forms, that they know how to use them. With scripting, you can do simple things to make the form easier to use, but don't change the way the user interacts with the form to the extent that they have to relearn a new interface. The form should work as you'd expect.

Keep it standard: JavaScript is fairly well defined, but keep in mind that browsers vary a lot in what they'll support, and that there are a lot of browser-specific extensions. Stick to well-defined, standard features, and avoid the rest.

Keep it simple: Client-side processing can help offload work from the server, but you don't want to do too much on the client either (especially since you may have no control over what hardware or software the user will have on the client side.)

Keep it optional: Some users may not have JavaScript enabled on their browser, or may be accessing your forms in such a way that client-side processing may not even be possible. Whatever you do in client-side scripting should only enhance the user experience if scripting is used. It should never make the form impossible to use if scripting is not used. (For example, scripting can be used to auto-submit a form when a critical field is modified. However, you should not eliminate the submit button, assuming it's no longer needed. There's nothing more frustrating than a form you can't complete, simply because you don't have the proper scripting support enabled.)

Focus, Please!

Here's a very simple bit of scripting that can be very helpful, yet not intrusive, nor critical, to the use of a form. This is particularly useful if the form has one input field that the user will always want to enter first, once the page is loaded.

```
<script>
function set_focus()
{
  document.sample_form.input_field.focus();
}
window.onload = set_focus;
</script>
```

The HTML code for the corresponding form would look something like this:

```
<form method="post" name="sample_form"
action="/cgi-bin/sample.cgi">

<input type="text" size="30"
name="input_field" value="">

<input type="submit" value="Go">

</form>
```

Note the names given to the form and input field must correspond to those used in the script that refers to them. You are, of course, free to pick whatever suitable names you would like to use.

When the document containing the above script and form is loaded into a browser window, the keyboard focus will automatically be set to the named input field. The user can then start typing into that field right away, without having to click the mouse in that field first. However, it's not a big loss if scripting is disabled: the form still works as the user would expect, and only a bit of convenience is lost.

Auto-submission, If You Must...

Although I'm not a big fan of auto-submission - after all, the user should be in control of when the submission actually happens - there may be times where you want to do this. Here's a fairly simple way to code it, while still allowing submission by the traditional means. In your form, the input field(s), on which you'd trigger form submission when changed, would look like this:

```
<input type="text" size="30"
name="input_field" value=""
onChange="this.form.submit()">
```

But remember to include an actual submit button as well, in case JavaScript is not used by the client.

Error Checking

A very good use of client-side scripting is to perform simple error checking on input fields as they are changed. That way, the user gets immediate feedback on what needs to be fixed, rather than submitting bad data to the server, and having to reenter the correct information again. The onChange attribute, shown above, can be used to call up the appropriate function to perform the error checking.

Such a function could issue an alert message in a pop-up window, or clean up the input field, as appropriate.

In addition to checking fields as they are entered, you may also want to perform further checking on the entire form's content before allowing it to be submitted. That can be accomplished as follows:

```
<form method="post" name="sample_form"
action="/cgi-bin/sample.cgi"
onSubmit="check_form()">
```

You would then include a function called check_form() within <script> and </script> tags. This function would do the appropriate checking, then return a value of false to prevent the submission from being done, or a value of true to continue with the submission. The onSubmit attribute can alternatively be used on the input tag for the submit button, rather than on the form tag.

Note that client-side error checking is merely a convenience for the user. It should never be used instead of server-side checking. Since the client-side checking may not even be performed, and since you may not have any control over what clients submit to the server anyway, you still need to do thorough checking of all input on the server side.

Field Completion, Formatting, and Calculation

Similar to the error checking above, another function that can be performed on the client side is to fill in or adjust particular fields in response to values entered in other, related fields. Also, you could clean up the formatting of an input field to a more common, consistent format. (For example, adding dashes at the right places to a phone number that was entered.)

Since the possibilities are limited only by your imagination, and by the limitations of the scripting language used, there's a lot you can do. You could even have code that computes values to be filled in for particular fields, based on values in other input fields. Examples on the web abound, and some of them are quite sophisticated, such as various types of calculators that give users the results they need right from the browser - without having to actually submit the form to a server for processing!

Putting a Label on It

Although the following tip has nothing to do with client-side scripting, per se, it is a nice use of an HTML 4.0 extension (now recognized by most browsers), which makes forms easier to use. By using label tags around the text associated with particular input fields (check boxes and radio buttons, in particular), you make it easier to select these fields by clicking, since users can now click anywhere within the label text. For example:

```
<input type=radio name=sort_order value="A"
id="ascend"><label
for="ascend">&nbsp;Ascending</label>

<input type=radio name=sort_order value="D"
id="descend"><label
for="descend">&nbsp;Descending</label>

<input type=checkbox name=full value="yes"
id="full"><label
for="full">&nbsp;Long&nbsp;format</label>
```

Now, instead of having to click exactly in the radio button or check box, the user can click on the associated label text to select that particular input. Label tags can also be used in a similar way for text input fields, so that the keyboard focus goes to that input field when the label is clicked.

