WebDialogs – The Lost Manual

Revision: R1 - 09 November 2009

I've recently been working more with WebDialogs and I have noticed there's a great deal of got-cha's which is not mentioned in the documentation, which you can find buried in this forum. Therefore I want to collect all this info into one single post where all the important stuff is in the top - not scattered in various threads or buried deep in a topic. Please drop me a message or add a comment to this thread for more information to add.

Contents

Cross Platform

Async vs sync Communication

On PC the communication between javascript and ruby is synchronous. That means when you call a ruby method using `window.location = 'skp:rubyMethod'`, the javascript will wait for the ruby method to complete before proceeding with the next javascript statement.

On OSX it just sends the command and continues with the javascript without waiting - making it impossible call sequential callbacks to ruby too quickly as they will override each other.

Calling from ruby to javascript using `.execute_script` is synchronous on both platforms. The method will wait for the javascript to finish before processing next line of ruby.

So you can do stuff like this:

```
dialog.execute_script('add(2,3);')
value = dialog.get_element_value('sketchupPump')
```

The add javascript function will here take both arguments and add them together and put the value into a hidden input field with the id `'hidden_input'`.

```
function add(n1, n2)
{
    document.getElementById('sketchupPump').value = n1 + n2;
}
```

The input field in the HTML is something like this:

```
<input id="sketchupPump" type="hidden">
```

Further reading:

- http://forums.sketchucation.com/viewtopic.php?f=180&t=13394
- http://forums.sketchucation.com/viewtopic.php?f=180&t=22698

window.onload

On Windows, the HTML document appear to be created when you call `dialog.show`. If you attach an event to `window.onload` which send a command back to ruby to print `"Hello World"` in the console you'll see `"Hello World"` printed as you call .show. But on OSX it seems that the HTML document is created as you use `.set_file` or `.set_html` - before calling `.show`.

Example script:

```
module TT_Test
	@d = nil
	@d = UI::WebDialog.new('On Load Test')
	@d.add_action_callback('onload') { |dialog, params|
		puts '>> window.onload'
	}
	@d.set_html('<html><body onload="window.location=\'skp:onload\';">Onload Test</body><html>')
	def self.onload
		@d.show {
			puts '>> ruby block'
		}
	end
end
```

On Windows, when you run the script:

```
load 'test/webdialog.rb'
true
TT_Test.onload
true
>> ruby block
>> window.onload
```

Notice that the ruby block executes before the WebDialog `onload` event, which might indicate the block is run before the HTML document is ready.

On OSX:

```
> load 'test/webdialog.rb'
true
>> window.onload
> TT_Test.onload
true
```

Notice that `window.onload` triggers immediately as we've added the HTML to the WebDialog object. And that the `.show` block never executes.

So if you create a WebDialog object when you load your plugin, beware that it will consume resources from the moment you add the HTML and not wait for the dialog to actually show.

WebDialog.show / WebDialog.show_modal

On OSX, `.show_modal` only makes the window stay on top of the Sketchup window, but the window is not really modal.

On PC the window always stays on top of Sketchup, but `.show_modal` is truly modal.

It also seems that when you pass a block along with `.show` or `.show_modal`, the block will never execute on OSX.

Quirks vs Standard vs Superstandard

When you create a HTML document, it behaves differently depending on the DOCTYPE you choose. You must choose the DOCTYPE you want to work with. For further reading: http://www.quirksmode.org/css/quirksmode.html

Note! Because WebDialogs are embedded browser controls they behave differently than a normal website on Windows. Internet Explorer 8, as a browser, will default to Super Standard mode when you use Strict DOCTYPE. But when embedded as a WebBrowser object it will default to IE7 compatibility mode. Microsoft says that you have to set a registry key for the application that embeds the WebBrowser to make it use IE8 rendering mode. But of course we can't do that for Sketchup since some plugins might rely on the IE7 mode.

But what you can do is include the meta tag `<meta http-equiv="X-UA-Compatible" content="IE=8"/>`. Note that this meta tag should be placed at the top of the <head> tag.

Beware that the user agent string will still report `MSIE 7.0` for embedded WebBrowsers - even though you use IE8 mode. This differs from when you test the same HTML in the normal web browser where it returns `MSIE 8.0`. To check the rendering mode: `document.documentMode`.

