TeXWorks Scripting (QtScript) and related

Current Writing Activity 2012 05 14th Thursday (NZST)

2012 08 26

Cursor Position...

Added note from http://code.google.com/p/texworks/source/detail?r=1019 regarding new cursor position property requested by Jonathan Tomm ... http://code.google.com/p/texworks/issues/detail?id=592

TW.target.cursorPosition()

2012 05 14

Spelling functions: Spell Checking
Global Variables — Un-setting Global variables: Globals

2012 03 (March)
(small fixes)

2012 02 12th Thursday

Added note to section heading for TW.target Window functions and procedures.
Added note to TW.target.showSelection()

Updated launchFile(QString) with http:// information

Added Build in Update Checking

2012 01 11th Wednessday

Wrote up new TW.script. properties introduced by Stefan Löffler in r.952 (.fileName, .title, .description, .author, .version) Script properties (fileName etc)

Script property TW.script.**fileName** now depreciates __FILE__

2011 11 06th Sunday

Updated the TW.target text category text.
balanceDelimiters() ... toggleCase()

... More details, some examples, and presentation upgrade.

2011 08 23rd Tuesday

System and File access advices updated...
Use with care a write or system call could damage a User's system.

2011 08 1st Monday

Updated set_syntax.js Hook script and various under TeXworks as a QtScript editor
Fixed leading numeral problem in tag outline regular expression in Worked Example - Consistently Filling Drop Down Boxes etc

---

2011 07 31st Sunday

Wrote up working examples of using TW.system() calls and LaTeX \immeadiate \write18{}

Getting Take Aways (sending out) - TW.app.system()
    JabRef - MySql - Bibliographies - Using php to return text
    Worked Example - Consistently Filling Drop Down Boxes etc

---

2011 07 23rd Saturday
24th Sunday
26th Tuesday

Mention of new 'free' JavaScript e-books

Special mention: Eloquent JavaScript A Modern Introduction to Programming

Added details on a simple naming convention for working between .js and Qt Creator and .ui, and auto-opening the correct dialogue to work on. Script/Interface Naming

General update over Api, added file overview File Functions, Procedures and Properties
Updating TW.app and TW.target sub entries

Tw.app tree slightly reorganised to place most Don't Use items in separate area.

TW.target File and Find/Replace entries updated.

General update all over.

Yet to update PhpJs Function List and TeXworks as a QtScript editor and its associated listed files.

Updated setSyntaxColoringMode(QString)

2011 06 23rd Thursday

Added a small explanatory example at the top of JSON

TW.result and TW.target.consoleOutput

2011 06 22nd Wednesday

Updated UI Dialogue/Form Scripts
to include .deleteLater() for forms/dialogues
and a possible approach to initialisation and finalisation details.

Updated `progressDialog`

2011 05 14th Saturday

Wrote introduction to using Qt Creator to make GUI forms for scripts, and handling signals (initialisation and finalisation), and objects in script. UI Dialogue/Form Scripts

Revamped pages under heading TeXworks as a QtScript editor

Various corrections during last month.

2011 04 17 April

Updated information for

system(QString) examples

updated helper_twPan.mod

2011 03 14 Monday)NZST)

Added in JavaScript Garden a fine description of Javascript - QtScript language family - must read!

Note: Qt have their own modifications to the Object prototype http://doc.qt.nokia.com/latest/scripting.html and specifically http://doc.qt.nokia.com/latest/scripting.html#prototype-objects-and-shared-properties

Put updated source file on web

2011 03 12 Saturday (NZST)

Completed noting unusable signals / methods as per Saturday, 1 January 2011, 2:42:26 p.m. (SL)

Writing up Tw.target

2011 02 13 Sunday (NZST)

Started on openFileFromScript(QString,int,bool)

2011 02 12 Saturday (NZST)

Started writing up the message and input boxes:

- message boxes
  - critical(QWidget*,QString,QString)
  - information
  - question
  - warning
- progressDialog(QWidget*)
- get User Input
  - get Text
  - TW.getItem - get List Choice
- get Numerical Input
  - getInt(QWidget*,QString,QString,int,int,int,int)
  - getDouble(QWidget*,QString,QString)
- Setting Preferences
- Scripts Management

2011 02 10 Thursday (NZST)

Various edits and additions in the last week.

CHM and PDF available at

http://twscript.paulanorman.com/docs/index.html

http://twscript.paulanorman.com/

2011 01 26 Wednesday (NZST)

Writing up system(QString)

Examples yet to be added

2011 01 25 Tuesday (NZST)

Redrafted panTw and loading modules
Added panTw -
bibleNamesAbrvs array of names and abbreviations
getBibleVersion()
getDate

Moved creation of PhpJs ($P) into top of panTw (also means that panTw can use PhpJs in scope) and left PhpJs able to be created separately from panTw id panTw not in use.

Updated makeScript.js to reflect changes.

Updated existing scripts that used either panTw or PhpJs to reflect the fact that panTw creates PhpJs automatically.

Preparing to release Tw modules for panTw and PhpJs.

2011 01 17 Monday (NZST)

Updated copyright notices.

Updated and moved helper modules. User 'Library' Modules and helper objects
download being prepared.

2011 01 16 Sunday (NZST)

Edited TW.app.applyTranslation(QString)

added tr - Turkish
TW.app.cursorFlashTime
TW.app.doubleClickInterval

2011 01 13 Thursday (NZST)

Up to TW.app.applyTranslation(QString)

2011 01 04 Tuesday (NZST)

Various fixes ad updates.

2011 01 03 Monday (NZST)

Wrote up adding file extensions to the
File Open dialogue under:
TeXworks as a QtScript editor
list in texworks-config.txt

Inserted more of Stefan's comments.
Wrote up some commands.

2011 01 01 Saturday (NZST)

Happy New Year - Every One!

Wrote up `TWScriptAPI::launchFile`

Inserted a number of Stefan's comments.

2010 12 19 Sunday (NZST)

updated syntax-patterns.txt

updated pan_Tw.mod

updated tag-patterns.txt

updated PhpJs twPan twConst msgBox

Made correction to getOpenFileName()

Set message box constants get User Input

2010 12 15 Wednessday (NZST)

File Functions readFile() writeFile() Return Values for Status

Globals Default Conversion from Qt Script to C++

2010 12 14 Tuesday (NZST)

File Functions

Globals

2010 12 13 Monday (NZST)

Writing up hook scripts. Automatic Hook Scripts

2010 12 12 Sunday (NZST)

Been bogged down with TW.getItem() returning unexpected values last week or more. Taken up a lot of time. See http://developer.qt.nokia.com/forums/viewthread/2280 and TW mailing list http://tug.org/pipermail/texworks/2010q4/003487.html

Done some basic tasks on Api

Working through TW.app.getFileName(s) etc... getOpenFileName() and forward.

2010 11 23 Tuesday (NZST)

Completed initial skeleton of TW api functions properties tree

Prepared OpenOffice document of above - for Stefan to fill in notes.

2010 11, 21 Saturday (NZST)

Working on syntax patterns for using Tw as editor for QtScripts

Writing introduction to Tw as a Script IDE TeXworks as a QtScript editor

tag-patterns.txt for Tag tree Window *function* **myFunctionName***()* and *//%:* **blah blah** as tagged display information

(All of these sections are a work in progress!)

2010 11, 19 Friday (NZST)

Working on syntax patterns for using Tw as editor for QtScripts

Made some of the needed key words and did some trials.

syntax-patterns.txt

2010 11, 11 Thursday (NZST)

Tim Caswell (http://howtonode.org/) has given permission for his articles on Object design and usage to be used in these notes.

Working on TW.target target

2010 11, 10 Wednessday (NZST)

Working on TW.target target

made batch script (see TW) to explore .h and .cpp in TeXWorks source.

Example of TW.target documentation here

Stefan Löffler Am 2010-07-27 02:53, schrieb Paul A Norman: Writing libraries for common code... 2 Aug

from **Paul A Norman**<paul.a.norman@gmail.com> hide details 7 Nov (5 days ago)
to "Discuss the TeXworks front end." <texworks@tug.org>
date 7 November 2010 00:38
subject Re: [texworks] Scripting: import libraries, retrieving saving text. web location, Api
mailed-by gmail.com

Ok, as discussed earlier (August), here is a little bit of progress on sharing notes, experiences, and information on Tw Scripting.

This is just a start - but even some of the Tw Api structure is in place ready for filling out.

The table of contents gives an idea of where I am aiming (book marks in the .pdf, or left hand TOC links in the .chm)

This is very much from our experiences with QtScript, so there is not much that is Python or Lua specific.

Links to the .chm and .pdf are at the top left hand column of this page here: http://twscript.paulanorman.com/

Any early feedback would be appreciated thanks.

Paul

- Show quoted text -

Reply

show details 9 Nov (3 days ago)

Paul A Norman

to Discuss

Got a bit more done on the TW.target hierarchy.

Stefan, I'll be able to pass you the source for all that once its useful - even as an Open Office document or LaTeX from there, but if you're not using MS Windows, the actual help generator may run under WINE but would require a couple of things if it would work there at all. If you still have access to Windows I can give you the .hnd file for HelpNdoc, that will all end up on the web with the 'finished' help any way.

Any one with any Scripting Api or related notes, even if rough, please
throw them at me.

I'll do my best looking through the .cpp(s) but I am not a native C++ developer.

Paul

Reply

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

Main Web Page

Main TwScripting Notes page: http://twscript.paulanorman.com

Home Page: http://PaulANorman.com

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

Welcome

TwScript.paulanorman.com

Welcome to Some Information on TwScripting

This is not an official manual

A collection of notes and ideas I've found useful
and which is still very much being
written and edited!

Rough and Ready Draft - July 24th, 2011

Paul A Norman
http://PaulANorman.com

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

Introduction - Disclaimer - Copyright

This information, prepared for our internal use, focuses on a scripting environment, QtScript, that has been built in as the native scripting tool for Tw. QtScript is an EMCA script, virtually equivalent to JavaScript with some additional modifications (mostly in prototyping) made by Nokia the makers of Qt, and the Api that the developers make available for Tw itself.

There are Tw modules for incorporating Python and Lua as well, though at present the full Tw Api may not be available to those tools.

I've tried here merely to pull together information on the Api for Tw and some references on using QtScript peculiar to Tw, that we rely on and use.

I have tried to generalise these where this may be useful.
It is my hope that others may find this some help, and we give it back as our small contribution to the Tw and TeX community.

Disclaimer

I can make no promises as to the accuracy of this information, nor provide any warranty for safe application in any thing you choose to do, nor be held responsible for anything that happens from your choice to use any information contained in here or pointed to or linked to elsewhere.

Copyright

copyright (c) 2010 2011 2012 Paul Anthony Norman. http://PaulANorman.com P.O. Box 1005, Nelson 7040, New Zealand.

Portions copyright,
TeX Users Group - President Karl Berry karl@freefriends.org
Jonathan Kew jfkthame@googlemail.com
Stefan Löffler st.loeffler@gmail.com
Alain Delmotte esperanto@swing.be
Kevin van Zonneveld http://phpjs.org php.js is copyright 2010 .

Triiple licensed under the

GPL (GPL-LICENSE.txt)

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT.
IN NO EVENT SHALL PAUL ANTHONY NORMAN BE LIABLE FOR ANY CLAIM,
DAMAGES
OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN EITHER THE SOFTWARE OR DOCUMENTATION.

MIT (MIT-LICENSE.txt)
licenses.

GNU Free Documentation License version 1.3 http://www.gnu.org/licenses/fdl.html

TwScripting Notes page: http://twscript.paulanorman.com

Home Page: http://PaulANorman.com

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

MIT-LICENSE.txt

Permission is hereby granted, free of charge, to any person obtaining a
copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS

OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT.
IN NO EVENT SHALL PAUL ANTHONY NORMAN BE LIABLE FOR ANY CLAIM,
DAMAGES
OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN EITHER THE SOFTWARE OR DOCUMENTATION.

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

GPL-LICENSE.txt

http://www.gnu.org/licenses/old-licenses/gpl-2.0.html

Copyright (c) 2010,2011 Paul A Norman and others ijn their own right

```
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301,
USA.
```

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

Sources of Information

Some Sources of Information Used Here

Jonathan Kew jfkthame@googlemail.com

Stefan Löffler st.loeffler@gmail.com

http://code.google.com/p/texworks/

http://code.google.com/p/texworks/wiki/TipsAndTricks

http://code.google.com/p/texworks/wiki/ScriptingTeXworks

http://code.google.com/p/texworks/wiki/AdvancedTypesettingTools

List of some scripts: http://code.google.com/p/texworks/issues/detail?id=261&can=5

http://qt.nokia.com

http://doc.qt.nokia.com/4.4/qscriptengine.html

http://doc.qt.nokia.com/4.4/qtscript.html

http://doc.trolltech.com/4.4/ecmascript.html

http://developer.qt.nokia.com/search?search=QtScript&x=0&y=0

Nokia Corporation:
Â© 2008-2010 Nokia Corporation and/or its subsidiaries. Nokia, Qt and their respective logos are trademarks of Nokia Corporation in Finland and/or other countries worldwide.
All other trademarks are property of their respective owners. Privacy Policy

Licensees holding valid Qt Commercial licenses may use this document in accordance with the Qt Commercial License Agreement provided with the Software or, alternatively, in accordance with the terms contained in a written agreement between you and Nokia.
Alternatively, this document may be used under the terms of the GNU Free Documentation License version 1.3 as published by the Free Software Foundation.

Alain Delmotte's manual found under Menu://Help inside Tw, or here for now:
http://www.leliseron.org/texworks/
(Update on Manual by Stefan Löffler in progress 2010.)

And postings on http://tug.org/mailman/listinfo/texworks mostly by Jonathan Kew (TeXworks project leader, and copyright holder) and Stefan Löffler (a major developer) and myself a lowly user.

JavaScript/EMCA

https://developer.mozilla.org/en/javascript

https://developer.mozilla.org/en/JavaScript/Guide

https://developer.mozilla.org/en/JavaScript/Reference

http://devedge-temp.mozilla.org/library/manuals/

http://brendaneich.com/

http://www.davidflanagan.com/javascript5/

http://bonsaiden.github.com/JavaScript-Garden/

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

Getting Started

Obtain TeXworks

If you have not already got it, get a copy of TeXworks.

As at July 2011 this is the development site to use: http://code.google.com/p/texworks/
Look under Featured downloads:

If that is down, or no longer in use, then the most likely permanent site that will at least tell you where to get the most up to date version is this one: http://texworks.org redirects to http://www.tug.org/texworks/

What is TeXworks technically?

It is an Open Source, multi OS, compiled Qt framework C++ application. released under the GPL version 2 (as at revision 0.3 ver 649).

It provides editing and *Tex running tools with an in place pdf viewer that can be live synced to the editing of source *TeX text.

If you are reading this then you already know that part of Tw can be scripted to extend or refine its functionality.

What is Tw Api?

Tw Api is basically what the window object is to browser scripting.

Its a description of the functions and properties that can be reached through scripts written for use with Tw.

The Tw Api is structurally and conceptually very different to the browser objects you might be familiar with in web page scripting, but it fulfills many of the same purposes — it lets you 'hear from' and 'talk to' the underlying application and its windows and parts or 'widgets' as the Qt framework that Tw is built on, calls them.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

System requirements

From the Tw Development web site:–

"The current version of TeXworks has been successfully built with

- Xcode (using gcc 4) on Mac OS X (built on 10.5, but should run on 10.4 or later)
- MinGW release 5.1.4 on Windows XP (also runs on Vista and Windows 7)
- gcc 4 on GNU/Linux, various BSDs, etc. "

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Getting Tw Scripting Help

Mail List

Before using the mail list check its **archives** out: http://tug.org/pipermail/texworks/

And search on the Development site's **Issues List**: http://code.google.com/p/texworks/issues/list

There is a **mail list** available for TeXworks, you could start your subject line:

SCRIPTING: What You Want Help on

— that would help people avoid it, if they are not doing any scripting:–

http://tug.org/mailman/listinfo/texworks

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

Setup for Portable Usage

Setup for Portable Usage

We have chosen to use Tw in a portable type mode so at a pinch it can be copied to a portable or USB type drive for remote usage.

However this configuration type option also can make it easier to find the related files you may need to access, if you are not concerned about having one set of Tw configuratoin files for a computer, even if there are multiple users of it.

Many of the issues are covered in this Tw issues entry:–

http://code.google.com/p/texworks/issues/detail?id=95#c14

And in the Tw Manual, look for section 5.7 or in the index:- texworks-setup.ini

To do this we followed these steps:

1. A plain text file called `texworks-setup.ini` was created in the same directory as the TeXworks.exe
2. We chose to use purely relative paths in it
3. This is all that is in our file(s).

```
defaultbinpaths = ../../../LatexUtils/MiTeX 2.8.3541 Portabe/miktex/bin
inipath = ./config
libpath = ./config
```

This causes our Tw script directory to be below where the Tw executable is:

TeXworks/config/scripts

(And also means that the general config files for Tw are in the same

tree.)

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

Scripts Basics

Scripts Basics

These are just a few things about scripts required to get started, more is covered lower down under Script Authoring.

For a review of Qt's QtScript model see http://doc.qt.nokia.com/latest/scripting.html

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Setting Preferences

TexWorks Scripting Application Programing Interface

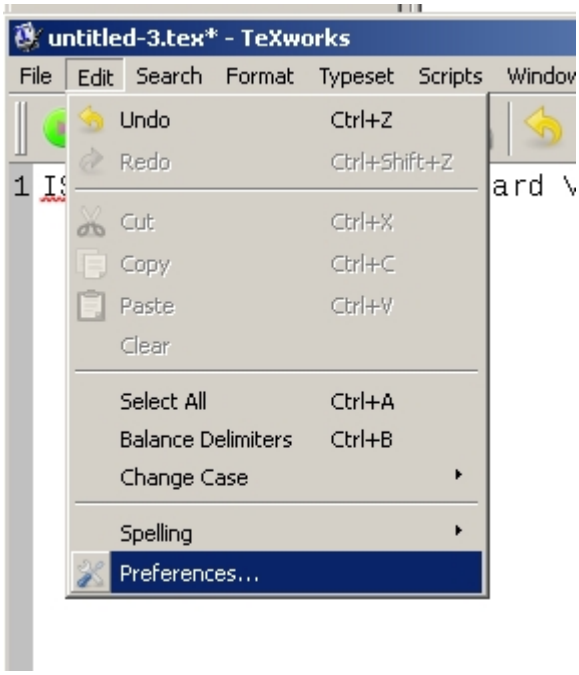
Tw Api

The Api may be viewed in the QtScript debugger that comes behind the scenes with Tw.

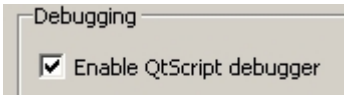
To show the Tw Scripting Api, we'll walk through a few needed first steps.

You need to activate the Debugger.

Go - Menu://Edit/Preferences

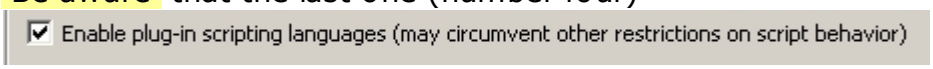


If not already ticked, tick – Enable QtScript debugger.



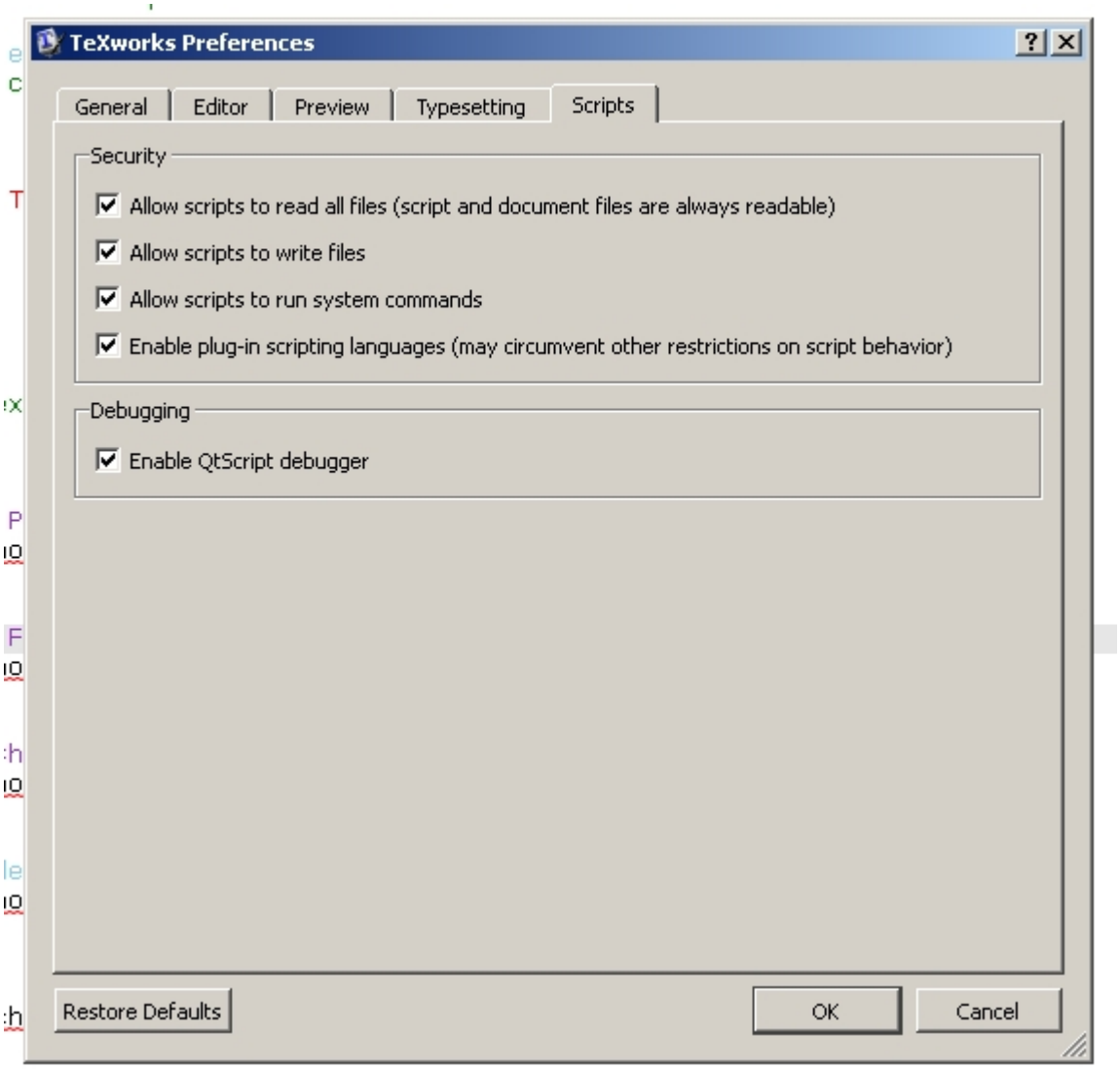
And consider the other options here.

Be aware that the last one (number four)



could allow anything to happen even if you have not ticked the top three.

The other scripting languages referred to are, at present, Python and Lua.