Note that this example is also in keeping with my philosophy of simply enhancing the form in a non-intrusive way, and that the form will still function correctly in older browsers that don't recognize the label tags.

Further Reading

A nice, concise tutorial on JavaScript, focussing on use in forms, is available from Olaf Gobruen's personal site. (**http://home.att.net/~gobruen/progs/javascript/index.html**) “Using window.onload” is a brief tutorial on just that. (**http://javascript.about.com/library/blonload.htm**) A good summary of HTML 4.0 extensions is “HTML 4.0 Explained.” (**http://www.yourhtmlsource.com/accessibility/html4explained.html**)

This article is the third part of a series on using web-based forms. Part 1, on coding forms in HTML, can be found online here: **http://www.muug.mb.ca/tutorials/form1.html** Part

2, on writing CGI scripts, can be found online here: **http://www.muug.mb.ca/tutorials/form2.html** The current version of this third part can be found online here: **http://www.muug.mb.ca/tutorials/form3.html**

OpenSSH 4.2 released

2005-09-01 - OpenSSH 4.2 has just been released. It will be available from the mirrors listed at **http://www.openssh.com/** shortly.

OpenSSH is a 100% complete SSH protocol version 1.3, 1.5 and 2.0 implementation and includes sftp client and server support.

We would like to thank the OpenSSH community for their continued support of the project, especially those who contributed source, reported bugs, tested snapshots and purchased T-shirts or posters.

T-shirt, poster and CD sales directly support the project. Pictures and more information can be found at **http://www.openbsd.org/tshirts.html** and **http://www.openbsd.org/orders.html**

MUUG Board Elections - Call for Nominations

Every October the Manitoba Unix User Group holds its Annual Meeting, the main goals of which are to elect a new Board of Directors and to pass any special resolutions. (Aside from that, it is a regular meeting.) Any member in good standing can be nominated to run for a position.

As of this writing, the following members of the current Board have let their names stand for re-election:

Gilbert Detillieux	Systems Analyst	University of Manitoba
Kevin McGregor	Systems Specialist	City of Winnipeg
Doug Shewfelt	Systems Specialist	City of Winnipeg
Adam Thompson	Consultant	athompso.net
Shawn Walbridge	System Administrator	Frantic Films

Of course, this list is just a starting point. Any member in good standing of the group can be nominated simply by getting the support of one other member. If you feel you would like to contribute to the group by running for a board position, please don't hesitate to do so. (In fact, we'd like to see the number of board members increase.)

If you want to be nominated, or to nominate someone else, send a letter to the group's postal box or deliver it in person to a current board member. The letter must contain the name, title, and employer of the nominee, along with a short (100 word or so) biography, and must contain the signatures of the nominee and one other member. The letter must be received no later than September 27, 2005, which is 14 days prior to the October 11 meeting.

Although the by-laws require that the nominations be done in writing, with signatures, you can speed up the process by sending us e-mail to <**election@muug.mb.ca**>, with the above information, and sending the signed paper copy later. In this case, please include the e-mail address of both the nominee and the supporter on the CC: list of the message, so that all parties concerned have a record of the communication.

Nominees should familiarize themselves with the MUUG bylaws, found here:
http://www.muug.mb.ca/pub/bylaws/

If you have any questions about the election or the nomination process, please contact Gilbert Detillieux, either by phone (474-8161) during business hours, or by e-mail to <**election@muug.mb.ca**> anytime.

Gilbert Detillieux, Election Committee Chair

Sending Us E-Mail?

Due to the amount of e-mail MUUG receives, we've set up an auto-reply to give you immediate feedback, and redirect some of the e-mail to the appropriate places. Why not look at **http://www.muug.mb.ca/about.html#contacts** first?

Share Your Thoughts

E-mail us with your comments on the newsletter, whether it's criticisms or commendations, and continue to send in articles or ideas for the same. Specifically, what sort of material you would rather see: Announcements, technical articles, new products, or…?

If you have a How-To or other idea, and aren't ready to give a presentation at MUUG, an article is a great alternative! If you can write better than the editor, that's terrific; if you can't, submit it anyway and we'll get it into shape for publication. We know that many of you have some great ideas and lots of knowledge. Why not share? Send Mail to: **editor@muug.mb.ca**.