HTML, CSS and Javascript

Personally I use Strict DOCTYPE as it render more consistently between platform.

To simplify javascript work I use jQuery which is a lightweight framework that let you quickly manipulate the DOM and deal with events in a manner where you don't have to concern yourself much about differences between the platforms.

For in introduction to HTML and CSS I recommend HTML Dog. It's up to date and will recommend best practices. http://htmldog.com/

For lookup references to most web related languages: http://www.w3schools.com/

The ultimate reference to web standards: http://www.w3.org/

WebDialog.new

You can specify the arguments as either a comma separated list, or as a single Hash object. Currently the manual is missing information for both of these.

Comma Separated Arguments

- `dialog_title` The title to be displayed in the webdialog.
- `scrollable` true if you want to allow scrollbars, false if you do not want to allow scrollbars.
- `pref_key` The registry entry where the location and size of the dialog will be saved. If preferences_key is not included, the location and size will not be stored.
- `width` The width of the webdialog.
- `height` The height of the webdialog.
- `left` The number of pixels from the left.
- `top` The number of pixels from the top.
- `resizable` true if you want to allow the window to be resize by the user.

Hash argument

```
keys = {
    :dialog_title => title,
    :scrollable => false,
    :preferences_key => 'MyDialog',
    :height => 300,
    :width => 400,
    :left => 200,
    :top => 200,
    :resizable => true,
    :mac_only_use_nswindow => true}
@dialog = UI::WebDialog.new(keys)
```

Note that when using a hash, you cannot set the title. But the hash exposes a new undocumented argument, `:mac_only_use_nswindow`.

Fixed window size

When you specify position and size Sketchup only uses those values the first time you create the dialog. After that it reads the last used values from the registry. That might be want to you want for resizable windows. But maybe not for windows with a fixed size which the user can't resize.

When you develop a plugin you might find that you need to change the size of your fixed size window, but you can't see the effect because Sketchup just reads your last values. In which case you need to delete the registry settings for your Webdialog, or use the `.set_size` method after you create the fixed size window to ensure the correct size.

And if you do not specify a preference key, Sketchup seem to disregard both size and position.

Scrollbars

The scrollbar argument does not seem to work on PC. In order to disable the scrollbars you must set a CSS property. What HTML to assign this property to depends if your HTML uses Quirks Mode or Standards Mode.

If you're using Quicks Mode you assign it to the BODY element:

```
body {  overflow: hidden; }
```

If you are using Standard Mode you assign it to the HTML element:

```
html {  overflow: hidden; }
```

This is because the effective root element in an HTML document differs from Quicks to Standards Mode.

Paths - WebDialog.set_file vs WebDialog.set_html

When you use `.set_file` to populate the HTML document, all resources (images,CSS,scripts etc.) will be relative to the file you specify.

But when you use `.set_html`, all resources are relative to a temp file created on the system. So if you need to reference external resources from the HTML, either use absolute paths, or add a BASE tag to the HEAD of the HTML - specifying where relative paths should be resolved from:

```
<base href="c:/absolute/path/" />
```

Bugs and Issues

OSX .execute_script and quotes

There was some issues with `.execute_script` and quotes in Sketchup prior to 7.0 on OSX.
http://forums.sketchucation.com/viewtopic.php?f=180&t=8316#p49259

WebDialog Javascript callback Maximum message size

The Javascript callback to ruby has a size limitation. When sending data back to ruby it's better to store the data in a hidden input field and use `.get_element_value`.
http://forums.sketchucation.com/viewtopic.php?f=180&t=8683#p52107

#<Exception: Invalid Dialog>

Unknown error. Caused by BoundingBox?

http://forums.sketchucation.com/viewtopic.php?f=180&t=22567#p190035

<LocalJumpError: return from proc-closure> in WebDialog

Issues with using return in callback methods from Javascript? Think I've experience this myself.

http://forums.sketchucation.com/viewtopic.php?f=180&t=22022#p185099

Debugging

You can use Firebug Lite to help debugging your HTML+JS in WebDialogs.

http://getfirebug.com/lite.html

Thanks

Thanks to everyone on SCF that has contributed with their share of this collective knowledge.