If you want TwScript to get text back (for e.g. for QtScript to insert in your editor) from the command shell, via DOS, Perl, Php, Python etc ... or to actually start other programs, then enable ☑ Allow scripts to run system commands
- be aware this will allow a TwScript to do anything your system level of login lets you do - so read any scripts that you do not write yourself first before using if you tick
this ☑ Allow scripts to run system commands or the ☑ Allow scripts to write files option.

To get the debugger to show, run a script with a mistake in it

Contd ...

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

Scripts Location

This will do it:—

```
// TeXworksScript
// Title: Debug
// Description: Starts the Debugger
// Author: Jonathan Kew
```

```
// Version: 0.1
// Date: 2010-03-18
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+S, Alt+D

debugger;
```

In a plain text editor save the above into where ever your TwScripts are.

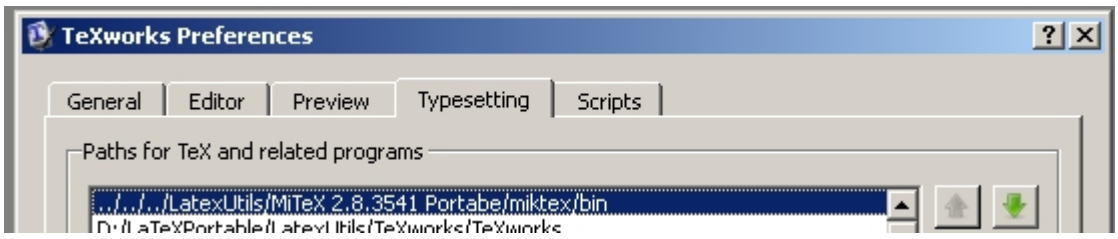
Location of TeXWorks Scripts

> "Scripts for TeXworks should be created as text files with the extension `.js` and placed in the `scripts` subdirectory of your TeXworks resource folder (typically found within your home directory, exact location is platform-dependent).
>
> The first time TeXworks 0.3 is launched, it will create the `scripts`folder if it does not already exist, and install a few example scripts there. If more examples are added in later releases, you will need to remove or rename the `scripts` folder in order for the new ones to be installed; the program will not modify an existing `scripts` folder." ScriptingTeXworks

You can open the Scripts folder from the TeXworks editor Menu> Scripts/ Scripting TeXworks / Scripts Folder

(Other configuration files can be accessed from Menu> Help / Settings and Resources)

I reconfigured my Tw install to a directory where I can pull it off for portable use along with MikTeX portable. (If you do that you'll have to set the Tw preferences to tell it where MiKTeX is —)



So under MS Windows my scripts are below where TeWorks.exe is, i.e. / TeXworks/config/scripts

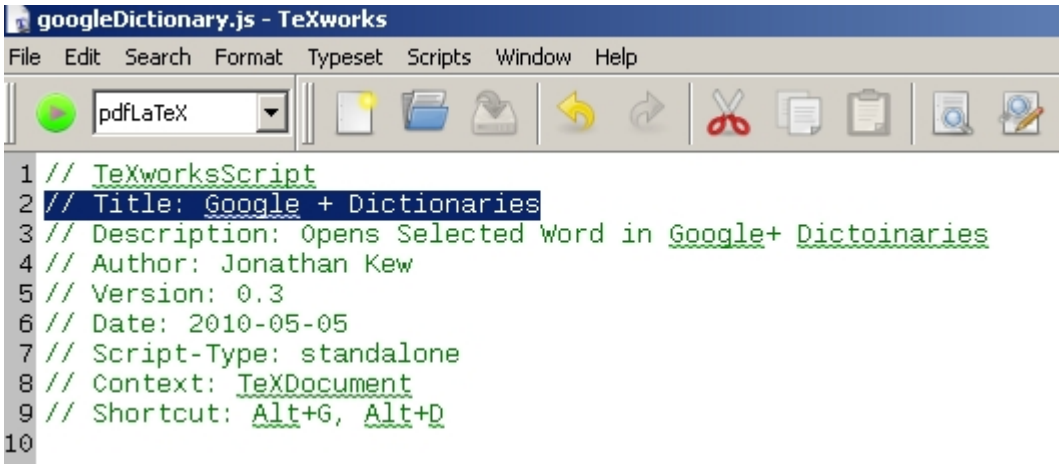
On Windows, yours might be:—

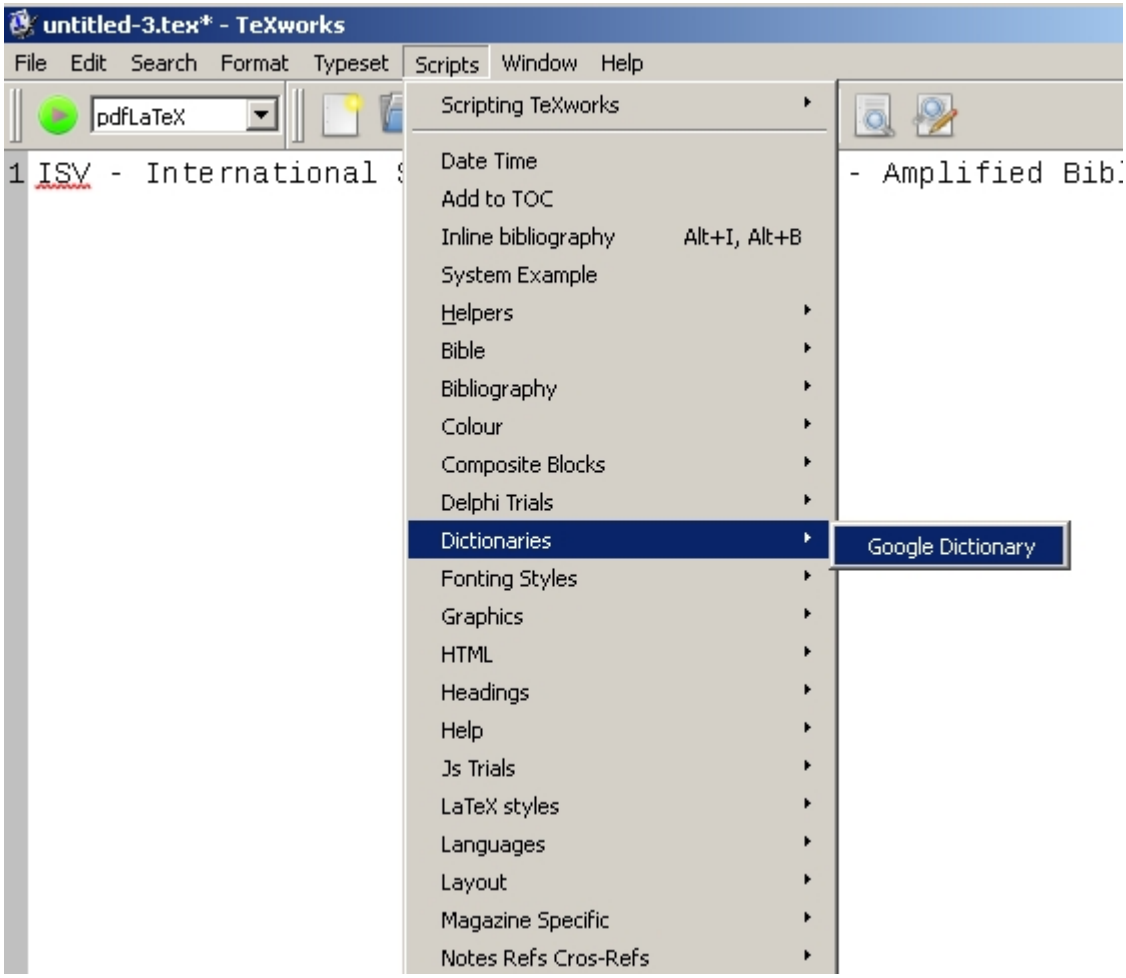
C:\Documents and Settings\Your Name\TeXworks\scripts

Under Linux and Mac that will be in your /home directory.

You can make sub-folders under the script directory, and place scripts in these. That will build a sub-menu tree on the Tw Menu://Script/ menu. The sub menus will use the text that you place in the Script header under, and the name of folder that you saved your script into.

//Title:





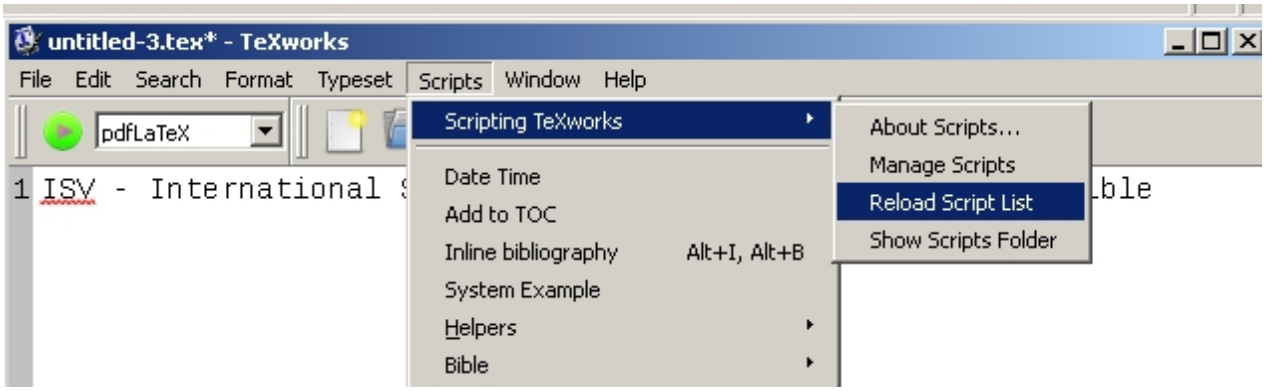
Contd ...

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

Scripts Management

Managing, and Loading New Scripts

When you download (and have checked!), or made a new script, and have placed it in the Tw script tree (see), if the Tw programme is running, you will need to tell it to reload the Scripts' List.



If the programme is not running, when it is next opened it will automatically find all scripts in the script folder tree, you will not have to do anything.

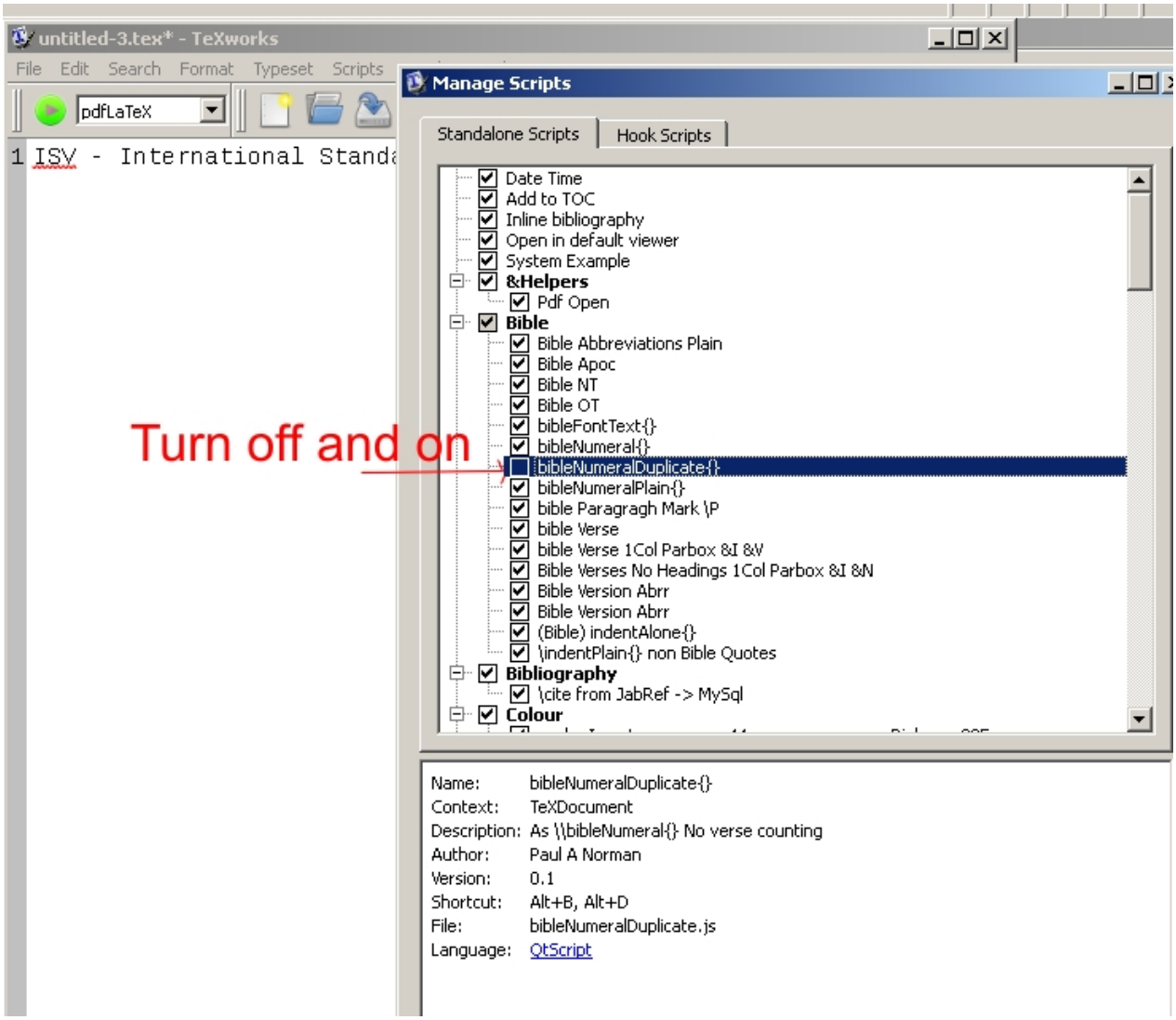
The other menu items there speak for themselves with the exception of Manage Scripts

With security concerns coming to the fore with QtScript needing to 'quietly' read and write files in background for its own purposes, keep an eye on this area, as you may need to specifically authorise each script before it can be used in future versions of Tw.

"Manage Scripts" also lets you see the brief description of each script put in by the script writer.

You can deactivate a script so that it will neither run or show in the Script menu tree.

ALso if you associate Tw to edit/open .js files, douple clicking a Script file in the Script Manager will open it in Tw for editing (see TeXworks as a QtScript editor).



Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Starting a Script

Using/Running Scripts

As we saw in the last picture there are two main types of scripts, with a third called libraries on the drawing board as well.

The two types we are concerned with here are "Standalone Scripts" and "Hooks".

Unless unticked in the Script Manager, standalone scripts attach themselves to the Tw Scripts menu according to their Title.

Lets look again at the debugger script. That will cause the debugger to show:-

// TeXworksScript
This first line tells Tw that this script is a QtScript not Python, Lua or anything else.

// Title: &Debug
This is the title that will appear on the Tw script menu The ampersand "&" indicates what letter would have an underline under it and so also be available as an Alt short cut key.

// Description: Starts the Debugger
This description will show in the Script Manager and might in a future release of Tw also show as a mouse over in the Scirpt menu. (http://code.google.com/p/texworks/issues/detail?id=17&q=hint#c17)

// Author: Jonathan Kew
As stated

// Version: 0.1
...

// Date: 2010-03-18
...

// Script-Type: standalone
This means that the user has to start the script, it will not be automatically started by Tw say when the Type Setting finishes, automatically started scripts are called hook scripts.

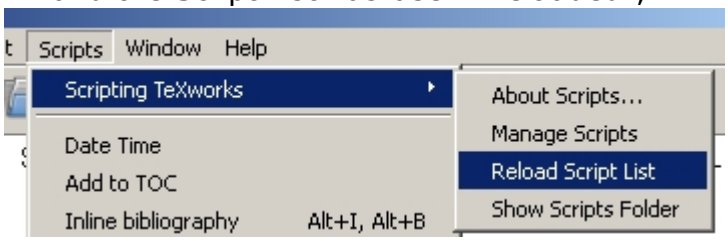
// Context: TeXDocument
If written in here (optional) this means that the script it can appear in the Scripts menu in an editor window, but won't appear in the PDF preview windows' Script menus.

// Shortcut: Alt+S, Alt+D
Keyboard shortcuts that can be used to start the script.

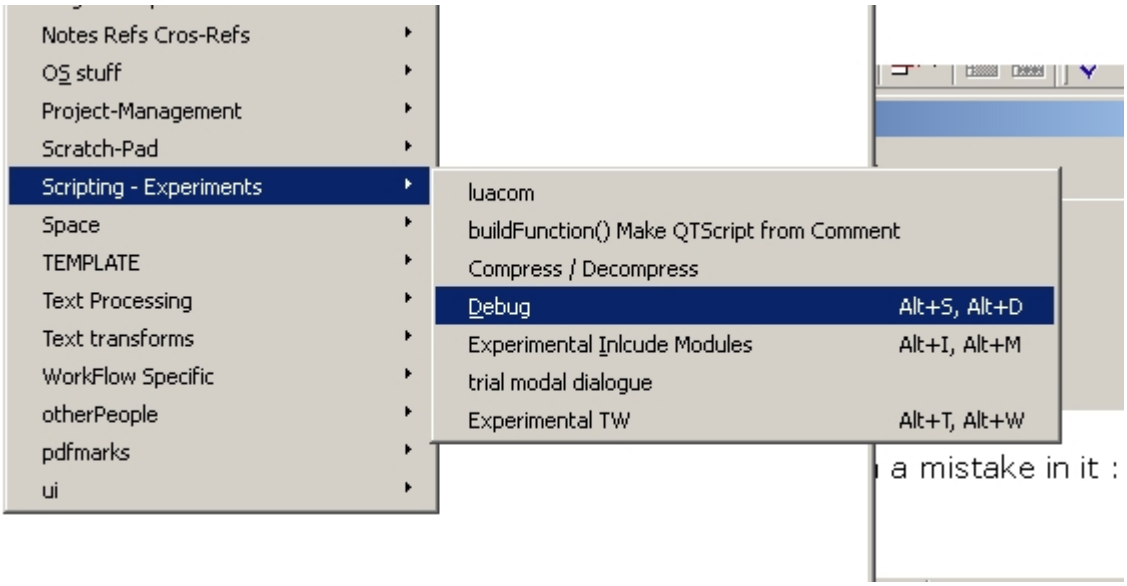
debugger;
Our command in QtScript. This is not a Tw api or QtScript command so it literally starts the debugger to tell you an error has happened.

Once this is saved, in my case in scripts/Scripting - Experiments/debugger.js

... and the Script List has been "Reloaded",



or the programme has been restarted,
then we can run it from the menu, or use its short cut keys (if some were added) from the key board.



To use the short keys, on Windows, holding down the Alt Key, tap the S and then the D key(s) before releasing the Alt Key.

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Showing the Application Programing Interface

Showing the Application Programming Interface

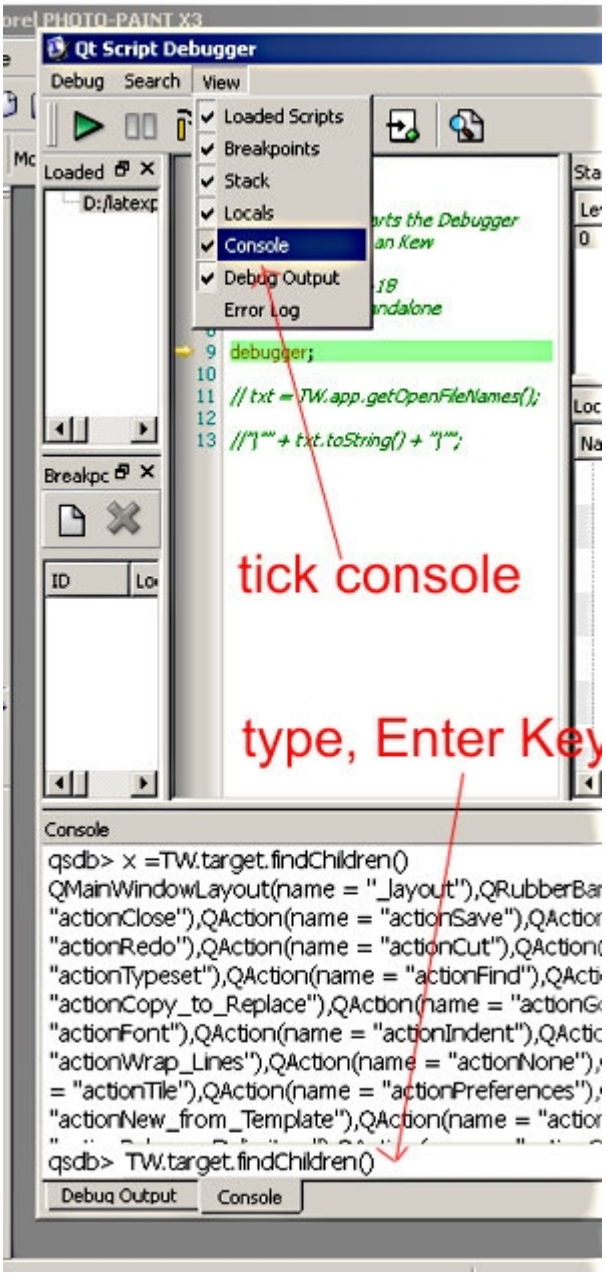
If that worked, you will now see the debugger.
(If it didn't open up, all I can suggest is that you retrace your steps through these notes.)

(More information on the debugger can be found here http://qt.developpez.com/doc/4.7/qtscriptdebugger-manual/
And in a video here http://www.youtube.com/watch?v=pyUKtOV9qn8.)

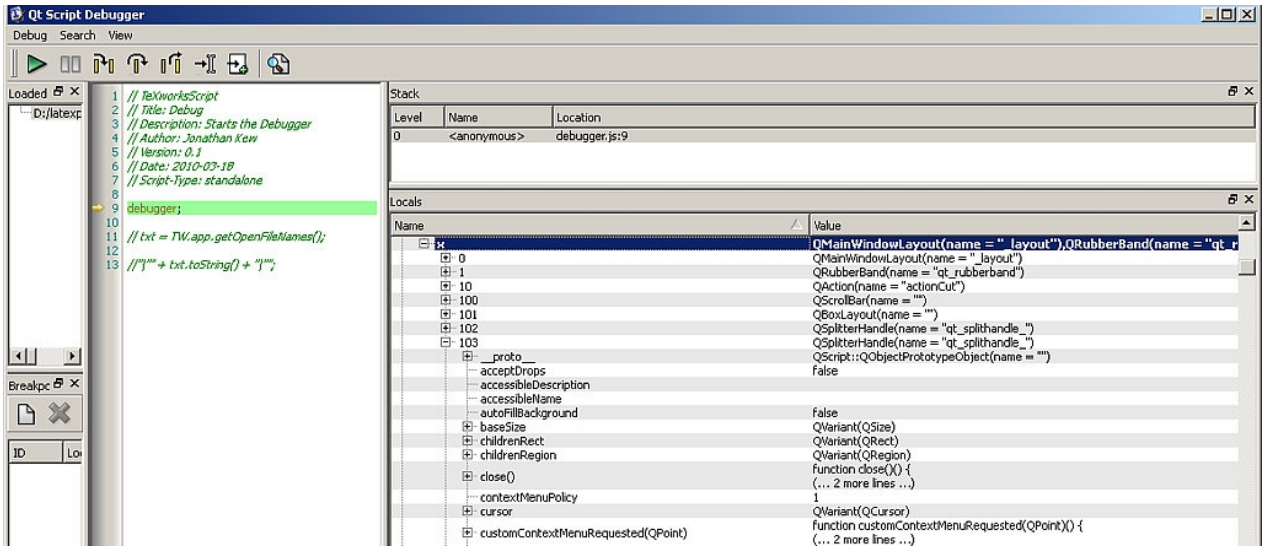
To see the whole object tree,

Run this in the Qt Script Debugger Console prompt as shown below:

```
qsdb> x =TW.target.findChildren();
```



And then look for "x" in the "Locals" pane in the debugger.



Only focus on things immediately below "x" that have a "name="Something" for if it has no name. you probably will not find a way of reliably addressing / accessing it.

You'll be able to read any of the actually used/available properties and functions. You can check their details in the Qt objects functions and properties on the Qt web site

In the Locals panel in the debugger. take note of the QtSomething name like:

QSplitterHandle(name = "qt_splithandle_") in the image above for item 103.

QSplitterHandle is what you would search for in the Qt documentation. Keep an eye on Qt version number, 4.4 is probably safe for general use according to postings here, but if you know that your version of your Tw release was built on a higher number of Qt, use that version's documentation (check Tw's Help/About dialogue - for example my help about says thta Tw 0.3 ver649 was built using Qt 4.6).

The higher Qt versions do introduce more functionality - but standard Tw releases and distributions may, as of November 2010, only use 4.4., although as of February 2011 4.7.1 is still being used.

http://doc.trolltech.com/4.4/qtscript.html

On the web page use the menu at top once you've read a bit about scripting and how it relates to this sort of tree ("x").

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

Making A Testing Script

Making A Testing Script

Make a Script as a general test script.

You can alter it, save it, and simply use its Alt short cut keys to run it each time you have made a saved change.

```
//TeXworksScript
//Title: Experimental TW
//Description: Exploration of TW objects
//Author: Paul A. Norman
//Version: 0.1
//Date: 2010-11-04
//Script-Type: standalone
//Context: TeXDocument
//Shortcut: Alt+T, Alt+W

// example

var mYchildrenRegion = TW.target.childrenRegion;
var howMany = mYchildrenRegion.numRects;
TW.information(null, "childrenRegion Rectangles", howMany);

null;
```

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

TeXworks as a QtScript editor

(All of this section is a work in progress!)

TeXworks as an IDE for QtScript Tw-Api

TeXworks is not designed for editing scripts as such, but has design features that may help people new to scripting, who are not used to using any other IDE (Interactive Development Environment) for scripting already.

You may find another editor more suited to your on-going scripting needs, but initially TeXworks is a surprisingly good place to start.

One of the decided advantages of using Tw as its own script editor, is that with many script it is quite useful to be able to execute them using their shortcut keys from within the script editor itself (see as an example.)

Another advantage of using Tw to edit scripts, is that when making user interfaces using Qt Creator, you can make a Tw script to get the Qt Creator interface to load the correct .ui, if it has a name based on the script you are currently editing, see Script/Interface Naming for more information.

Customisable features of TeXworks as a script editor include.

1. Coloured commands and functions
2. Tab Key based autocompletion.
3. Paired delimiters for matching/balancing
4. Tag window for function, var, const, with - list display

(See the following: texworks-config.txt syntax-patterns.txt Tab Key based auto-complete tag-patterns.txt set_syntax.js)

By altering the list of file types that Tw will filter in the File Open dialogue, common extensions for the Tw script plugins can be added. E.g.—

file-open-filter: Script documents (*.js *.qs *.py *.lua *.mod *.php *.bat *.txt *.csv)
file-open-filter: Graphics (*.jpg *.png *.pdf)

TeXworks lets you add coloured highlighting for common scripting editing needs. This also gives a common point of reference for explaining Tw scripting concepts.

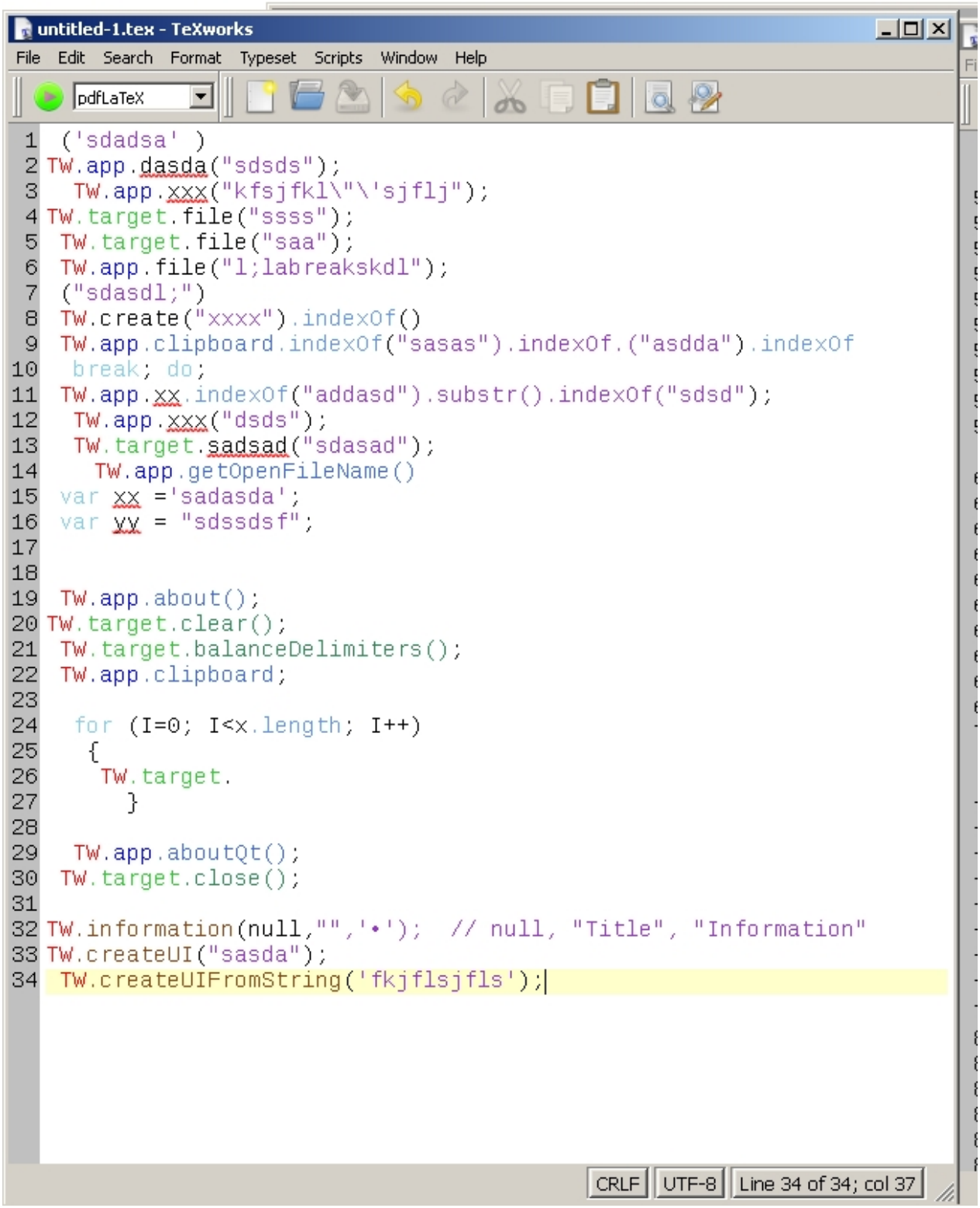
I'll be using that colour scheme in these helps.

Follow the instructions on this page `syntax-patterns.txt` if you want to add QtScript "syntax-pattern" colourings to your Tw editor.

The colour scheme means when you have correctly typed a known QtScript or Tw-Api object, function name, or property, it will change colour for you.

Correctly named functions and objects will be coloured when typed accurately as in the loose example below, while punctuation, brackets, and operators, or unknown and user functions you make yourself, will all generally be black.

Some non-functioning examples of this:–



Paired delimiters for matching/balancing

If you are between matching braces {} () [] these may be detected by using Crtl B on the keyboard, or Balance Delimiters on the Edit menu.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

texworks-config.txt

(Updated 2011 August, 1st)

Under your /config folder as a sibling directory to your scripts folder there is a folder called /config/configuration.

Locate the <TeXworks>/config/configuration folder on your system and edit texworks-config.txt to include this kind of list,
note the addition of "Script documents (*.qs *.js *.py *.lua)" just before All files (*.* *)

If your texworks-config.txt already has some of these, they will all need to be made active (remove the leading #). If you remove one # all have to be removed for the filter list. Or else only the fileter with their # removed will show in your Open dialogue box.

The #file-open-filter: Dont want these (*.xxx) example below illustrates how to remove one from the list.

Left un-altered (all with # in front of them), an application standard default list shows in the File Open dialogue box.

```
# To override the built-in list of file types in the File/Open dialog,
# uncomment and customize these entries:
#
file-open-filter: TeX documents (*.tex)
file-open-filter: LaTeX documents (*.ltx)
file-open-filter: BibTeX databases (*.bib)
file-open-filter: Style Class etc files (*.sty *.cls *.dtx)
#file-open-filter: Class files (*.cls)
#file-open-filter: Documented macros (*.dtx)
file-open-filter: Auxiliary files (*.aux *.toc *.lot *.lof *.nav *.out *.snm
*.ind *.idx *.bbl *.log)
file-open-filter: Text files (*.txt)
file-open-filter: HTML files (*.html *.htm *.xml *.ui)
file-open-filter: PDF documents (*.pdf)
#file-open-filter: Dont want these (*.xxx)
file-open-filter: Script documents (*.js *.qs *.py *.lua *.mod *.php *.bat *.
txt *.csv *.log)
file-open-filter: Graphics (*.jpg *.png *.pdf)
file-open-filter: All files (*.* *)
```

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

syntax-patterns.txt

(Updated 2011 August, 1st)

Under your /config folder as a sibling directory to your scripts folder there is a folder called /config/configuration.

In there you will find `syntax-patterns.txt` — make a back up of this file first, then edit `syntax-patterns.txt` adding the following at the very end of the existing file, and save it.

```
[QtScript]

red N ((\/\/\sScript\sBegins)|(\/\/\sScript\sEnds))

green Y       \/\/.*

green   Y  \/\*.*.*\*\/

purple N ([A-Z_0-9]{2,}+)

red N \s*(function\s*.*)\(

maroon N \((.*)\)

blue N \s((break)|(case)|(catch)|(continue)|(default)|(delete)|(do)|(else)|(false)|
(eval)|(for)|(function)|(if)|(in)|(instanceof)|(new)|(null)|(return)|(replace)|
(super)|(switch)|(this)|(throw)|(true)|(try)|(typeof)|(undefined)|(var)|(const)|
(while)|(with))

# EMCA standard functions properties
cornflowerblue N \.((indexOf)|(toLowerCase)|toUpperCase()|(lastIndexOf)|
(substr)|(length)|(split)|(join))

# additions via EMCA prototype didn't include Array.lastIndexOf()
crimson N \.((contains)|(toTitleCase))

# twPan related

indianred N ((\$tw)|(twPan)|(callingScriptPath\(\))|(citationType)|(insertCite)|
(citationOrder)|(citeOrderBy)|(bibleNamesAbrvs)|(global_load)|(create_helper)|
(alert)|(select)|(chooseFromDirectoryListing)|(prompt)|(confirm)|
(file_get_contents)|(file_put_contents)|(getRelativePath)|(initialise)|(finalise)|
(getBibleVersion)|(getBibleVersionName)|(getBibleVersionAbrv)|(osCmd)|
(getBibleVersionFull)|(getSuffixDateTh)|(insertSuperscriptDateTh)|(getDateWords)|
(getDate)|(trim)|(ltrim)|(rtrim)|(LineBreakIs)|(convertSpeechMarks)|
(cleanParaMarks)|(LineBreakIs)|(findFromScript)|(escapeLatex)|(pasteToLatex)|
(latexFileName)|(responseObject)|(getSnippet)|
(dataRecords_ArrayOfObjectsof_fieldName_And_fieldValue)|(commandUsed)|
(toTitleCase)|(getArrayIndexKeyFromValue)|(cleanLineEnd)|(emptyString))

red N ((\.abbrv)(\.full))

# PhpJs related

limegreen N PhpJs\..*.\s

limegreen N $P\..*.\s

# global constants
```

```
darkorange N ((twConst)|(SystemAccess_NotAttempted)|(SystemAccess_OK)|
(SystemAccess_Failed)|(SystemAccess_PermissionDenied))

# msgBox button return constants

tomato N msgBox\.((Ok)|(Open)|(Save)|(Cancel)|(Close)|(Discard)|(Apply)|
(Reset)|(RestoreDefaults)|(Help)|(SaveAll)|(Yes)|(YesToAll)|(No)|(NoToAll)|
(Abort)|(Retry)|(Ignore)|(NoButton))


red N (TW)

blue N  \.app

# TW.app. //functions properties
royalblue N  \.((about\(\))|(aboutQt\(\))|()|(hasGlobal)|(getGlobal)|(setGlobal)|
(clipboard)|(about\(\))|(aboutQt\(\))|(aboutToQuit\(\))|(applyTranslation)|
(applicationName)|(applicationVersion)|(getOpenFileName)|(getOpenFileNames)|
(scriptListChanged\(\))|(openFileFromScript)|(openHelpFile)|(closeAllWindows
\(\))|(deleteLater\(\))|(doResourcesDialog\(\))|(keyboardInputInterval)|
(layoutDirection)|maybeQuit\(\)|(newFile\(\))|(openFileFromScript)|
(openHelpFile)|(organizationDomain)|(organizationName)|(quit)|
(quitOnLastWindowClosed)|(closeAllWindows\(\))|(setStyleSheet\(\))|
(showScriptsFolder\(\))|(updateScriptsList\(\))|(syncPdf\(\))|())

limegreen N  \.target

# TW.target functions properties
seagreen  N  ((hasGlobal)|(getGlobal)|(setGlobal)|(balanceDelimiters\(\))|(clear
\(\))|(close\(\))|(\.insertText)|(selection)|(selectionStart)|(selectRange)|
(copyToFind)|(doFindDialog)|(fileName))

# show quoted material
darkorchid Y (["'])(?:.*)(\1)
 # \1 requires exact matching quote mark - spell checking on

chocolate N  ((system)|(writeFile)|(readFile)|(information)|(warning)|(question)|
(critical)|(getDouble)|(getInt)|(getItem)|(getText)|(createUI)|
(createUIFromString)|(deleteLater\(\))|(platform\(\)))

# coral N \..*\(

black N [;,=.+]

[ConTeXt]
# special characters
darkred              N      [$#^_{}&]

# start/stop
darkgreen    N      \\(?:start|stop)[A-Za-z]+

# control sequences
blue           N      \\(?:[A-Za-z@]+|.)
```

```
# comments
gray                    Y       %.*

# other possibilities to be added....
# [BibTeX]
# [Metapost]
# etc

[HTML]

chocolate n class=.*"\s
MediumBlue;B n <[bBrRhH]{2}>

#http://www.webdeveloper.com/forum/showthread.php?t=175987
Gold n &[^\s]*;
darkgoldenrod n \d*pt

IndianRed/LightGray;B n ^[^\s]*:

Red/LightYellow  n (\[)|(\])

SeaGreen/LightCyan  n (\{)|(\})

 #D7700A  n .*\.(jpg|JPG)
 #D7700A  n .*\.(png|PNG)

Black/AliceBlue;I Y <(I|i)>[a-zA-Z0-9\s]*</\1>
Black/AliceBlue;B Y <(B|b)>[a-zA-Z0-9\s]*</\1>
Black/AliceBlue;U Y <(U|u)>[a-zA-Z0-9\s]*</\1>

SlateBlue/LightGoldenrod;I n <[a-zA-Z]*>
SlateBlue/LightGoldenrod;I n <[/a-zA-Z]*\s*
SlateBlue/LightGoldenrod;I n >

goldenrod;U n (\s*rgb\(.*\))

# show style quoted material
darkorchid n style=(["'])[^\s]*(\:)
darkorchid n (["'])
# \1 requires exact matching quote mark - spell checking on
```

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

Tab Key based auto-complete

(Updated 2010 December, 19th)

Tab Key based Auto-Complete

tw-api.txt in config/completion/

```
%%!TEX encoding = UTF-8 Unicode
%#INS# #RET# •
TW.readFile("#INS#");
TW.writeFile("#INS#", •); // pathFileName, Contents (- utf-8) #RET#
TW.information(null,"#INS#","•");  // null, "Title", "Information"
TW.warning(null,"#INS#","•");  // null, "Title", "Warning"
TW.question(null,"#INS#","•", ( msgBox.Yes | msgBox.No ) , msgBox.Yes  );  // null, "Title Here",
"Question", [ buttons (Yes/No, [default Yes ] ]
% TW.question( null, "Use Immeadialtely?", "Use This Text Where You Are" ,( msgBox.Yes | msgBox.
No ) , msgBox.Yes  )
TW.question( null, "title", "dislaytext" , buttons , default button   ) // e.g. msgBox.Yes
TW.critical( null, "title", "displaytext" , buttons , default button  )
TW.app.getOpenFileName();
TW.app.getOpenFileNames();
TW.app.getSaveFileName();
TW.target.selectionStart
twConst.SystemAccess_NotAttempted
twConst.SystemAccess_PermissionDenied
twConst.SystemAccess_Failed
twConst.SystemAccess_OK
msgBox.Ok
msgBox.Open
msgBox.Save
msgBox.Cancel
msgBox.Close
msgBox.Discard
msgBox.Apply
msgBox.Reset
msgBox.RestoreDefaults
msgBox.Help
msgBox.SaveAll
msgBox.Yes
msgBox.YesToAll
msgBox.No
msgBox.NoToAll
msgBox.Abort
msgBox.Retry
msgBox.Ignore
msgBox.NoButton
twPan.callingScriptPath(__FILE__)
twPan.citationType
twPan.insertCite
twPan.citationOrder
twPan.citeOrderBy
twPan.bibleNamesAbrvs()
twPan.os
twPan.initialise();
twPan.select("#INS#", ["optionsArray"])
```

```
twPan.prompt("#INS#");
twPan.confirm("#INS#");
twPan.alert("#INS#");
twPan.select("#INS#", ["optionsArray"])
twPan.file_get_contents("#INS#");
twPan.file_put_contents("#INS#", "");
twPan.getRelativePath("#INS#", "");
twPan.finalise();
twPan.getBibleVersion(#INS#)
twPan.getBibleVersionName()
twPan.getBibleVersionAbrv()
twPan.getBibleVersionFull(#INS#)
twPan.getDateWords
twPan.getDate
twPan.trim(#INS#)
twPan.ltrim(#INS#)
twPan.rtrim(#INS#)
twPan.convertSpeechMarks("#INS#");
twPan.cleanParaMarks("#INS#");
twPan.osCmd("cmd /c #INS#", true);
twPan.findFromScript("#INS#");
twPan.escapeLatex("#INS#");
twPan.getSnippet(•database, •table, •id, •chosenColumn(,s));
responseObject=twPan.getSnippet(•database, •table, •id, •chosenColumn(,s));
responseObject.dataRecords_ArrayOfObjectsof_fieldName_And_fieldValue;
responseObject.commandUsed;
twPan.toTitleCase("#INS#")
twPan.getArrayIndexKeyFromValue(•myArray, •value);
twPan.cleanLineEnd(#INS#);
twPan.emptyString(#INS#)
% http://henry3:8181/reference/
PhpJs.("#INS#")
PhpJs.abs(#INS#)
PhpJs.acos(#INS#)
PhpJs.acosh(#INS#)
PhpJs.addslashes("#INS#")
PhpJs.array_change_key_case(#INS#); //  Either ,CASE_UPPER or ,CASE_LOWER (default)
PhpJs.array_chunk("#INS#"array $input , int $size [, bool $preserve_keys = false ] ) //
PhpJs.array_combine(#INS# array $keys , array $values )
PhpJs.array_count_values(#INS#)
PhpJs.array_diff(#INS# array $array1 , array $array2 [, array $ ... ] );
PhpJs.array_diff_assoc(#INS# array $array1 , array $array2 [, array $ ... ] ); //Unlike array_diff() the
array keys are used in the comparison.
PhpJs.array_diff_key(#INS# array $array1 , array $array2 [, array $ ... ] );  // This function is like
array_diff() except the comparison is done on the keys instead of the values.
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
```

PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")

PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")

PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")

PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")

PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")
PhpJs.("#INS#")

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

tag-patterns.txt

(Updated 2011 August, 1st)

tag-patterns.txt -- add these

```
#
0 \\(?:begin|end)\{document\}
1 \\(?:clearpage|newpage)
#0 \\label\{(.+.)\}
4 (\\label\{.+.)\}
5 \\bibleNumeral\{(\d?.*[a-zA-Z].+.\s).+\s([A-Z]{3})\}
#
# QtScript comment single line must be started //%: to be shown in tag tree
0 \s*//%:\s*(.+)
# QtScript function name
1 (\s*function\s.+)\(
```

```
# QtScript variable name
1 (\s*var\s.+)
# QtScript const name
1 (\s*const\s.+)
# QtScript with block
1 (\s*with\.*\)\s(.+)
```

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

set_syntax.js

```
// TeXworksScript
// Title: Syntax for Scripting
// Description: Colourises  .js .qs .mod files
// Author: Paul A Norman
// Version: 0.3
// Date: 2010-12-20
// Script-Type: hook
// Hook: LoadFile



 var  fileName = TW.target.fileName;

  var   extPos = fileName.lastIndexOf('.') +1;

 if (extPos > 0)
  {
   var ext = fileName.substr(extPos);

   // updated 2011 08 01

// N.B. that the sections HTML QtScrpt and anything you add will need to already exist in your syntax-patterns.txt or be deactivated here - error messages will follow otherwise

   switch(ext)
        {

        case 'html':
        case  'htm':
        case  'xml':
         TW.target.setSyntaxColoringMode('HTML');
        break;

        case   'qs':
        case   'js':
        case 'mod':
        case  'php':
         TW.target.setSyntaxColoringMode('QtScript');
        break;
```

```
        default:
         TW.target.setSyntaxColoringMode('LaTeX');
        break;
        }

  }  //End. if (extPos > 0)
```

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Script Authoring

This script walks you through creating a script for QtScript, it focuses on the Header part, which can otherwise seem a little baffling at first.

Copy it into a sub-directory under config/scripts/
Update Scripts, and run.

(Fist Make a New Blank Document and position the editing cursor in it.)

You can use this, then copy to your script editor of choice or alter it to place the result on the clipboard, instead of inserting into a Tw document.

... For script listing (2011 07 23) see TW.app.updateScriptsList()

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

Standard Standalone Script

Standalone Scripts

```
// TeXworksScript
// Title:  Script Menu Title Here
// Description: This is an example Script
// Author:  Your Name here
// Version: 0.1
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+X, Alt+X



// Title:  Script Menu Title Here

        This will appear on your Script menu according to the subdirectory that your
        script is saved into.

// Version - is your own made up number
        and has nothing to do with the version of Tw that you are using or intending
        the Script for.

//Context can be either of:-

    // Context: TeXDocument

    or

    // Context: PDFDocument
```

TeXDocument is the editor, and PDFDocument is the pdf previewer.

Created with the Personal Edition of HelpNDoc: Free help authoring environment

Automatic Hook Scripts

Hook Scripts

```
// TeXworksScript
// Title: LaTeX errors
// Description: Looks for errors in the LaTeX terminal output
// Author: Jonathan Kew & Stefan Löffler
// Version: 0.4
// Date: 2010-11-02
// Script-Type: hook
// Hook: AfterTypeset
```

This is the header from a script that, well does what it says above!

Hook scripts do not appear in the Script menu and have their own Tab in the Script manager.

The two main areas that change it form a standard 'Stand Alone' script that the User makes run, is that it is triggered into running by something that the TeXworks programme has just done.

Here is an example of a hook script [setSyntaxColoringMode(QString)]

```
// Script-Type: hook

// Hook: AfterTypeset
```

So you really need to be careful with what you get hook scripts to do - they will just kick in when the Hook that you have specified triggers, with out any automatic question to the User.

Available hooks are at present (2010 December):—

`TeXworksLaunched`

This triggers when TeXworks has completely initialise at its startup

`NewFile`

Under default settings, NewFile will happen each time Tw is opened for first use .

`NewFromTemplate`

`LoadFile`

`AfterTypeset`

At present you can only use one hook per script.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

UI Dialogue/Form Scripts

(All of this section is a work in progress!)
June 22nd 2011

This is based on just making a few forms for TeXworks' Qt Scripts, web gleanings, the Qt site and various forums, and TeXworks specific information received (gratefully) from **Jonathan Kew** and **Stefan Löffler**.

Forms / Dialogues

Introduction —

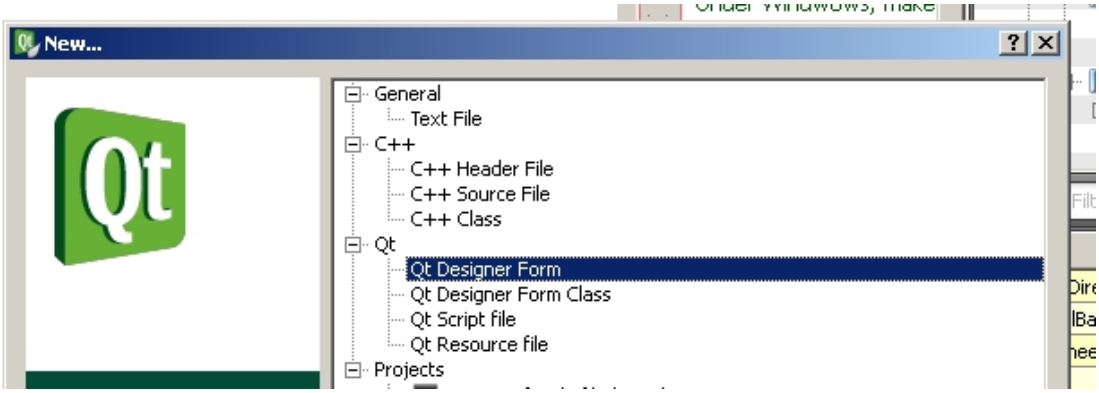
(Example below - Putting it all together)

References here to objects and functions commencing twPan refer to a wrapper object (not required to work with Qt forms) helper_twPan.mod described a bit at present in User 'Library' Modules and helper objects

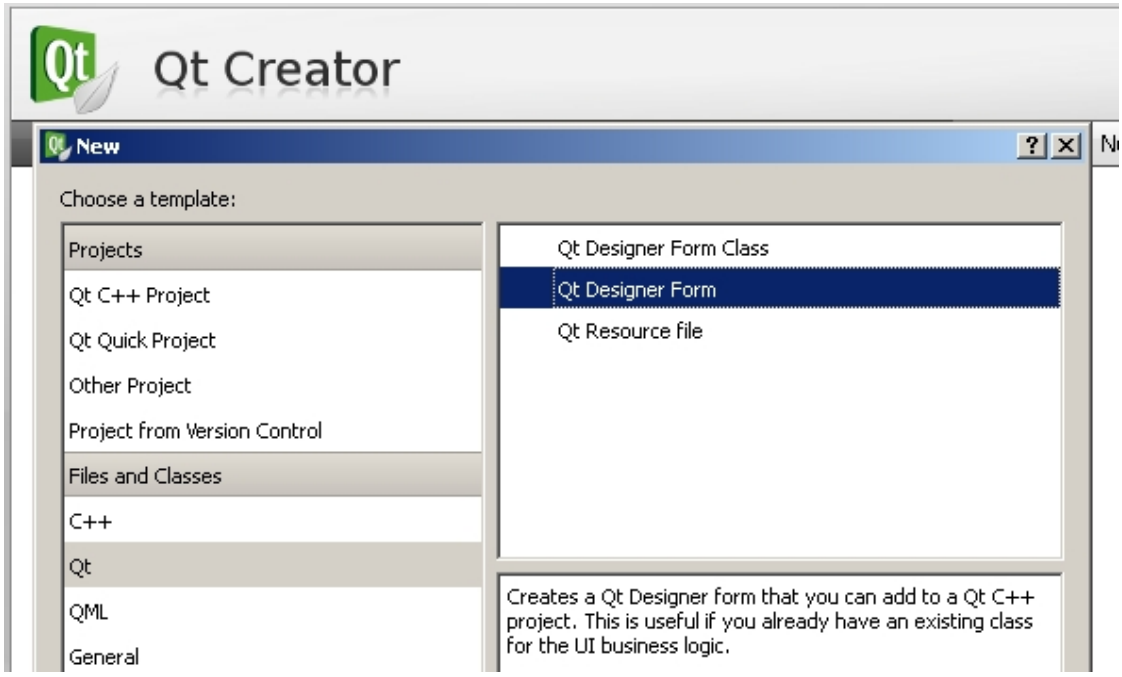
Qt Creator for Form Design

Qt provides a basically free WYSIWYG User Interface editor called Qt Creator http://qt.nokia.com/products/developer-tools/, that can be used to design forms / dialogues (saved as an xml file with .ui extension) that can be linked to your Qt .js scripts *at Script run time*.

You can start off directly from File/New and save your form into your TexWorks script sub-directory.



You can also make your Script File in Version 1 series of Qt Creator, if not using TeXworks,
or another scripting editor (use Qt Script File above).



In 2.1 series of Qt Creator the same thing now looks like this, and possibly no direct provision is now made for Qt Script Files.

In your TeXworks script, you can read some properties, connect to some events (signals), and call some functions (slots) of the Qt objects contained in your .ui form(s). It is a bit hit and miss to find out what works, as there does not yet appear to be a definitive Qt list of what parts, of which Qt standard widgets, are scriptable at present. Words in Qt documentation like "public" don't necessarily mean that what is being referred to is available to script.

There is no way of avoiding reading various pieces of Qt documentation and becoming familiar with Qt's terms and vocabulary, it is actually quite well written in a fairly consistent manner, and is well worth the effort.

To use form objects (widgets), you need to connect *in script* to the form's objects, and also separately make connections to any of the form's events (signals) that you want to respond to, like `.clicked` `.textChanged` and so on. Later, before the script finalises, you need to "disconnect" from these "signals". This is done in the script editor, and executed at script run time, and not at the moment in the Qt Creator form editor.

You can also use the Qt script debugger to help, there is a video here http://www.youtube.com/watch?v=pyUKtOV9qn8.

Detail

In script, when you create your form (dialogue), you assign the form to a QtScript variable.

Later when you have finished with the form, *before* your Script closes, you must mark it for removal from memory.

You do this by calling a Qt QDialog function, deleteLater() (see: http://tug.org/mailman/

htdig/texworks/2011q2/004428.html)

```
  var myDialogue = TW.createUI(__FILE__.replace(".js",".ui"));
                   // or form an xml string  TW.createUIFromString
(QString)

// do some stuff


// when dialogue is no longer required at all, but in any event before
script finishes ...

myDialogue.deleteLater();
```

In the above example, the form's xml is in a file by the same name, in the same folder, as the script that is going to use it, but with a .ui extension instead of a ,js extension.
So if you use this naming method, and have say myScript.js and myScript,ui in the same directory, then make yourself a helper script to open the Qt Creator from inside your .js while you are editing it in TeXworks. For example here is a Windows standard while using a standard installation of Qt Creator 1.2 activated by, Alt(OQC) (change as needed) —

```
// TeXworksScript
// Title: Open Qt Creator on File
// Description: Opens Qt dialogue builder using current Script's Name
// Author: Paul Norman
// Version: 0.1
// Date: 2011-07-23
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+O, Alt+Q, Alt+C

eval(TW.app.getGlobal("helper_twPan")); // Comment if NOT Needed - This includes PhpJs
($P), twConst, msgBox, twPan ($tw)

var uiFileName = TW.target.selection;

if (uiFileName === "")
    {
    //dialogue .ui name is in same directory as script being edited, and has same stem name--

        uiFileName = TW.target.fileName.replace(".js",".ui") ;
    }
else
{
//dialogue .ui name is in same directory as script being edited, and is highlighted in Tw editor as
in:
// showMoreDialogue = TW.createUI(scriptPath + "showMore.ui");

        uiFileName = TW.target.fileName.substr(0, TW.target.fileName.lastIndexOf("/") +1) +
uiFileName;
}

var retVal = TW.system("cmd /c start \"G:/Qt/qtcreator-2.1.0/bin/qtcreator.exe\" \"" +
```

```
        uiFileName +"\"");

        retVal = null;
```

As you might write code that uses User input to create a form (see below) in a function using one of createUIFromString(QString,QWidget*) or more generally createUIFromString(QString), and need to call it later, it may be worth adopting a set approach to the whole thing and always start by declaring all the variable names you are going to name forms by, at the top of you script (in global space), and put them in an array at the same time.

Then later either at the end of your main run before the Script closes, iterate through the array, and test for whether the variable was ever created as a form, and if so call deleteLater() on it.

Perhapos even create and hide (as shown below here) all your dialogues at once at the beginning of your script. There are other possibilities but this looks for the moment to be fail safe, with the draw back only of a one time unprofessional flicker when created and hidden.

```
   var myFirstForm = null; // main form
   var mySecondForm = null;
   var myThirdForm = null;

   var myForms = [];

// any time later or straight a way

   myFirstForm = TW.createUI("a/path/to/myFirstForm.ui");
   myForms.push(myFirstForm);
```

You can then hide it straight away if you wish.

```
 mySecondForm = TW.createUI("a/path/to/mySecondForm.ui");
 myForms.push(mySecondForm);

  mySecondForm.setVisible(false); // there will be a momentary flicker

// And so on.
```

While the script is running, any dialogue/form that has been created and has not already been .deleteLater() -ed can be called (again) using

```
   var  answer = myForm.exec();
```

And it will show modally (see below).

Then at the end of the script run something like:—

```
 for (var x = myForms.length -1; x > -1; x--)
{ // done reverse to destroy main form last
        {
              if (myForms[x] !== null)
                {
```

```
            myForms[x].deleteLater();
           }
        }
```

"Finished"

There is a signal "finished" emitted by the form when it closes.

If you consider the first form that you show when the script runs your 'main' form (i.e. it is around for the whole of you script while other forms/dialogues may come and go),
you could connect into that signal as part of `UIconnectVarious()` (below `UIconnectVarious("connect"|"disconnect")`)

```
eval("main_form.finished." + connectAction + "(main_form_finished)");
```

and put your end of script (finalization) stuff in the function you connect to that signal.

```
function main_form_finished(response)
       {

        // response will be as shown below: Cancel, or Ok  0, or 1

       UIconnectVarious("disconnect") // see below UIconnectVarious
("connect"|"disconnect")

       //  save settings to disk or do anything else need at the end of the script

    for (var x = myForms.length -1; x > -1; x--)
{ // done reverse to destroy main form last
       {
              if (myForms[x] !== null)
               {
                myForms[x].deleteLater();
               }
        }
```

How ever you do it, it needs to be done!

Once Created

Once created, you can then use dot notation to walk through the form's objects. As an object's immediate parent is related to the design object (widget) it is on, this can make the dot notation method a bit complicated.

```
        myDialogue.niceLookingPanel.NiceFrame.MyObject.text = "Hello";
```

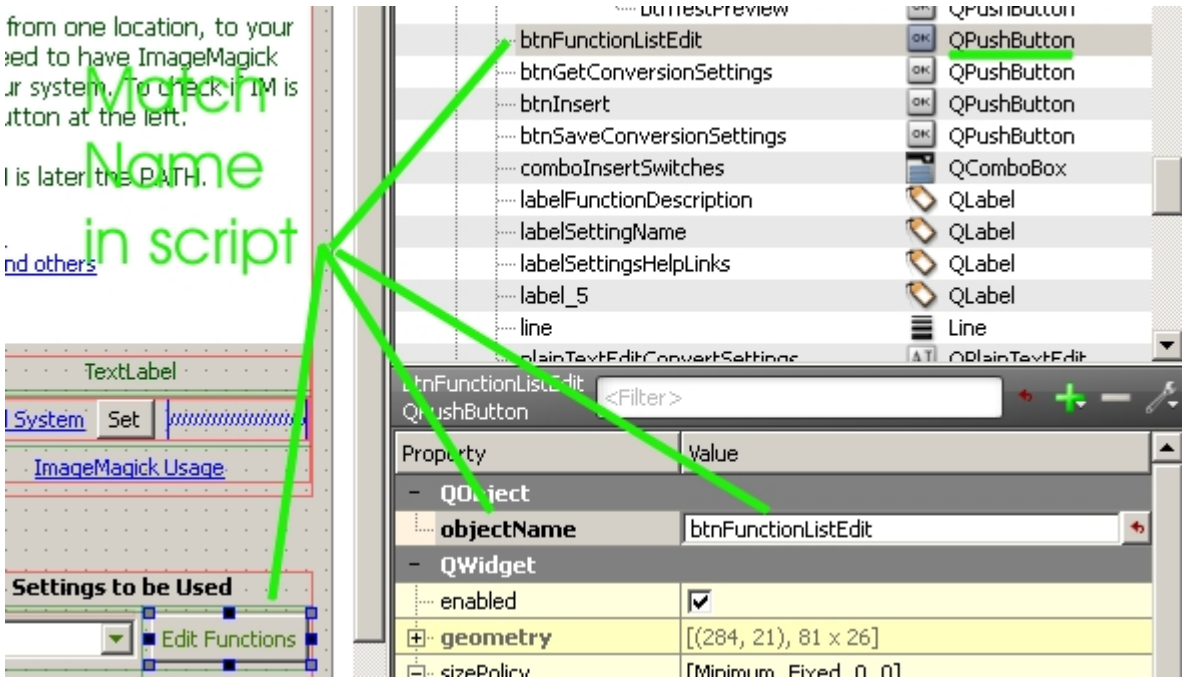
There are at present some widgets that you have to completely build in the Qt form Creator, to which you can *not* add items later, the QComboBox is one example. You can however use a function in your script to pre-load the .ui xml file that will build the form (as plain text), reprocess it, and in script pass the altered xml to a top level TW function as a string -

```
var myDialogue =  TW.createUIFromString("<My alterd> <XML / ...>";
```

The string may be contained in a script variable or script function that you write. You can write a whole form this way from scratch (just look at a simple form's disk file in a text editor to get the formats required).

As said before, .dot paths like: myDialogue.niceLookingPanel.NiceFrame.MyObject.text = "Hello"; can be hard to keep track of.

A simple method is to simply assign the same names in script, to the form's objects' names, that you want to work with.



When scripting a 'widget' (form object), google "Qt QPushButton" for example, and/or go directly to http://doc.qt.nokia.com/4.7/index.html **and pages like** http://doc.qt.nokia.com/4.7/**qpushbutton.html** **for more programming information.**

There is a top level function in TwScript that can find form objects, TW.findChildWidget... and obtain references for scripting to the form's objects directly, ignoring how they are positioned on the form. You pass the Var name (as a script object) that you gave the form in script when you created it (this can be different from the form's own name in Qt Creator, which is 'dialog' by default), and the name of the widget you are trying to find as a string (examples below).

There are a number of approaches of exposing the form's objects, that you need, to script. Here is one simple method.

Create an array of object names (as strings), and then iterate through them, assigning the names as objects in script. This requires that the widgets (labels, textboxes etc ..) on the form have the same names as you intend to use for them in script, although you can creatively devise strategies to alter this requirement.

Here is a direct approach I use, if for no other reason than it keeps all the names I want in front of me in the script editor, while scripting.

Only enter the names of form objects that you need.

```
 var scriptWidgets = ["showImageInformation", "radioPerrcentage",
"radioWidth", "radioHeight",
                                          "editSize",  "editDpi",
"checkSaveDirectory",
                                       "plainTextEditConvertSettings",
"btnGetAnotherImage", ... etc
                    ];
```

Then later after creating/loading the form :—

```
for (widget in scriptWidgets)
{
// Define required  dialogue .ui widgets here in QtScript as
programmable objects

eval("var " + scriptWidgets[widget] + " = TW.findChildWidget(myDialogue,
\"" + scriptWidgets[widget] + "\")");

}
```

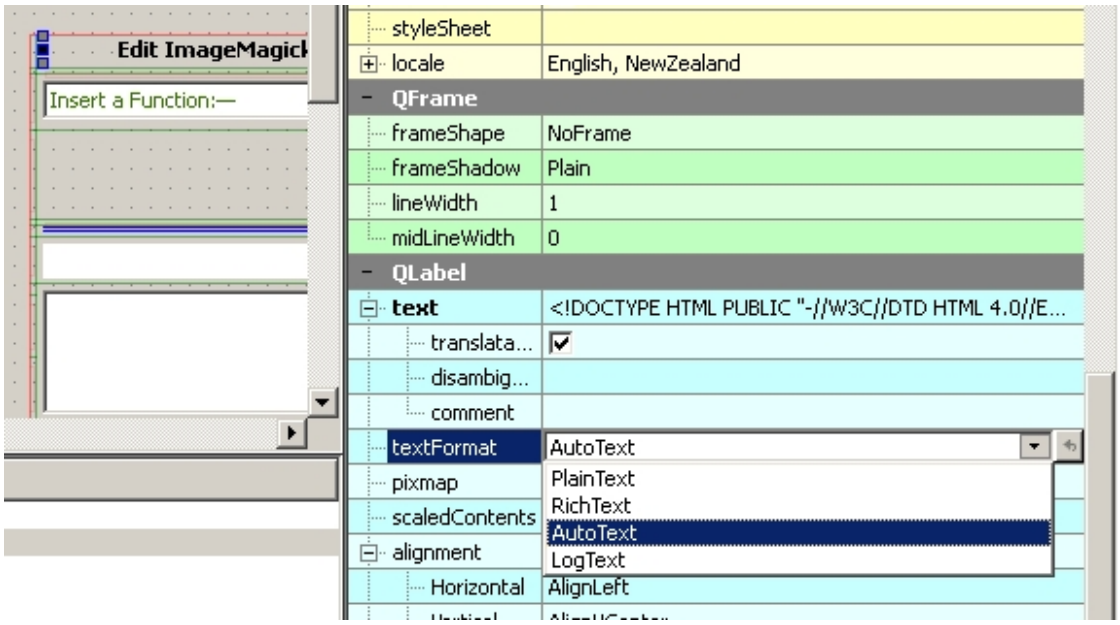
For scope reasons, this needs to be done at top level, and not in a function call.

You can then directly refer to a "found" form object's properties and functions (or at least to the ones that Qt exposes by default to scripting).

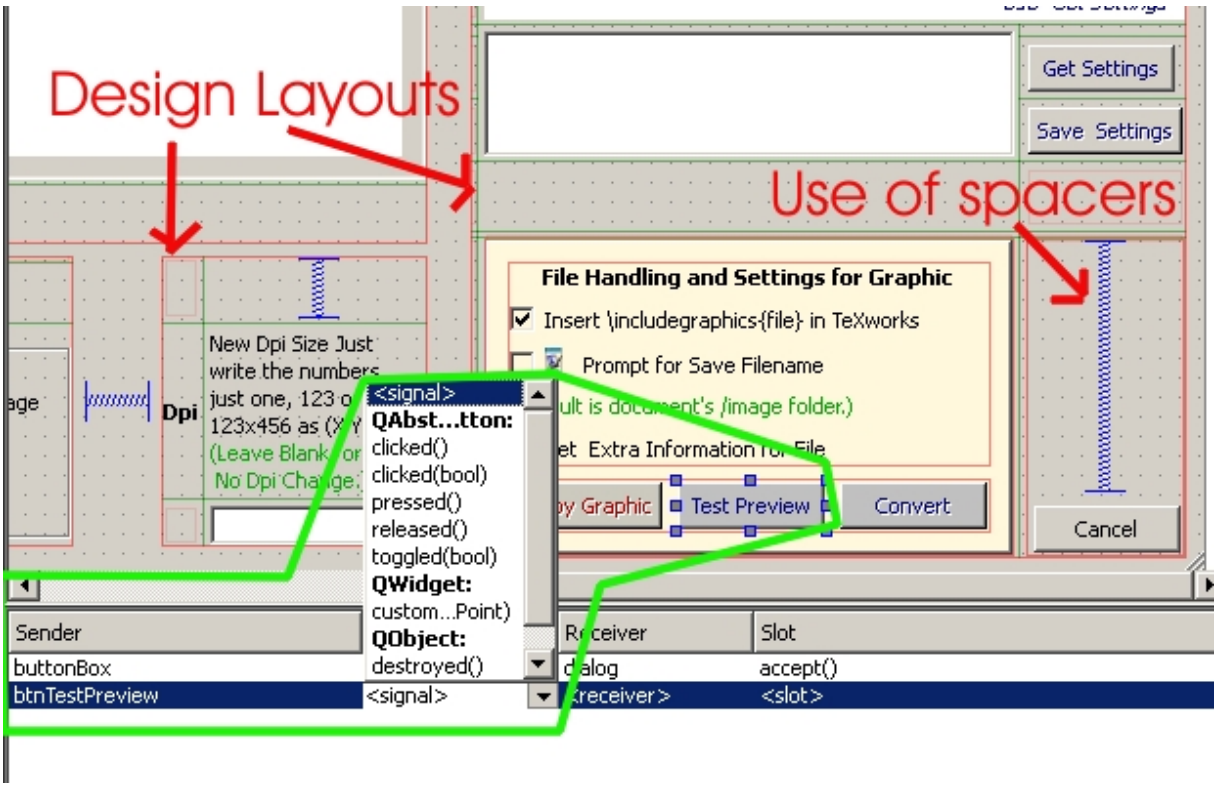
```
myDisplayAtrea.setHtml("<span style=\"color:navy; background-color:rgb
(20,45,77)\">Hi TeXworks User</span>");


myLabel.text = "Show This";

myLabel.text = "Show <span style=\"font-weight:bold\">This</span>"; //
```

To show HTML based text, you'll need make sure that the textFormat of the widget is set either to AutoText (detection) or explicitly to RichText.

Qt uses a subset of HTML to show rich text, see http://doc.qt.nokia.com/latest/richtext-html-subset.html



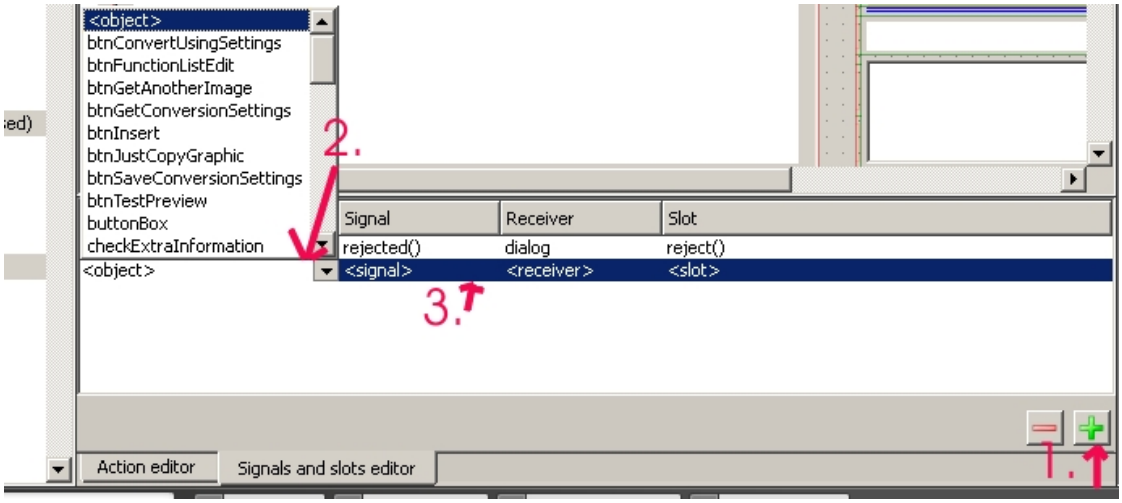
When designing a Qt form or dialogue, you'll need to get use to using Design Layouts to make
sure your material works and looks right if resized, and on different operating systems
and their versions. See http://doc.qt.nokia.com/4.7/designer-manual.html
http://doc.qt.nokia.com/latest/designer-layouts.html
http://doc.qt.nokia.com/4.7-snapshot/widgets-and-layouts.html **and**
http://doc.qt.nokia.com/latest/layout.html



Once the script's side of variables have been created to represent form objects, and linked to in the script's global space, any of the form objects that have events (signals) that you want to 'listen' for, can be "connected" to. While in Qt Creator you can view the name of a widget's possibly available signals (and slots), in the **bottom tab** called 'Signals and Slots Editor' (shown above).

Steps:

Choose the Green plus sign + (**1.** below) .
Then choose your form object (widget) **2.** and then the available signals for it will show in the signals dropdown **3.** as shown in the bottom of the green area above.



As said before, widgets (form objects) you are 'listening to', have to be "disconnected" prior to the script finishing. This necessitates that you call the form's (dialogue's) .exec() function after having created the form, to keep the script held running, while the dialogue shows in a modal fashion. Otherwise the script would just run straight through and effectively disconnect the form's objects from User interaction.

So using .exec(), halts the script's primary linear execution, but allows functions that have been connected to signals in the form, to operate effectively. When the User closes the form, linear script execution continues form the point immediately after .exec(), as shown in the example below Putting it All Together

When the form has .exec() called on it in script, when the User closes the form, it will return one of two values:

	Integer Returned
QDialog::Accepted (e.g. OK)	1
QDialog::Rejected (e.g. Cancel)	0

These, `Accepted or Rejected` can be set on any form widget addressing the form object itself (default name "dialog" in Slot column) in the Signals and Slots Editor in Qt Creator.

See http://doc.qt.nokia.com/latest/qdialog.html#exec and http://doc.qt.nokia.com/latest/qdialog.html#DialogCode-enum

To easily enable initialising and finalisation of events (signals), a function something like this can be written—

```
function UIconnectVarious(connectAction)
	{
	/* This streamlines initialisation and finalisation of the script
	dialogue
	slot connections connectAction passes either 'connect' or 'disconnect'
	.slotEvent is something like .clicked
	*/
	// eval("nameofWidget.slotEvent." + connectAction +
	"(functionYouWriteInThisFile)");
	                  //                    functionYouWriteInThisFile - !
	make function in Global Space!
```

```
eval("btnGetAnotherImage.clicked." + connectAction + "(showImage)");

eval("plainTextEditConvertSettings.textChanged." + connectAction +
"(plainTextEditConvertSettings_textChanged)");

eval("comboInsertSwitches['currentIndexChanged(int)']." + connectAction
+ "(comboInsertSwitches_SwitchChosen)");
}
```

When called, connectAction will be either "connect" and "disconnect", run at the begging of things, and before the script finishes up (after .exec() see also "Finished").

Events for objects will sometimes return values, but afaik, *never the sender object itself.* So this in a function connected to a form object's signal, never refers to the form object / widget itself, but to the script's Global this.

If there is more than one possible returned 'thing' for an event (an overload) you need to use the style marked above `comboInsertSwitches['currentIndexChanged(int)']`..

Check the Qt documentation for the object (widget) concerned i.e. http://doc.qt.nokia.com/4.7/qcombobox.html#currentIndexChanged there you will see that you will either have an integer indicating which item was chosen, or as shown there by the next entry, the text of the item as a QString (in JavaScript casts to a normal string), returned to your function, in this case `comboInsertSwitches_SwitchChosen`

If you wanted the string option, when available, you would need to write the signal connection like this.

```
eval("comboInsertSwitches['currentIndexChanged(QString)']." + connectAction +
"(comboInsertSwitches_SwitchChosen)"
```

And then a script function called `comboInsertSwitches_SwitchChosen` would expect a string, instead of a number. The widget's Qt documentation is the best way to tell what you could expect to find as workable.

This then means that when you write `comboInsertSwitches_SwitchChosen` (you make the name up) you can collect the value returned from the Script Engine supplied arguments array, or if you know what you are expecting, directly as in ...

```
function comboInsertSwitches_SwitchChosen( index)
            {
                if (index == 0) { return}
                }
```

However, you *never* put the variable name `( index )` in ...

```
eval("comboInsertSwitches['currentIndexChanged(int)']." + connectAction
+ "(comboInsertSwitches_SwitchChosen( index ))");
```

Should look like (`functionName()` no following parenthesise) :—

```
eval("comboInsertSwitches['currentIndexChanged(int)']." + connectAction
+ "(comboInsertSwitches_SwitchChosen)");
```

You can test what is returned to your function by using the **arguments** array and iterating over it.

```
  function comboInsertSwitches_SwitchChosen()
              {
                     for (I in arguments)
                         {
                            TW.information(null, "", "Argument Number: "  I +
",   Value: "  + arguments[I]);
                         }
                  }
```

So far I think that I have never found more than one thing returned. So generally the following will always work, if you already have researched and found that there will be something you need there.

```
  function comboInsertSwitches_SwitchChosen(whatEver)
             {

               TW.information(null, "", whatEver);

               anotherFormObject.text = whatEver;

               yetAnotherFormObject.setHtml(whatEver);

             }
```

Putting it all together ...

A script header could look something like the following

With this kind of script operation it is convenient to place the vast majority of functions, unconventionality, at the end of the script :—

```
// TeXworksScript
// Title:  Convert Image
// Description: Uses ImageMagick to make image alterations
// Author:  Paul Norman
// Version: 0.1
// Date: 2011-05-06
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+C, Alt+I


eval(TW.app.getGlobal("helper_twPan")); // Comment if NOT Needed - This
includes PhpJs ($P), twConst, msgBox, twPan ($tw), string.toTitleCase()

var convertImage; // dialogue Script Object
var scriptPath = twPan.callingScriptPath(__FILE__);
var scriptDir = scriptPath.substr(0,scriptPath.length -1);
var documentPath = twPan.callingScriptPath(TW.target.fileName);
var ... etc
var myForms = [];

if (twPan.os == "Windows")
{ var lineBreak = "\r\n";
var cmdSeperator = "&";
}
else
```

```
{var lineBreak = "\n";
var cmdSeperator = ";";// http://vic.gedris.org/Manual-ShellIntro/1.2/
ShellIntro.pdf
}
var scriptWidgets = ["showImageInformation", "radioPerrcentage",
"radioWidth", "radioHeight", "editSize",  ... etc}];

function UIconnectVarious(connectAction)
{
/* This streamlines initialisation and finalisation of the script
dialogue
slot connections connectAction passes either 'connect' or 'disconect'
.slotEvent is something like .clicked
*/
// eval("nameofWidget.slotEvent." + connectAction +
"(functionYouWriteInThisFile)");
// !make function in Global Space!

eval("btnGetAnotherImage.clicked." + connectAction + "(showImage)");
eval("btnConvertUsingSettings.clicked." + connectAction +
"(IM_convert)");
eval("btnJustCopyGraphic.clicked." + connectAction +
"(JustCopyGraphic)");
... etc
eval("main_form.finished." + connectAction + "(main_form_finished)");

}
```

// note above: *main_form*.finished, when referring to the form use its Script given variable name, not necessarily the one seen in the Qt Creator interface.

```
function main_form_finished(response)
        {
```

// put your end of script (finalization) stuff in the function you connect to that signal.

```
        // response will be as shown below: Cancel, or Ok  0, or 1

       UIconnectVarious("disconnect") // see below UIconnectVarious
("connect"|"disconnect")
```

// save settings or do anything else need at the end of the script

```
for (var x = myForms.length -1; x > -1; x--)
{ // done reverse to destroy main form last
        {
               if (myForms[x] !== null)
                {
                 myForms[x].deleteLater();
                }
         }

//           --- Script BEGIN ---

// convertImage = TW.createUI(__FILE__.replace(".js",".ui"));
convertImage = TW.createUIFromString(reBuildUi()); // read in QComboBox
members in own function reBuildUi();
```

```
myForms.push(convertImage);


/* This next loop has to be done in Global space and not in a general
function,
for Script widget objects, and `in memory functions_()' generated here,
to be available generally to script.
Done this way as you don't need to know the parent of a widget first
(as in complex layouts).
*/

for (widget in scriptWidgets)
{
// Define required convertImage dialogue .ui widgets here in QtScript as
programmable objects

eval("var " + scriptWidgets[widget] + " = TW.findChildWidget
(convertImage,\"" + scriptWidgets[widget] + "\")");

}

UIconnectVarious("connect")

// setup procedures (could be in function calls)

showImageInformation.setHtml('<h2 style=\"font-weight:bold\">Please
\'Get A Graphic\' to Use ...</h2>');
// labelScriptLogo.text = "<p><img height=\"185\" src=\""+scriptPath
+"TwScripting_width100px.png\"></p>";

labelVersion.text = "<p>Convert Image v."
+ twPan.file_get_contents(__FILE__).split("\n")[4].replace("// Version:
","")
+ "</p>" ;

labelIMlocalHelp.text = twPan.file_get_contents(scriptPath +
"labelIMlocalHelp.txt");

ans = convertImage.exec();  // this will wait for the dialogue to be
closed.

/* You can test ans for an integer
  ans will generally  be one of:
```

	Integer Returned
QDialog::Accepted (e.g. OK)	1
QDialog::Rejected (e.g. Cancel)	0

```
These can be set on any form widget addressing the form object itself
(default name "dialog" in Slot column) in the Signals and Slots Editor
in Qt Creator.

 */

UIconnectVarious("disconnect")

//           --- Script  END ---.
```

... then heaps and heaps of functions() that you write!

Including functions for the things in the form that you have
made 'connects' to that the script's User will click on and so on ...

Created with the Personal Edition of HelpNDoc: Free Web Help generator

Global variables, and Modules function collections

Global variables, and Modules — function, constants collections

In TeXworks scripting you can store basic structures like stings and possibly dialogues as in memory objects between scripts being run.

All global objects *do not* automatically persist when the User closes Tw.

There ar two main types of Global objects, ones that are as it were attached to the script name, and ones that could be considered as attached to the application above scripting.

Ones attached to the script name can only be accessed from within running instances of that particular script.

Global objects managed by the application can be accessed from within any running script.

For details see Globals

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Importing other function collections

Importing other EMCA function collections

With the development of projects like Node.JS and the like, there is more work being done on JavaScript outside the browser hosting environment. So there are many generically useful JavaScript function collections around.

This collection has proved very useful to me in Tw scripting, it is based on php syntax and is able to use the Php general help files as the functions names and attributes are mostly identical.

http://phpjs.org/pages/home

Some very clever fellows have ported a considerable number of string handling, array handling and other potentially very useful functions across from php to EMCA style scripting, and made them all work as per the PHP manual (documentation as it already exists).

http://phpjs.org/functions/index

As said there would be many things that are too web orientated and would never be used. And some stuff that wold be directly useful now.)

Even more cleverly they have made their 'library' customisable pre-download so that a script-developer could define their own package(s) as needed, with only QtScript relevant functions and dependencies in it.

http://phpjs.org/packages/configure

"Use PHP functions in JavaScript"

"php.js

"php.js is an open source project that brings high-level PHP functions to low-level JavaScript platforms such as web browsers, browser extensions (Mozilla/Firefox, Chrome), AIR, and SSJS engines like V8 (node.js, v8cgi), Rhino, and SpiderMonkey (CouchDB)

"If you want to perform high-level operations on these platforms, you probably need to write JS that combines its lower-level functions and build it up until you have something useful like: strip_tags(), strtotime(), number_format(), wordwrap().

"That's what we are doing for you.

"Pure JavaScript so no additional components required

There are considerable number array and string handling routines ready to go. An estimated 7-8, 000 lines of code.

I import the library run it through eval() and then use its prototype creation call to make an object available to QtScripting. More details below Modules and helper objects and helper_PhpJs.mod.

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

Getting Take Aways (sending out) - TW.app.system()

Getting Take Aways (sending out) - TW.system()

Any thing that you can run from the command line in your OS can be called from Tw's system call —
TW.system()
or throuh a wrapper that handles the response like twPan.osCmd()

JabRef - MySql - Bibliographies - Using php to return text

Worked Example - Consistently Filling Drop Down Boxes etc

Created with the Personal Edition of HelpNDoc: Free help authoring tool

JabRef - MySql - Bibliographies - Using php to return text

JabRef - MySql - Bibliographies - Using php to return text

We extensively use TW.system() among other things, to obtain bibliographical references form an MySql export of JabRef.

JabRef is a Java based Open Source cross platform application that can maintain LaTeX bibliographical listings in required format for direct use in LaTeX documents.
JabRef reference manager

Import of various formats

BibTeXML, CSA, Refer/Endnote, ISI Web of Science, SilverPlatter, Medline/ Pubmed (xml), Scifinder, OVID, INSPEC, Biblioscape, Sixpack, JStor and RIS.

Built-in and custom export formats

HTML, Docbook, BibTeXML, MODS, RTF, Refer/Endnote and OpenOffice.org.

etc ...

JabRef can also export to an Sql (MySql in our case) database.

And form there TeXworks scripting can help you get the correct bibliographical references for use in your LaTeX documents using down boxes.
(See the bottom of system(QString) for the specific TeXworks Scripting files used, here is a brief overview of the other things needed to be done.)

Here are the steps we use inside JabRef and Heidi (an Open Source front end to MySql MariaDB etc - but any MySql manager could be used http://www.heidisql.com/. "HeidiSQL runs fine on Windows (2000, XP, Vista, 7) and on any Linux with Wine" Heidi can export table rows into LaTeX).

JabRef steps

From http://jabref.sourceforge.net/help/SQLExport.php

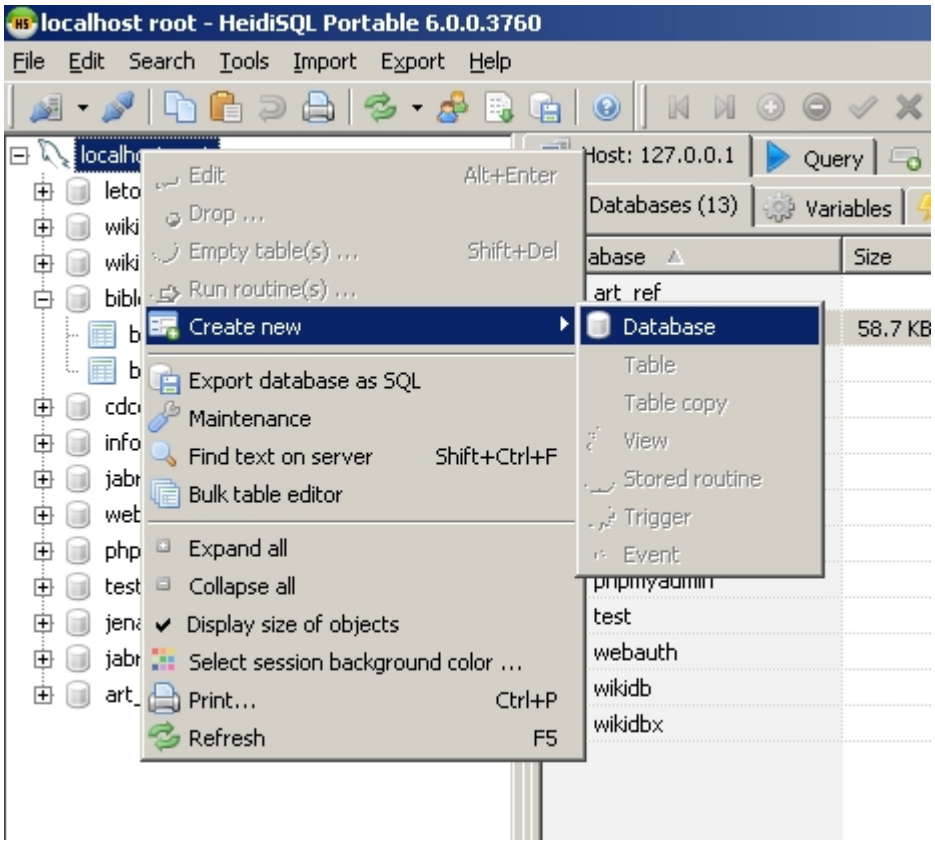
Export to an External SQL Database

JabRef is capable of exporting the contents of the BibTeX database, along with groups information, to an external MySQL database.

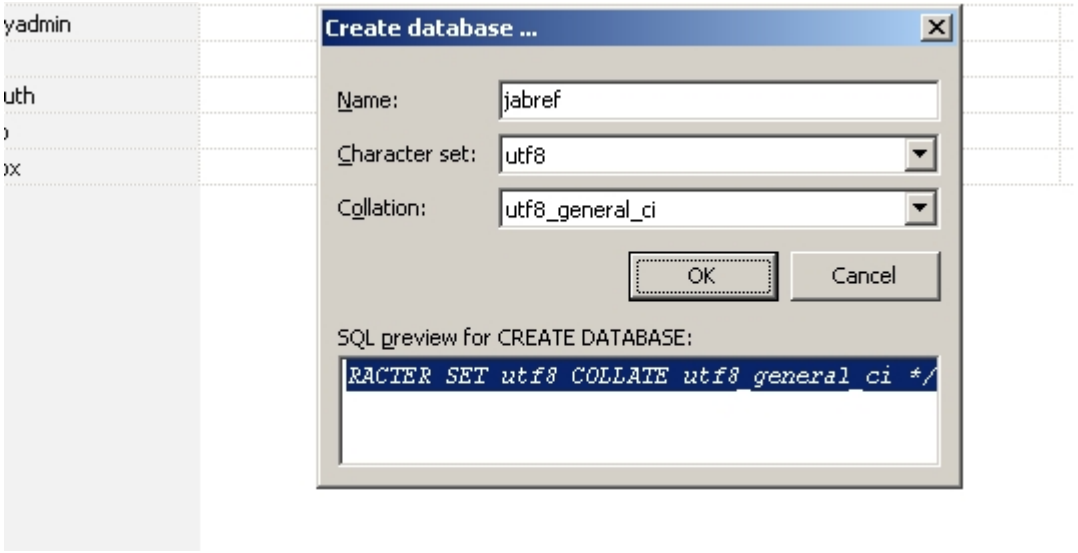
Setup

1. Using your favorite MySQL administration tool, create an empty MySQL database. (You only do this once - on the first time)

   In Heidi, right mouse button localhost



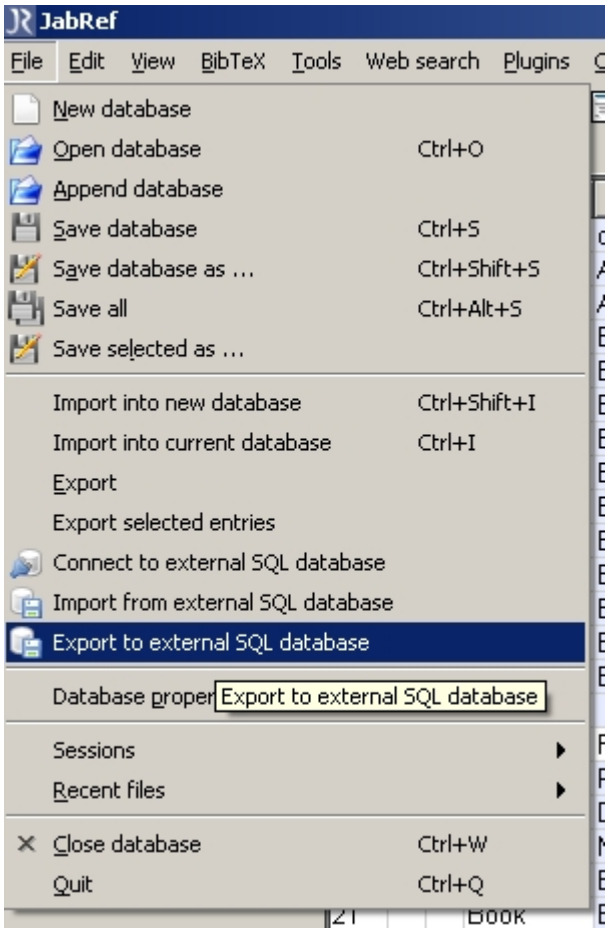
CREATE DATABASE `jabref` /*!40100 CHARACTER SET utf8 COLLATE utf8_general_ci */



2. Make sure there is a user for this database that has *full privileges*.

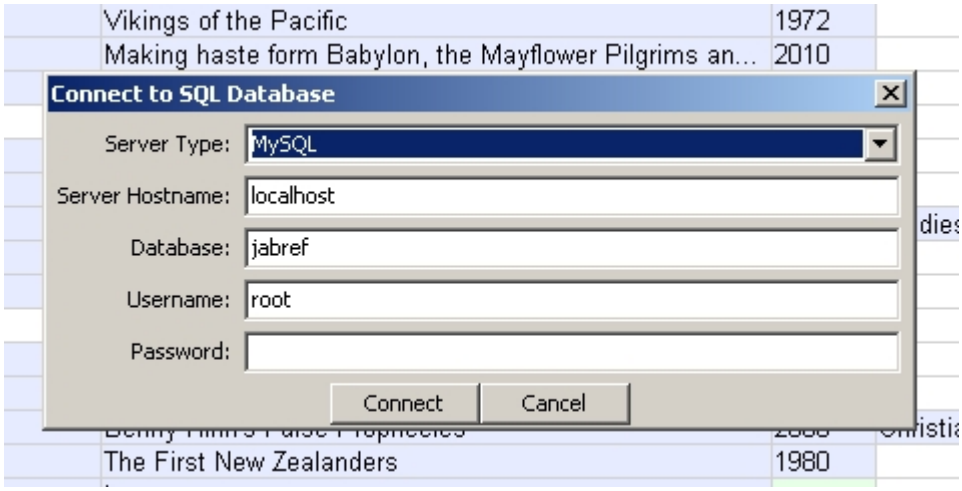
Export

1. Choose **File -> Export to external SQL database**,



or click the corresponding button on the toolbar.

2. Enter the database connection information, and click **Connect**.



JabRef will then connect to the specified database, ***drop the existing tables***, create new tables, and populate those tables with entries and groups information. Note that you will not be prompted for the connection information on subsequent exports. If you would like to export to a different database, you can change the connection information by choosing **File -> Connect to external SQL database** (or by clicking the associated toolbar button), and then performing an export.

You need to reexport to MySql from JabRef when you have changed your JabRef LaTeX biblkiography entries. You might leave doing that until you actually

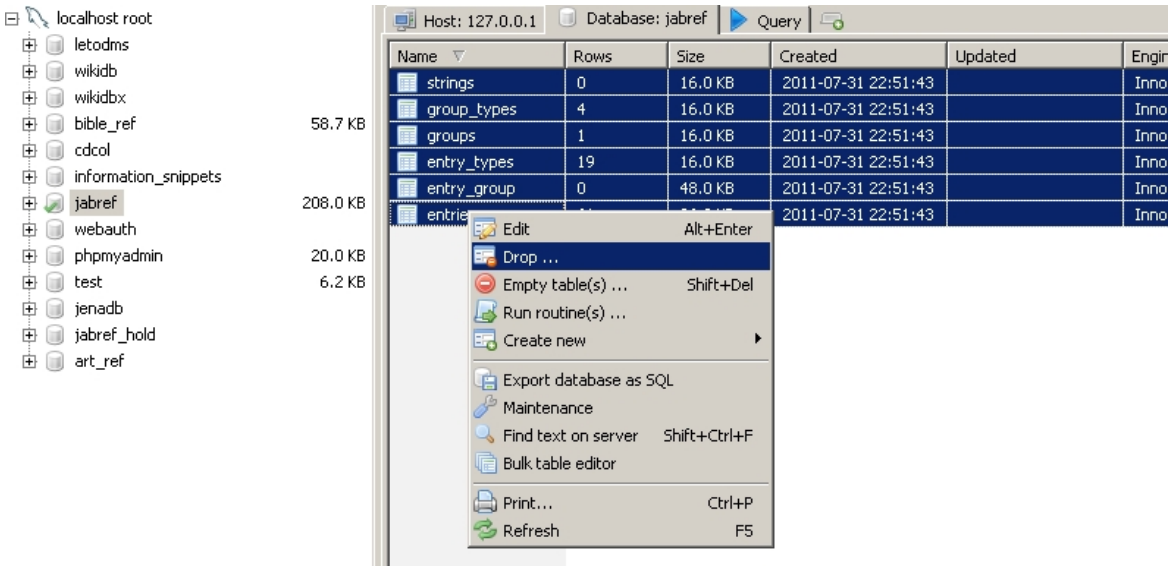
need to use them in LaTeX via TeXworks.

On later attempts, if JabRef can not export into the MySql, you may get a message like this:—

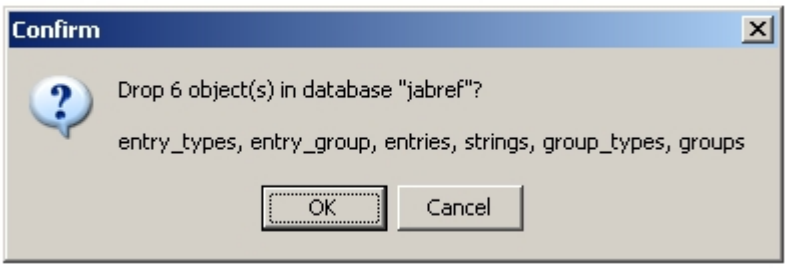
You need to remove the tables under the jabref database, leaving the jabref database there.

In Heidi you can do the following:—

Click on the database jabref, and select all the tables (Crtl and shift key combinations with clicks), and then right mouse button while over the selected tabes...



Then OK the next box...



Go back to JabRef and again Export to MySql again.

Now it is time to use the TeXworks scripts. (See the bottom of system(QString) for the specific TeXworks Scripting files used. Also check http://twscript.paulanorman.com/ downloads on http://twscript.paulanorman.com/downloads/ in August 2011 for a zip with project files.)

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

Worked Example - Consistently Filling Drop Down Boxes etc

Consistently Filling Drop Down Boxes With Up To Date Information

Obtaining Consistency and Accuracy Through Scripting Automation

N.B. Some of these techniques use \write18{...} from within a .sty. Whether used in the preamble of your document or form a .sty through using that package, \write18 has certain security issues. **I strongly suggest** that you acquaint yourself with those issues before using the techniques using \write18{} here.

Under Windows, Heidi can be used to create and fill out MySql databases. MySql provides an efficient and effective means of centralising data that will be used for various drop-down boxes in TwScripting, and can be accessed by other applications as well. More details follow...

> http://www.heidisql.com/
>
> "**HeidiSQL** is an easy-to-use interface and a "working-horse" for web-developers using the popular MySQL-Database. It allows you to manage and browse your databases and tables from an intuitive Windows® interface."

The following example is specific to some very definite needs, yet has in it elements and features that may be found useful in many publishing projects. But as it has grown by accretion is not necessarily the best solution to the overall problem it

seeks to solve, and defiantly is not any sort of an example of optimised code. Plus I am not a LateX or TeX guru by any means!

We have recourse to quote a lot of Scripture. Various Bible publishers can be *very stringent* in their requirements for counting the number of verses used, and for correct placement of copyright notices.

So we needed to quote scripture verses in such a manner that an accurate reference was made to the quoted bible, the verses were counted, and when the document was prepared, copyright notices for any bibles used in the document are placed in the appropriate section with a tally of verses used.

It is surprisingly easy to get this wrong, when it is all done by hand, and that increases the burden for proof readers.

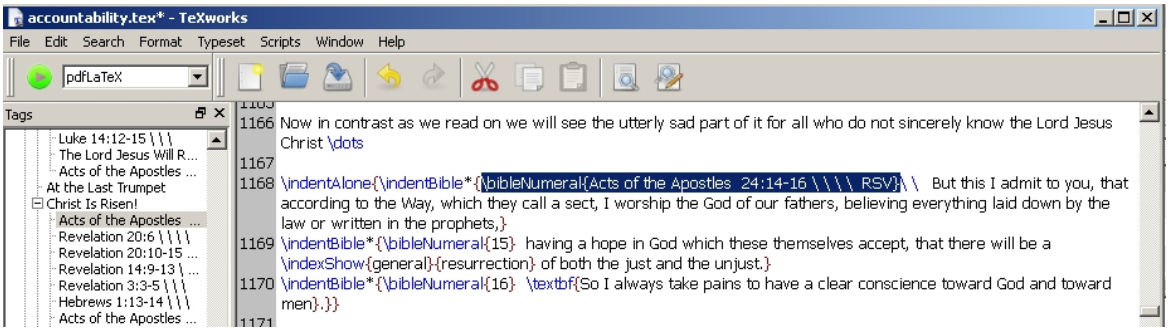
We also needed that any corrections or updates to copyright notices and other bible details, would be universally corrected in all .tex that use them when making a new .pdf

The task was divided up into areas of technology. (Please see the following files for the detail - this overview will hopefully help pull what you read in those together.)

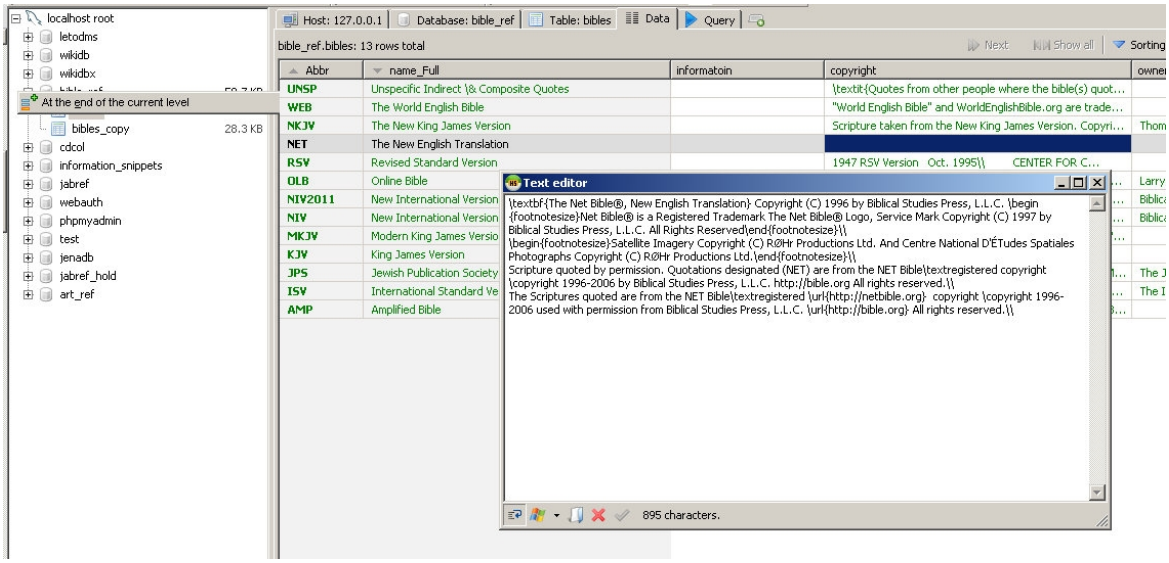
bibleVerseParNoHeads.js
getBibleAbrsCommaList.php
doBibleInfo.php
panMagUseAa.sty
helper_twPan.mod

TeXworks Scripting would handle the insertion of the verses, and ensuring that correct references were being made, syntax-patterns.txt would handle making an inclusion of the bible reference in the document's tag window in the outline section ...

```
5 \\bibleNumeral\{(\d?.*[a-zA-Z].+.\s).+\s([A-Z]{3})\}
```

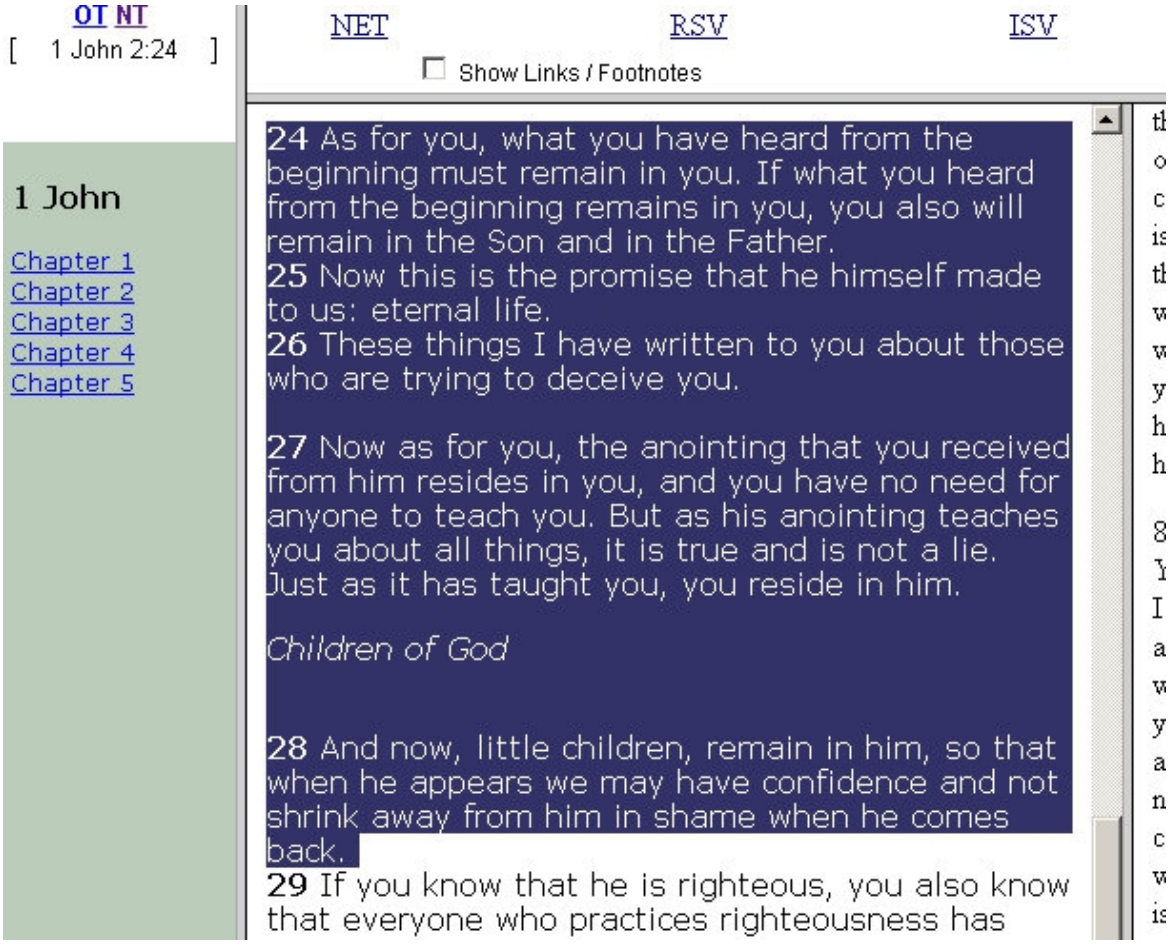


... through Heidi, MySql would store the bible and publishers' information as LaTeX.




```
CREATE DATABASE `bible_ref` /*!40100 CHARACTER SET utf8
COLLATE utf8_general_ci */

CREATE TABLE `bibles` (
`Abbr` VARCHAR(15) NOT NULL,
`name_Full` TEXT NOT NULL,
`informatoin` TEXT NOT NULL,
`copyright` TEXT NOT NULL,
`owner` TEXT NOT NULL,
`contact` TEXT NOT NULL,
`contact_email` TEXT NOT NULL,
`http` TEXT NOT NULL,
`permission` TEXT NOT NULL,
PRIMARY KEY (`Abbr`)
)
COLLATE='utf8_general_ci'
ENGINE=MyISAM;
```

In any arbitrary bible display layout, the bible verses would be selected and copied to the clipboard,



TWScript system() calls would use a batch file (shell script) in the DOS path, which then calls a command line (CLI) php to poll the MySql databse and bible_ref table to obtain and prepare material for the TwScript drop-down list for the User to choose from (covered below).

A LaTeX style sheet (for shared document usage) would format the verses,

1 John 2:24-28 NET As for you, what you have heard from the beginning must remain in you. If what you heard from the beginning remains in you, you also will remain in the Son and in the Father.

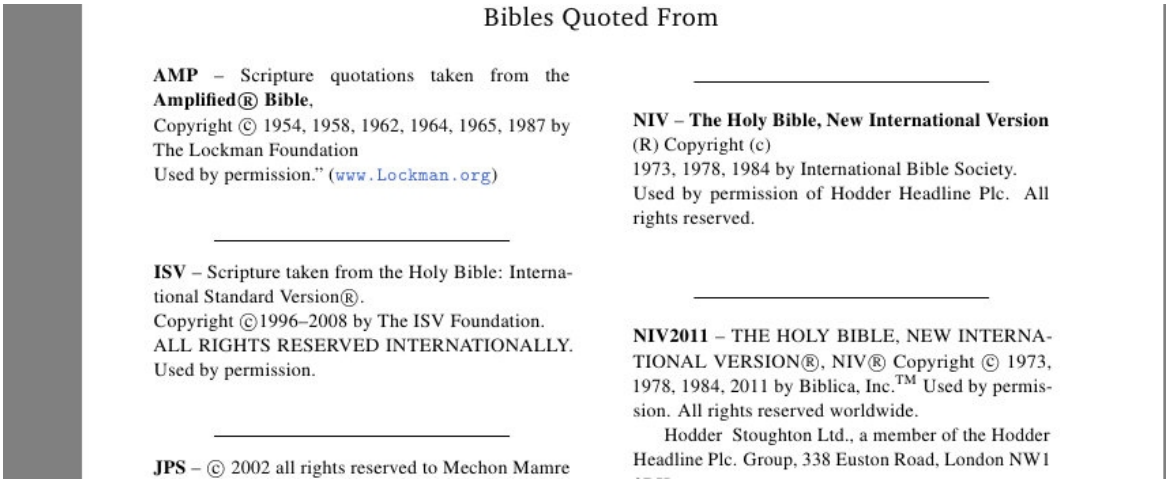
25 Now this is the promise that he himself made to us: eternal life.

26 These things I have written to you about those who are trying to deceive you.

27 Now as for you, the anointing that you received from him resides in you, and you have no need for anyone to teach you. But as his anointing teaches you about all things, it is true and is not a lie. Just as it has taught you, you reside in him.

28 And now, little children, remain in him, so that when he appears we may have confidence and not shrink away from him in shame when he comes back.

... using \immediate\write18{da dahd dah} the .sty retrieves copyright information for only the
actual bibles used in a particular document,



Bibles Quoted From

AMP – Scripture quotations taken from the **Amplified® Bible**, Copyright © 1954, 1958, 1962, 1964, 1965, 1987 by The Lockman Foundation Used by permission." (www.Lockman.org)

ISV – Scripture taken from the Holy Bible: International Standard Version®. Copyright ©1996–2008 by The ISV Foundation. ALL RIGHTS RESERVED INTERNATIONALLY. Used by permission.

JPS – © 2002 all rights reserved to Mechon Mamre

NIV – The Holy Bible, New International Version (R) Copyright (c) 1973, 1978, 1984 by International Bible Society. Used by permission of Hodder Headline Plc. All rights reserved.

NIV2011 – THE HOLY BIBLE, NEW INTERNATIONAL VERSION®, NIV® Copyright © 1973, 1978, 1984, 2011 by Biblica, Inc.™ Used by permission. All rights reserved worldwide.

Hodder Stoughton Ltd., a member of the Hodder Headline Plc. Group, 338 Euston Road, London NW1

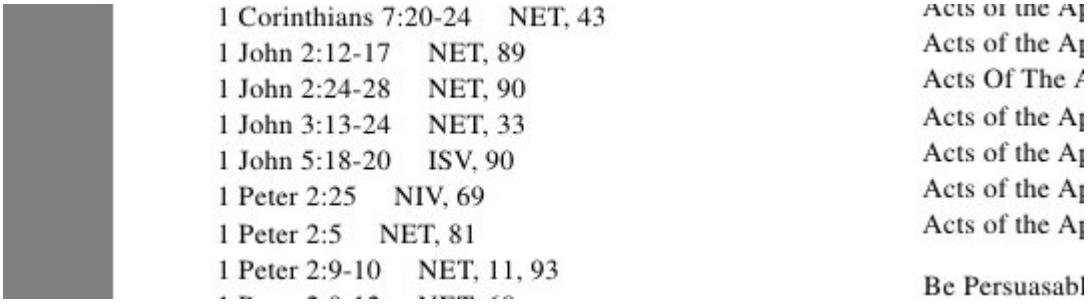
and the .sty would keep track of the bible and verse usage,

Amplified Bible	AMP	118
International Standard Version	ISV	130
Jewish Publication Society's 1917 edition of the Hebrew Bible in English	JPS	5
The New English Translation	NET	505
New International Version 1984	NIV	127
New International Version 2011	NIV2011	1
Revised Standard Version	RSV	79
Unspecific Indirect & Composite Quotes	UNSP	12
	Total:	977

ISBN 978-1-877464-08-9

@Thursday 28th July 2011 Download from pdf.PaulANorman.com

and insert an entry in a sperate index for bible quotes, for each verse portion referenced.



This would be achieved by Document calls to the .sty macros

We work a lot from portable path set ups, and so use intermediary batch files which are placed in the path (even temporarily set so) to explicitly call php in case php is unaware of various path aspects of our Tw and MiKTeX setups. In such circumstances php will require an absolute path to the php script to be run. And we have found this approach to be stable so far.

We reuse a lot of these functions so this is the flow...

From a function in the main script helper module (loaded at start up into a global variable. the module is viewable here helper_twPan.mod

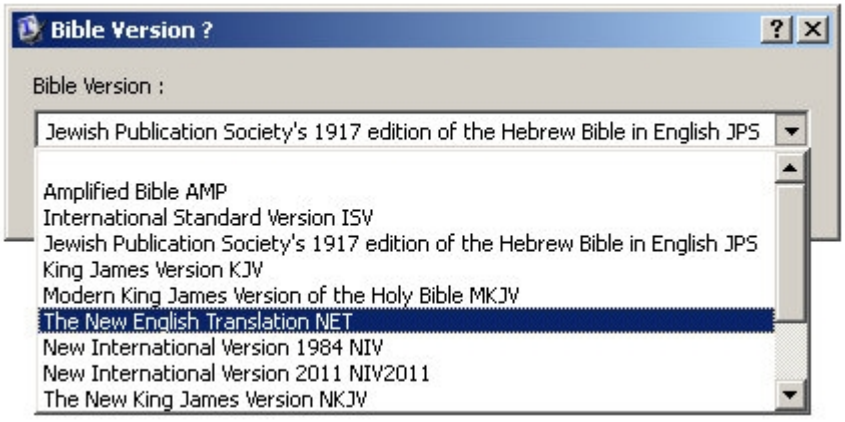
```
var bibleList = this.osCmd('cmd /c getBibleNamesAbrs.php.bat',
true);

   return bibleList.split('\n');
```

And in `getBibleNamesAbrs.php.bat`

```
@echo off
rem make sure this bat file directory is in path
php \LaTeXPortable\LatexUtils\TeXworks\TeXworks\config
\scripts\PHPtexworks\getBibleNamesAbrs.php
```

With `bibleList` assigned as an array `getNamesAbrvs`, it is used in this function call:

```
     userChoice  = TW.getItem( null,  "Bible Version ?", "Bible Version :
",
                                          getNamesAbrvs ,  choiceIndex ,
true ) ;

      var result = [];

       if( userChoice == undefined)
          {
         result.abbrv= '';
         result.full= '';
           }
        else
        {
         var abrvStart = userChoice.lastIndexOf(' ');

         // line returned from dialogue box split into component parts
          result.abbrv = userChoice.substr(abrvStart + 1);

          result.full  = userChoice.substr(0, abrvStart );
        }

         return result;
```



Then a drop-down asks for the book and chapter reference in a stipulated (conventional) form.

ntBible*{\bibleNumeral{}}
ntBible*{\bibleNumeral{28} And now, little chilc
have confidence and not shrink away from him

This ends up by inserting the following in the Tw Editor (script here) [bibleVerseParNoHeads.js](#) ...

```
4560
4561 So as it was with Eve, so it is now for us\dots
4562
4563 \indentAlone{\indentBible*{\bibleNumeral{1 John 2:24-28 \ \ \ \ NET}\ \  As for you, what you have heard from the
     beginning must remain in you. If what you heard from the beginning remains in you, you also will remain in the Son
     and in the Father. }
4564 \indentBible*{\bibleNumeral{25} Now this is the promise that he himself made to us: eternal life. }
4565 \indentBible*{\bibleNumeral{26} These things I have written to you about those who are trying to deceive you.}
4566 \indentBible*{\bibleNumeral{27} Now as for you, the anointing that you received from him resides in you, and you
     have no need for anyone to teach you. But as his anointing teaches you about all things, it is true and is not a lie. Just
     as it has taught you, you reside in him.}
4567 \indentBible*{\bibleNumeral{28} And now, little children, remain in him, so that when he appears we may have
     confidence and not shrink away from him in shame when he comes back.}}
4568
4569 Satan knows that if the church remains in Christ's teachings then ---
```

Which displays as the above [pdf output](#)

The .sty initially retrieves a list of current abbreviations from the MySql again through \immediate\write18
(portions shown below - present .sty version - *rough and ready!* 2011-07-31 available for perusal
here: [panMagUseAa.sty](#))

```
\newcommand{\twScriptPhp}[1]{/LaTeXPortable/LatexUtils/TeXworks/
TeXworks/config/scripts/PHPtexworks/#1}
```

`\twScriptPhp` used below stores a hardwired relative path to the php files needed, and applies it to a specific php file as pased.

```
\immediate\write18{php "\twScriptPhp{getBibleAbrsCommaList.php}"
>  getBibleAbrvs.txt}
%
\RequirePackage{catchfile}%
\CatchFileDef{\versionList}{getBibleAbrvs.txt}{}
```

`getBibleAbrvs.txt`, depending on the current document's bible usage, contains a

comma separated list like this

```
AMP,ISV,JPS,KJV,MKJV,NET,NIV,NIV2011,NKJV,OLB,RSV,UNSP,WEB
```

We chose for long term stability instead to use Heiko Oberdiek's **catchfile** package
http://www.ctan.org/pkg/catchfile
This gives us a globally accessible variable `\versionList` which contains a list of currently stored bible name abbreviations for use in other macros. This is only retrieved once during the document typesetting but is used at least five times I think.

The `\versionList` varible is used in this manner:-

```
%\newcommand{\makeCounters} {%
\@for\val:=\versionList\do{%
\newcounter{ver\val}% used for keeping track of bible verse usage on for each version of the
bible
}
%
\newcommand{\checkVersion}[1]{% helper function
\@for\val:=\versionList\do{%
\IfSubStr{#1}{\val}{\global\let\bibleVersion\val}{}%
}
}%
```

and among other places like this:-

```
%:      \bibleAllUsedCopyrights    \bibleCopyright{ useabreviation
from MySql/bible_ref/bibles/Abbr}
%    and    \bibleAllUsedCopyrights does all where verse count
above zero
%
\newcommand{\bibleAllUsedCopyrights}{%
\@for\val:=\versionList\do{%
\ifthenelse{\value{ver\val}>0}{\bibleCopyright{\val}}{}% // some
verses from this bible version
}}
%
\newcommand{\bibleCopyright}[1]{
\immediate\write18{php "\twScriptPhp{doBibleInfo.php}"
"copyright" "#1" "Abbr"  >  getScripturePermissions#1.txt}
\input{getScripturePermissions#1.txt}}
%
```

No doubt those better acquainted with TeX and LaTeX could devise better more optimised macros, and would avoid the obvious mistakes that I have made (and somehow got a way with). I hope none-the-less that this project of ours may ideas wise, prove helpful to others in solving their needs.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

bibleVerseParNoHeads.js

```
//TeXworksScript
```

```
//Title: Bible Verses No Headings 1Col Parbox &I &N
//Description: Puts the current selection of verses in Document No
headings
//Author: Paul A Norman
//Version: 0.1
//Date: 2009-09-26
//Script-Type: standalone
//Context: TeXDocument
//Shortcut: Alt+I, Alt+N

// Insert bible Verse



eval(TW.app.getGlobal("helper_twPan")); //  Comment if NOT Needed -
This includes PhpJs ($P), twConst, msgBox, twPan ($tw), string.
toTitleCase()


txt = TW.target.selection;
if ((txt == "") | (txt == null))
 {
  if (TW.app.clipboard != "")
       {
       txt = TW.app.clipboard;
        }
    else
          { txt = "verses here\\\\";
           }
     }
// '\u005C'+'\u005C'

txt = twPan.convertSpeechMarks(txt);

     // txt =  txt.replace(/\n/g, '\\\\ \n');

txt = twPan.cleanParaMarks(txt);

txt = txt.replace(/\n\n/g,"\n");   // get rid of double line breaks

 var  firstColon = txt.indexOf(":"); // see if we have a reference at
beginning
    if ((firstColon > 0) & (firstColon < 20)) // look within first 20
chars
        {
          var referencePrompt = txt.substr(0, firstColon +1); //
extract reference stem
              txt = txt.substr(firstColon +1); // remove reference
stem
           }

var bibleReference =  TW.getText( null, "Verse Refereence", "Please
enter Main Scripture reference\n\nUpto and including \":\"",
referencePrompt);

 if (bibleReference == undefined){bibleReference = ""; var dash = ""}
```

```
else {var dash  = "-";}

  var processBlocks = txt.split('\n');
     var keepLastReferenceNumerals = '';
     var keepFirstReferenceNumerals = '';

     // blockType = TW.getItem( null, "Block Type", "Choose Indent
Block Type: ", ["indentBible","indentPlain"] , 0 , true ) ;

        for (block in processBlocks)
               {
                              var findBlank = processBlocks[block].
split(" ");
                            if (! isNaN(findBlank[0]))
                                {// we have leading numerals
                                        keepLastReferenceNumerals =
findBlank[0];
                                           if (block == 0)

{ keepFirstReferenceNumerals = findBlank[0];
                                           bibleReference = "\
\bibleNumeral{" + bibleReference  + findBlank[0] ;
                                           findBlank[0] =
'referenceReplace';
                                             }
                                            else
                                            { findBlank[0] = "\
\bibleNumeral{"+findBlank[0]+"}";
                                             }


                                           processBlocks[block] =
findBlank.join(" ");
                                       }
                               processBlocks[block] = "\\indentBible*{"
+ processBlocks[block] + "}";
                              }


                            if (keepFirstReferenceNumerals ==
keepLastReferenceNumerals)
                               {bibleReference = bibleReference  ;}
                              else
                                  {bibleReference = bibleReference +
dash + keepLastReferenceNumerals ;}

    abbrvInfo = twPan.getBibleVersion().abbrv;

      var extraSpaceB4Aprev = " \\ \\ \\ \\ ";

  bibleReference = bibleReference + extraSpaceB4Aprev + abbrvInfo +
"}\\ \\ ";

 switch( TW.getItem(null,"Use Tag?","Loose Verses or \
\indentAlone ?",["\\indentAlone{}","Loose"],0))
   {
```

```
   case "Loose":
       TW.target.insertText(processBlocks.join('\n').replace
('referenceReplace', bibleReference));
     break;

   case undefined:
      break;

   default:
   TW.target.insertText("\n\\indentAlone{"
        +  processBlocks.join('\n').replace('referenceReplace',
bibleReference)
              +"}");
   }



//var bibleHeading =  TW.getText( null, "Top Heading", "Please enter
a heading \nor Scripture reference" );
//var TocEntry =  TW.getText( null, "TOC Entry", "Please enter TOC
entry\n or nothing", "" );

/*    TocLevel = "";

          if (TocEntry != "")
        {
            TocLevel =  TW.getItem( null, "TOC Level", "1 - section,
\n2 - subsection ,\n3 - subsubsection", ["1","2","3"] , 2 , true ) ;
   }
       else
       {TocLevel = "";}

       bibleRef = TW.getText( null, "Bible Reference", "Please enter
Bible Reference\nBible Version ,,,", "NIV" );

TW.target.insertText("% {Heading-Reference}{verses}{toc entry or
blank}{if toc then level 1,2,3}{Bible Ref ... Version}" +
"\n\\bibleVersePar{"+bibleHeading+"}{\n" + processBlocks.join('\n') +
"\\\\ \n}{"+TocEntry+"}{"+TocLevel+"}{"+bibleRef+"}");
*/
```

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

panMagUseAa.sty

This is a bad example of a .sty, hobbled together through Google searches! There are some latent errors in places, and superfluous code.

I am not a LateX or TeX guru by any means at all!

View it in TeXworks and use the tag Window.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{panMagUseAa}[2009/09/05 Misc commands]
```

```
\RequirePackage{setspace}
%
%: Drop Cap - Use As   \cappar Blah Dah De Blah
%\newcommand{\panyinipar}[1]{\par\noindent\paninitpar{#1}}
%\newcommand{\paninitpar}[1]{\setbox0=\hbox{{\fontfamily{times}\fontsize
{30pt}{42pt}\selectfont #1}}%
%\hangindent=\wd0\hangafter=-4\advance\hangindent by .25em
%{\dimen@=-3\baselineskip
%\dimen@=\baselinestretch\dimen@
%\hskip-\wd0 \hskip-.25em
%\raisebox{\dimen@}[0pt][0pt]{\unhbox0}\hskip.25em}}
%
\font\capfont=cmbx12 at 40 pt % or yinit, or...?
\newbox\capbox \newcount\capl \def\a{A}
\def\docappar{\medbreak\noindent\setbox\capbox\hbox{\capfont\a
\hskip0.05em}%
\hangindent=\wd\capbox%
\capl=\ht\capbox\divide\capl by\baselineskip\advance\capl by1\hangafter=-2%
\capl%
\hbox{\vbox to10pt{\hbox to0pt{\hss\box\capbox}\vss}}}
\def\cappar{\afterassignment\docappar\noexpand\let\a }
%
%: \doDropCap
\newcommand{\doDropCap}[1]{
\def\dropcapLetter{\StrLeft{#1}{1}}
\def\restPlain{\StrGobbleLeft{#1}{1}}
\newdimen\holdbaselineskip
\setlength\holdbaselineskip{\baselineskip}
\setlength{\baselineskip}{24pt}
\setstretch{2.0}
\noindent  \cappar \dropcapLetter {\fontfamily{bch}\fontsize{12pt}{42pt}
\selectfont \restPlain%
}
\setlength\baselineskip{\holdbaselineskip}\setstretch{1}
\par}
%: \RequirePackages
\RequirePackage{suffix}
\RequirePackage[greek, english]{babel}
\RequirePackage[OT1]{fontenc}% OT2
\RequirePackage{xkeyval}
\RequirePackage{calc}
\RequirePackage[svgnames, hyperref, x11names,rgb,usenames,table]{xcolor}
\RequirePackage{xifthen}
\RequirePackage{boolexpr} % SWITCH CASE
\RequirePackage{pgf}
\RequirePackage{xstring}
\RequirePackage[hyphens]{url}
\RequirePackage{boites,boites_exemples}
\RequirePackage{forarray}
%: --   hyperref stuff
%
\RequirePackage[bookmarks,bookmarksopen=true,colorlinks,breaklinks,
hyperindex,pdfpagetransition=Glitter, backref=pages,hyperfootnotes=true,
```

```
pageanchor,pagebackref=true,pdfdisplaydoctitle=true]{hyperref}
\renewcommand{\backrefxxx}[3]{%
 \hyperlink{page.#1}{\textcolor[rgb]{.1,.2,.9}{ Pg. #1}}}
\urlstyle{tt}
%: \ReverseHeading Preamble
%
% \makeatletter
\define@boolkey{ReverseHeading}{subLine}[true]{\def\ReverseHeadingsubLine
{#1}}
\define@key{ReverseHeading}{sublineWidth}
{\def\ReverseHeadingsublineWidth{#1}}
\define@key{ReverseHeading}{sublineStart}
{\def\ReverseHeadingsublineStart{#1}}
\define@boolkey{ReverseHeading}{putinToc}[true]{\def\ReverseHeadingputinToc
{#1}}
\define@key{ReverseHeading}{logoA}
{\def\ReverseHeadinglogoA{#1}}
\define@key{ReverseHeading}{logoB}
{\def\ReverseHeadinglogoB{#1}}
\define@key{ReverseHeading}{logoC}
{\def\ReverseHeadinglogoC{#1}}
\define@key{ReverseHeading}{TopHeading}
{\def\ReverseHeadingTopHeading{#1}}
\define@key{ReverseHeading}{SubHeading}
{\def\ReverseHeadingSubHeading{#1}}
\savekeys{ReverseHeading}{logoA,logoB,logoC,TopHeading,SubHeading,subLine,
putinToc,sublineWidth,sublineStart}
\presetkeys{ReverseHeading}%
{logoA=0,logoB=0,logoC=0,TopHeading=Type TopHeading,SubHeading=Type
SubHeading,subLine,putinToc,sublineWidth=364.955703bp,
sublineStart=14.255828bp}%
{}
%: ReverseHeading*
\newcommand*\ReverseHeading[2][]{%
\setkeys{ReverseHeading}{#1}%
{\vspace{15pt}
\ifthenelse{\boolean{\ReverseHeadingputinToc}}{%
%: -- Headings Footers TOC
\markright{\ReverseHeadingSubHeading}%
\phantomsection%
\addcontentsline{toc}{section}{\ReverseHeadingTopHeading}%
\sectionmark{\ReverseHeadingTopHeading}%
\phantomsection%
\addcontentsline{toc}{subsection}{\ReverseHeadingSubHeading}%
\subsectionmark{\ReverseHeadingSubHeading}%
\markright{\ReverseHeadingSubHeading}%
}{}%
\noindent\begin{pgfpicture}%{0bp}{0bp}{496.028168bp}{84.785092bp}
\begin{pgfscope}
\pgftransformcm{1.0}{0.0}{0.0}{1.0}{\pgfpoint{13.864171bp}
{23.11675bp}}
% SUB HEADING UNDER A REVERSE
\pgftext[left,base]{\fontfamily{bch}\selectfont\itshape\Large
```

```
\color[rgb]{0.0,0.0,0.0}\makebox[\textwidth][c]{\ReverseHeadingSubHeading}}
\end{pgfscope}
\begin{pgfscope}
\pgfsetlinewidth{1.1200000047683716bp}
\pgfsetrectcap
\pgfsetmiterjoin \pgfsetmiterlimit{10.0}
% Topheading reverse whjite on darkgray
\pgfpathmoveto{\pgfpoint{0.56bp}{81.225093bp}}
\pgfpathlineto{\pgfpoint{0.56bp}{49.333437bp}}
\pgfpathlineto{\pgfpoint{495.468156bp}{49.333437bp}}
\pgfpathlineto{\pgfpoint{495.468156bp}{81.225093bp}}
\pgfpathlineto{\pgfpoint{0.56bp}{81.225093bp}}
\pgfclosepath
\color[rgb]{0.5,0.5,0.5}\pgfseteorule\pgfusepath{fill}
% Topheading revers whjite on darkgray
\pgfpathmoveto{\pgfpoint{0.56bp}{81.225093bp}}
\pgfpathlineto{\pgfpoint{0.56bp}{49.333437bp}}
\pgfpathlineto{\pgfpoint{495.468156bp}{49.333437bp}}
\pgfpathlineto{\pgfpoint{495.468156bp}{81.225093bp}}
\pgfpathlineto{\pgfpoint{0.56bp}{81.225093bp}}
\pgfclosepath
\color[rgb]{0.5,0.5,0.5}
\pgfusepath{stroke}
\end{pgfscope}
\begin{pgfscope}
\pgftransformcm{1.0}{0.0}{0.0}{1.0}{\pgfpoint{12.168343bp}
{59.11675bp}}
\pgftext[left,base]{\fontfamily{bch}\selectfont\bfseries\LARGE%\slshape
\color[rgb]{1.0,1.0,1.0}\ReverseHeadingTopHeading}
\end{pgfscope}
\ifthenelse{\equal{\ReverseHeadinglogoA}{0}}{}{%
\begin{pgfscope}
\pgftransformcm{0.5}{-0.0}{0.0}{0.5}{\pgfpoint{370.53914bp}{37.5bp}}
\pgfputat{\pgfpoint{0pt}{0pt}}{\pgftext[top,left]{\pgfimage
{\ReverseHeadinglogoA}}}
\end{pgfscope}%
}%
\ifthenelse{\equal{\ReverseHeadinglogoB}{0}}{}{\begin{pgfscope}
\pgftransformcm{0.5}{-0.0}{0.0}{0.5}{\pgfpoint{413.876625bp}{37.5bp}}
\pgfputat{\pgfpoint{0pt}{0pt}}{\pgftext[top,left]{\pgfimage
{\ReverseHeadinglogoB}}}
\end{pgfscope}%
}%
\ifthenelse{\equal{\ReverseHeadinglogoC}{0}}{}{\begin{pgfscope}
\pgftransformcm{0.5}{-0.0}{0.0}{0.5}{\pgfpoint{457.214109bp}{37.5bp}}
\pgfputat{\pgfpoint{0pt}{0pt}}{\pgftext[top,left]{\pgfimage
{\ReverseHeadinglogoC}}}
\end{pgfscope}%
}%
\ifthenelse{\boolean{\ReverseHeadingsubLine}}{\begin{pgfscope}
\pgfsetlinewidth{1.4943959712982178bp}
\pgfsetrectcap
\pgfsetmiterjoin \pgfsetmiterlimit{10.0}
```

```
\pgfpathmoveto{\pgfpoint{\ReverseHeadingsublineStart}{8.995953bp}}%
14.255828bp
\pgfpathlineto{\pgfpoint{\ReverseHeadingsublineWidth}{8.995953bp}}
\color[rgb]{0.0,0.0,0.0}\pgfseteorule\pgfusepath{fill}
\pgfpathmoveto{\pgfpoint{\ReverseHeadingsublineStart}{8.995953bp}}%
14.255828bp
\pgfpathlineto{\pgfpoint{\ReverseHeadingsublineWidth}{8.995953bp}}
\color[rgb]{0.25,0.25,0.25}
\pgfusepath{stroke}
\end{pgfscope}
}{}
\end{pgfpicture}
\par
\ifthenelse{\boolean{\ReverseHeadingsubLine}}{}{\vspace{10pt}}
\ifthenelse{\equal{#2}{}}{\vspace{4pt}\noindent}{
\def\dropcapLetter{\StrLeft{#2}{1}}
\def\restPlain{\StrGobbleLeft{#2}{1}}
\newdimen\holdbaselineskip
\setlength\holdbaselineskip{\baselineskip}
\setlength{\baselineskip}{16pt}
\noindent  \cappar \dropcapLetter {\fontfamily{bch}\fontsize{12pt}{36pt}
\selectfont \restPlain}
 \setlength\baselineskip{\holdbaselineskip}
\par}
}
}
%\makeatother
\RequirePackage{xkeyval}
%
% \makeatletter
 \newdimen\shadowsize
\define@boolkey{Fbox}{frame}[true]{}
 \define@boolkey{Fbox}{shadow}[true]{}
 \define@key{Fbox}{framecolor}%
 {\def\Fboxframecolor{#1}}
 \define@key{Fbox}{shadowcolor}%
{\def\Fboxshadowcolor{#1}}
\define@key{Fbox}{framesize}%
{\setlength\fboxrule{#1}}
\define@key{Fbox}{shadowsize}%
{\setlength\shadowsize{#1}}
\savekeys{Fbox}{frame,framecolor,framesize}
\presetkeys{Fbox}%
{frame,framecolor=black,framesize=0.5pt}%
{shadow=\usevalue{frame},
shadowcolor=\usevalue{framecolor}!40,
shadowsize=\usevalue{framesize}*4}
\newcommand*\Fbox[2][]{%
\setkeys{Fbox}{#1}%
{\ifKV@Fbox@frame\else\fboxrule0pt\fi
\ifKV@Fbox@shadow\else\shadowsize0pt\fi
\sbox0{\fcolorbox{\Fboxframecolor}{white}{#2}}%
\hskip\shadowsize
```

```
\color{\Fboxshadowcolor}%
\rule[-\dp0]{\wd0}{\ht0+\dp0}%
\llap{\raisebox{\shadowsize}%
{\box0\hskip\shadowsize}}}%
}
%\makeatother
%
%
%: BoldColSubHd
\newcommand{\BoldColSubHd}[1]{
\vspace{5pt}
\phantomsection%
\addcontentsline{toc}{subsection}{#1}\subsectionmark{#1}%
\noindent \begin{center}\begin{minipage}[c]{\columnwidth}\LARGE\textbf{#1}
\end{minipage}\end{center}\\[5pt]
}
%: FootNote struff
%
% \setFooteNoteFont
\newcommand{\setFootNoteFont}{\setstretch{0.89}\fontsize{9}{7}\selectfont
\itshape}%
%
%:      \RequirePackage{endnotes}
\RequirePackage{endnotes}
%:      \let\footnote=\endnote
\let\footnote=\endnote
%:      \RequirePackage{footmisc}
\RequirePackage{footmisc}
\renewcommand\footnotelayout{\setFootNoteFont}
%
%: -- reset endnotes
%\makeatletter
%from endnotes.sty ADDED: [pg \thepage] as below
\long\def\@endnotetext#1{%
    \if@enotesopen \else \@openenotes \fi
    \immediate\write\@enotes{\@doanenote{\@theenmark ,\ Pg \thepage}}
     \begingroup
       \def\next{\setFootNoteFont #1}
       \newlinechar='40
       \immediate\write\@enotes{\meaning\next}%
    \endgroup
    \immediate\write\@enotes{\@endanenote}}
%\makeatother
%
%: Notes width etc
%
%:          \notesWidth \noteGap
\newcommand{\notesWidth}{8.2cm}%
%
\newcommand{\noteGapOffset}{-0.8mm}%
%
%:          \plainNote
\newcommand{\plainNote}[1]{% no url or heading or caption pre-set, all user
```

```
controlled
\footnote{\begin{samepage}\vspace{\noteGapOffset}\setFootNoteFont\\
\parbox[t]{\notesWidth}{\ #1 \\}\setstretch{1.0}\end{samepage}}%
}%
%
%:          \urlNote
\newcommand{\urlNote}[2]{%url then Note
\href{#1}{#2}\footnote{\begin{samepage}\vspace{\noteGapOffset}
\setFootNoteFont\\\parbox[t]{\notesWidth}{\ #2 \\ {\url{#1}}}\setstretch
{1.0}\end{samepage}}%
}%
%:          \urlCaption
\newcommand{\urlCaption}[2]{%url then Caption
\href{#1}{#2}\footnote{\begin{samepage}\vspace{\noteGapOffset}
\setFootNoteFont\\\parbox[t]{\notesWidth}{\ \bf #2} \\ \url{#1}}\setstretch
{1.0}\end{samepage}%
}%
%:          \urlCaptionNote
\newcommand{\urlCaptionNote}[3]{% #1 - url, #2 - Caption, #3 -  Note
\href{#1}{#2}\footnote{\begin{samepage}\vspace{\noteGapOffset}
\setFootNoteFont\\\parbox[t]{\notesWidth}{\ {\bf #2}\\ \\{\it #3} \\ {\url
{#1}}}\setstretch{1.0}\end{samepage}}%
}%
%:          \captionNote
\newcommand{\captionNote}[2]{% #1 - Caption, #2 -  Note
\footnote{\begin{samepage}\vspace{\noteGapOffset}\setFootNoteFont\\
\parbox[t]{\notesWidth}{\ {\bf #1}\\ \\ \textit{#2}}\setstretch{1.0}\end
{samepage}}%
}%
%:          \captionCentredNote
\newcommand{\captionCentredNote}[2]{% #1 - Caption, #2 -  Note
\noindent\begin{center}{\bf #1}\captionNote{#1}{#2}
\end{center}%
}
%: -- \internalLink
%
\newcommand{\internalLink}[2]{% makes footnote text sized link to #1 {\label}
#2 {shown text}
\hyperref[#1]{\textcolor{DodgerBlue4}{\footnotesize \textsc{#2}}\ref*{#1}}
}
%: electronic or printed \livePdfPrinted{#1 - live pdf version}{#2 - printed
version}
\newcommand{\electronicPdf}{yes}% if wanted - \renewcommand
{\electronicPdf}{no} in document preamble
\newcommand{\electronicOrPrinted}[2]{
\ifthenelse{\equal{\electronicPdf}{yes}}{#1}{#2}%
}
%:
%: -- Packages --
%:
\RequirePackage{minitoc}
\RequirePackage{eso-pic}
\RequirePackage{everyshi}
```

```
\RequirePackage{pdfpages}
\RequirePackage{framed}
%: fullRef
\newcommand{\fullref}[1]{\ref{#1} on page~\pageref{#1}}%
%: Colour Bible Verses
%:              Count Verses
\newcommand{\twScriptPhp}[1]{/LaTeXPortable/LatexUtils/TeXworks/TeXworks/
config/scripts/PHPtexworks/#1}
% See script helper_twPan.mod   bibleNamesAbrvs :  [...] and macro in here %:
          \showBibleVerseCount
\newcounter{versesCount}
%:      Package bibleref
\RequirePackage{bibleref}
%:              \bibleFontText
\newcommand{\bibleFontText}[1]{``{\fontfamily{ppl}\fontsize{10.25pt}
{11pt}\selectfont \textit{#1}}''}% double quoted
%
%:              \bibleNumeral
\def\BibleVersion#1{\global\def\bibleVersion{#1}}% for 2nd \stepcounter
statement below
%
%\global\def\versionList{UNSP,WEB,NKJV,NET,RSV,OLB,NIV2011,NIV,MKJV,KJV,
JPS,ISV,AMP}%
\immediate\write18{php "\twScriptPhp{getBibleAbrsCommaList.php}" >
getBibleAbrvs.txt}
%
%\global\def\versionList{\input{getBibleAbrvs.txt}}%
%\newcommand{\versionListComma{\input{getBibleAbrvs.txt}}}%
%\global\def\versionList{\versionListComma}%
\RequirePackage{catchfile}%
\CatchFileDef{\versionList}{getBibleAbrvs.txt}{}
%
%
%\newcommand{\makeCounters}{%
\@for\val:=\versionList\do{%
\newcounter{ver\val}%
}
%}
%
\newcommand{\bibleAbr}[1]{% helper function
\\\\\\\\\\\\\\\\\\\\ #1}%
%
\newcommand{\checkVersion}[1]{% helper function
\@for\val:=\versionList\do{%
\IfSubStr{#1}{\val}{\global\let\bibleVersion\val}{}%
}
}%
%:              \addVerses
\newcommand{\addVerses}[2]{% #1 - version e.g. NIV , #2 - how many verses
% Used when \bibleNumeral type of macro can not be used for one reason or
another
\addtocounter{ver#1}{#2}
\addtocounter{versesCount}{#2}
```

```
}
\newcounter{refLength}%
%
% This one is used for when we do not want to count the verses as they have
been used already and counted in.
\newcommand{\bibleNumeralColour}[1]{\textcolor{bibleNumeral}{\textbf
{#1}}}
%:  \bibleNumeralDuplicate
\newcommand{\bibleNumeralDuplicate}[1]{\textcolor{bibleNumeral}{\textbf
{#1}}
%: index to bible
% xstrings package \IfSubStr
\IfSubStr{#1}{---}{\index{bible}{\cleanIndexEntry{#1}{-}}}{
\IfSubStr{#1}{:}{% a Bible reference - so add to bible index
\index{bible}{#1}
}{}%
}%
}%
%
% This one counts the verses used and if necessary first sets the version if that is
named in the line e.g. NIV
%
\newcommand{\bibleNumeral}[1]{\textcolor{bibleNumeral}{\textbf{#1}}%
\processNumeral{#1}%
}
%
\newcommand{\bibleNumeralPlain}[1]{\textbf{#1}%
\processNumeral{#1}%
}
%
% Helper
\newcommand{\processNumeral}[1]{
\stepcounter{versesCount}%
%: index to bible
% xstrings package \IfSubStr
\IfSubStr{#1}{---}{\index{bible}{\cleanIndexEntry{#1}{-}}}{
\IfSubStr{#1}{:}{% a Bible reference - so add to bible index
\index{bible}{#1}
}{}%
}%
% selects/sets the current counter for verse counting and remains same until
changed here again-
\checkVersion{#1}%
%
% So each call of \bibleNumeral will use current \bibleVersion to count verse
versions
\stepcounter{ver\bibleVersion}% I was surprised this worked it must fully expand -
was using a switch statement
%
}%
%:              \showBibleVerseCount  #1 - Extra un-Tabulated #2 - known
doubleups and non copyright verses
% See script helper_twPan.mod   bibleNamesAbrvs :  [...] and list in here %:
```

```
      Count Verses
\newcommand{\showBibleVerseCount}[2]{%
\addtocounter{versesCount}{#1}%
\addtocounter{versesCount}{#2}%
\noindent{\scriptsize \textit{Total Bible Verse Count Estimate for Copyright
Requirements (may include double ups):} \arabic{versesCount}%
} \\
Version---Verse Count Estimate for Copyright Requirements\\ (may include some
double ups):\\*[16pt]
\newcounter{totaledVerses}%
\@for\val:=\versionList\do{%
\addtocounter{totaledVerses}{\value{ver\val}}%
}
\newcounter{otherVersions}%
\setcounter{otherVersions}{\value{versesCount}}
\addtocounter{otherVersions}{-\value{totaledVerses}}
\begin{tabular}{rcr}
% getBibleNameAbrvasTabeLine.php
%\versionList
\@for\val:=\versionList\do{%
\ifthenelse{\value{ver\val}>0}{%
\immediate\write18{php "\twScriptPhp{getBibleNameAbrvasTabeLine.php}"  "\val"
 >  getBibleNames\val.txt}
\input{getBibleNames\val.txt}}{}}%
\ifthenelse{\value{verotherVersions}>0}{Other Versions & \dots &\arabic
{otherVersions}\\}{}
&&\uline{ \ \ \ \ \ \ }\\*[2pt]
&\textbf{Total: }&\uuline{\arabic{totaledVerses}}\\
\end{tabular}
}
%:              \bibleVerse
\newcounter{pl}%
\newcommand{\bibleVerse}[4]{% title, columns, verses, toc entry
% \definecolor{shadecolor}{named}{Ivory2} //Above\\
\definecolor{shadecolor}{named}{bibleBack}
\ifthenelse{\equal{#4}{}}{}{\phantomsection%
\addcontentsline{toc}{subsubsection}{\textsc{#4}}}
 \begin{shaded}\protect\begin{samepage}%
\begin{center}\large{\textsc{#1}}\end{center}%
\nopagebreak%
\stepcounter{pl}\label{pl-\thepl}%
\ifthenelse{\isodd{\pageref{pl-\thepl}}}%
{\dtpvpos{1mm}{-16mm}{\magScrollxiimm}}% from dtp.sty
{\dtpvpos{158mm}{-14mm}{\magScrollxiimmRev}}%
\vspace{-3mm}%
\ifthenelse{\equal{#2}{1}}{}{\begin{multicols}{#2}}%

\setstretch{0.94}\begin{verse}
{\fontfamily{ppl}\fontsize{10.25pt}{12pt}\selectfont
#3}%
\end{verse}
%
\ifthenelse{\equal{#2}{1}}{}{\end{multicols}}%
```

```
\setstretch{1}%
\protect\end{samepage}\end{shaded}%
}% end \bibleVerse
%
%:      \bibleAllUsedCopyrights    \bibleCopyright{ useabreviation from MySql/
bible_ref/bibles/Abbr}    and    \bibleAllUsedCopyrights does all were verse count
above zero
\newcommand{\bibleAllUsedCopyrights}{%
\@for\val:=\versionList\do{%
\ifthenelse{\value{ver\val}>0}{\bibleCopyright{\val}}{}% // some verses from
this bible version
}}
%
\newcommand{\bibleCopyright}[1]{
\immediate\write18{php "\twScriptPhp{doBibleInfo.php}"  "copyright" "#1" "Abbr"
 >  getScripturePermissions#1.txt}
\input{getScripturePermissions#1.txt}}
%
%:          \bibleListAllAbrvsNamesCopyrights  backup-archival purposes
\newcommand{\bibleListAllAbrvsNamesCopyrights}{

\@for\val:=\versionList\do{%
\begin{tabular}{rcr}
\immediate\write18{php "\twScriptPhp{getBibleNameAbrvasTabeLine.php}"  "\val"
 >  getBibleNames\val.txt}%
\input{getBibleNames\val.txt}
\end{tabular}\\%
%\immediate\write18{php "\twScriptPhp{doBibleInfo.php}"  "http" "\val" "Abbr"
>  getScriptureUrl.txt}
%\url{\input{getScriptureUrl.txt}}\\*[3pt]
\immediate\write18{php "\twScriptPhp{doBibleInfo.php}"  "copyright" "\val"
"Abbr"  >  getScripturePermissions\val.txt}
\input{getScripturePermissions\val.txt}
}%

}
%:  \indentBible Range
%:                \indentBible  NO extra right edge background
\newcommand{\indentBible}[1]{% indents and leaves a bit at end of block
\noindent\colorbox{bibleBack}{\hspace{0.025\columnwidth}\parbox{0.87
\columnwidth}%bodoni
{\fontfamily{anttlc}\fontsize{10.5pt}{11.25pt}\selectfont #1}\hspace{0.025
\columnwidth}}%
\vspace{-0.85mm}}
%
%:                \indentBible* with extra right edge background - #1}\hspace{0.05
\columnwidth}
% suffix package
\WithSuffix\newcommand\indentBible*[1]{% indents and leaves a bit at end of
block
\noindent\colorbox{bibleBack}{\hspace{0.05\columnwidth}\parbox{0.87
\columnwidth}{\fontfamily{anttl}\fontsize{10.5pt}{11.25pt}\selectfont #1
\vspace*{0.007\columnwidth}}\hspace{0.025\columnwidth}}%
```

```
\vspace{-0.85mm}}
%
%:              \indentAlone
\newcommand{\indentAlone}[1]{% sets up with no heading or border \indentBible
chunk/verse(s)

\vspace*{10pt}
#1
\\*[3pt]

\noindent }
%:              \indentPlain
\newcommand{\indentPlain}[1]{% indents and leaves a bit at end of block
\vspace*{1mm}%
\noindent\colorbox{plainBack}{\hspace{0.05\columnwidth}\parbox{0.87
\columnwidth}%gfsartemisia-euler ptm ppl
{\fontfamily{ptm}\fontsize{10.5pt}{11.75pt}\selectfont #1\vspace*{0.007
\columnwidth}}\hspace{0.025\columnwidth}}%
\vspace{-0.85mm}}
%
%:              \bibleHeading
\newcommand{\bibleHeading}[1]{\textbf{\textsc{\textcolor{bibleHeading}
{#1}}}}%
\index{bible}{#1}}%
%
%:      \strongsRef
%\strongsRef #1 - number, #2 - Heading/Title/Word, #3 - information
\newcommand{\strongsRef}[3]{\textnormal{\textbf{Strongs} \bibleNumeral
{#1} #2\\
\indentPlain{#3}}}
%
%:              \bibleVersePar
% {Heading-Reference}{verses - newline each}{toc entry or blank}{if toc, then
level 1,2,3 (section -> subsubsection)}{ref - NIV ...}
\def\SectionVal#1{\def\sectionVal{#1}}%
\newcommand{\bibleVersePar}[5]{
\ifthenelse{\equal{#3}{}}{}{%
\switch[\pdfstrcmp{#4}]%
\case{{1}}\SectionVal{section}%
\case{{2}}\SectionVal{subsection}%
\case{{3}}\SectionVal{subsubsection}%
\otherwise%
\endswitch%
\vspace*{0.06\columnwidth}
\phantomsection\addcontentsline{toc}{\sectionVal}{\textsc{#3}}%
}% end conditonial toc entry
\begin{boitecoloriee}\setstretch{0.92}
\noindent\colorbox{biblePanels}{\hspace{0.05\columnwidth}\parbox{0.870
\columnwidth}{\fontfamily{ppl}\fontsize{10.45pt}{12pt}\selectfont%
\begin{center}\bibleHeading{#1}\end{center}}}\nopagebreak\vspace*{-
1.5mm}\nopagebreak#2%
\setstretch{1}%
%\indentBible{\byLine{\textcolor{bibleRef}{#5}}}
```

```
\colorbox{biblePanels}{\parbox{0.92\columnwidth}{\vspace*{0.03
\columnwidth}\hfill\textcolor{bibleRef}{#5}}}
\end{boitecoloriee}\vspace*{0.06\columnwidth}\noindent%
%:  Index Bible
\index{bible}{#1}}%
%: end  \bibleVersePar
%
%: Colour Blocks
%:          \colourBlock used in \item lists
\newlength{\blockWidth}
\newcommand{\colourBlock}[3]{
% colour, width, text (width can be empty defaults to :
\ifthenelse{\equal{#2}{}}{\setlength{\blockWidth}{0.82\columnwidth}}
{\setlength{\blockWidth}{#2}}
\colorbox{#1}{\parbox[t]{\blockWidth}{#3}}}
%: indexHere
\newcommand{\indexHere}[1]{\index{#1} #1}
\DeclareGraphicsExtensions{.pdf,.png,.jpg}%
%: byLine
\newcommand{\byLine}[1]{\begin{flushright}
\small #1
\end{flushright}\\%
}%
%: Set Spaces  dimensions
%: Graphics Related
%:     \ScaleIfNeeded  http://ctan.unsw.edu.au/info/l2picfaq/german/l2picfaq.pdf
GNU doc 1.2 http://texblog.wordpress.com/category/latex/
% Usage: \includegraphics[width=\ScaleIfNeeded]{Bild}
\def\ScaleIfNeeded{%
\ifdim\Gin@nat@width>\linewidth
\linewidth
\else
\Gin@nat@width
\fi
}
%
\RequirePackage[absolute]{textpos}
%: pictOneCol
\newcommand{\pictOneCol}[4]{% {pict.png.pdf,jpg}{quote text -optional}{brief
description/caption}{copyright info}
%\rule{\columnwidth}{1pt}%
\vspace*{1mm}
\noindent
\begin{minipage}{\columnwidth}
\begin{center}
\includegraphics[width=\ScaleIfNeeded]{#1}
\\*[2mm]
\parbox[t]{0.92\columnwidth} {
\ifthenelse{\equal{#2}{}}{}{#2}%
        \begin{flushright}\urlCaptionNote{}{\textsc{\protect\small{#3}}}{#4 }
\end{flushright}}% end \parbox
\end{center}%
\vspace{-3.2mm}%
```

```
\rule{\columnwidth}{1pt}
\end{minipage}
%\vspace{-1.2mm}%
 \\*[1mm]
}
%
%: pictTwoCol
\newcommand{\pictTwoCol}[4]{% {pict.png.pdf,jpg}{quote text -optional}{brief
description/caption}{copyright info}
\noindent% \rule{\columnwidth}{1pt}
\begin{center}
\includegraphics[width=\ScaleIfNeeded]{#1}
\\*[2mm]
\begin{minipage}[t]{0.92\columnwidth}
\ifthenelse{\equal{#2}{}}{}{#2}%
        \begin{flushright}\urlCaptionNote{}{\textsc{\protect\small{#3}}}{#4 }
\end{flushright}
\end{minipage}
\end{center}
%\vspace{-1.2mm}%
%\rule{\columnwidth}{1pt}
\\*[1mm]%
}
%
%: Other
  %:  \thiswidth \thisheight
\newlength{\thiswidth}%
\newlength{\thisheight}%
\setlength{\thiswidth}{0pt}%
\setlength{\thisheight}{0pt}%
\newlength{\storedwidth}%
\newlength{\storedheight}%
%:      \makewidth
\newcommand{\makewidth}[1]{% measures but no text  insert
\setlength{\storedwidth}{\thiswidth}%
\settowidth{\thiswidth}{#1}}%
%:      \makeheight
\newcommand{\makeheight}[1]{% measures but no text  insert
\setlength{\storedheight}{\thisheight}%
\settoheight{\thisheight}{#1}}%
\newcommand{\makewidthShow}[1]{% measures and inserts text in place
\setlength{\storedwidth}{\thiswidth}%
\settowidth{\thiswidth}{#1}#1}%
%:      \makeheight
\newcommand{\makeheightShow}[1]{% measures and inserts text in place
\setlength{\storedheight}{\thisheight}%
\settoheight{\thisheight}{#1}#1}%
%:      \insertwidth{}
\newcommand{\insertwidth}[1]{
\makewidth{#1}%
\hspace*{\thiswidth}%
}%
%:      \insertheight{}
```

```
\newcommand{\insertheight}[1]{
\makeheight{#1}%
\vspace*{\thisheight}%
}%
%:      \insertstrut{}
\newcommand{\insertstrut}[1]{
\makeheight{#1}%
\mbox{0}{\thisheight}%
}%
%: \textReuse
%  #1- new command name e..g.  \tRmyAddress, #2 - text or other contents
%%  #3 - =show for first Appearance or not where you define it (must be before
first re-use)
% from then on use \tRmyAddress whereever you want it repeated
\newcommand{\textReuse}[2]{\def#1{#2}}%
%\ifthenelse{\equal{#3}{show}}{ #2}{}}% whether to show it in the
document when/where defined first time
%: mainStory
\newcommand{\mainStory}[1]{{\fontfamily{bch}\fontsize{12pt}{14pt}
\selectfont\noindent #1}}%
%: Headings
%:          centreHead
\newcommand{\centreHead}[3]{
\vspace{3mm}
{\centering\fontfamily{bch}\fontsize{#1pt}{#2pt}\selectfont\noindent #3\
\[2mm]}

\ifthenelse{#1<15}{\phantomsection\addcontentsline{toc}{subsubsection}{#3}
\subsectionmark{#3}}{}%
\ifthenelse{#1=15}{\phantomsection\addcontentsline{toc}{subsection}{#3}
\subsectionmark{#3}}{}%
\ifthenelse{#1>15}{\phantomsection\addcontentsline{toc}{section}{#3}
\sectionmark{#3}}{}%
%\phantomsection\addcontentsline{toc}{subsection}{#3}\subsectionmark{#3}
}
%:          centreHeadToc
\newcommand{\centreHeadToc}[4]{
\vspace{3mm}
{\centering\fontfamily{bch}\fontsize{#1pt}{#2pt}\selectfont\noindent #3\
\[2mm]}
% \SectionVal defined in \bibleVersePar
\ifthenelse{\equal{#3}{}}{}{%
\switch[\pdfstrcmp{#4}]%
\case{{1}}\SectionVal{section}\subsectionmark{#3}\sectionmark{#3}
\subsectionmark{#3}%
\case{{2}}\SectionVal{subsection}\subsectionmark{#3}%sub
\case{{3}}\SectionVal{subsubsection}\subsectionmark{#3} %sub
\otherwise%
\endswitch%
\phantomsection\addcontentsline{toc}{\sectionVal}%{\ifthenelse{\equal
{\pdfstrcmp{#4}}{1}}{\textsc{#3}}{#3}}%
{#3}
}% end conditonial toc entry\sectionmark{#3}}{}%
```

```
%\phantomsection\addcontentsline{toc}{subsection}{#3}\subsectionmark{#3}
}
%
%: quotePerson
\newcommand{\quotePerson}[1]{\\[\baselineskip]
 \hspace*{0.05\columnwidth}\begin{minipage}[t]{0.9\columnwidth}\fontfamily
{pplr}\fontsize{11pt}{14pt}\selectfont\noindent\linespread{1.05}  #1\end
{minipage}\vspace{1mm}}%
%from http://anthony.liekens.net/index.php/LaTeX/
SubscriptAndSuperscriptInTextMode
%: super/sub shorts CO2
\newcommand{\superscript}[1]{\ensuremath{^{\textrm{#1}}}}
\newcommand{\subscript}[1]{\ensuremath{_{\textrm{#1}}}}
\newcommand{\Cgass}{CO\subscript{2} }
\newcommand{\smallgod}{{\footnotesize `god' }}
%: addToc
\newcommand{\addToc}[2]{
\phantomsection
\ifthenelse{\equal{#2}{1}}{\phantomsection\addcontentsline{toc}{section}
{#1}
\sectionmark{#1}}{}
\ifthenelse{\equal{#2}{2}}{\phantomsection\addcontentsline{toc}{subsection}
{#1}\subsectionmark{#1}}{}
\ifthenelse{\equal{#2}{3}}{\phantomsection\addcontentsline{toc}
{subsubsection}{#1}\subsubsectionmark{#1}}{}
}
%: magfollow
\newcommand{\magfollow}{``Following Jesus''}
%
%: newCols
\newcommand{\newCols}[1]{\end{multicols}
\begin{multicols}{#1}}
%: centreRule
\newcommand{\centreRule}{
\begin{center}\rule{7cm}{1pt}\end{center}
\\}
%: boxCentred #1 text #2 Colour
\newcommand{\boxCentred}[2]{\vspace{1mm}
\begin{center}\protect\framebox[0.797\textwidth][t]{\hspace*{0mm}{\fboxsep
3mm\colorbox{#2}{\parbox{0.75\textwidth}{\fontfamily{bch}\fontsize{12pt}
{14pt}\selectfont\textit #1\\*[1mm]}}}}
\vspace*{2mm}\end{center}}
%
\RequirePackage{dtp}% local sty some Destop Publishing macros
%\RequirePackage{kdgreek}
%: Greek Stuff
%: ==font stuff
%\usepackage{palatinox}
\usepackage{iwona}%kurier
%-------------------------------------------------
% Greek Keyboard Shortcuts
\newcommand {\gt}{\greektext} %Set the language from now on
\newcommand {\lt}{\latintext}
```

```
\newcommand {\tg}{\textgreek} %Next argument will be the language you set.
\newcommand {\tl}{\textlatin}
%: == font package
\RequirePackage{mathptmx} % Times
%: \showGreek \bothGreek \fontfamily{kurier}\fontencoding{OT2}\selectfont
\newcommand{\showGreek}[1]{{\usefont{OT2}{iwona}{m}{b}\greektext
#1}}
\newcommand{\bothGreek}[1]{#1 -- \showGreek{#1}}%{\usefont{OT2}
{iwona}{m}{b}\greektext #1}
%: Cardinals
% #1 Heading - can be blank, cardinals #2 - text, #3 - byline, #4 Img, #5 -
ImgRev
\newcommand{\giftS}{UK Cardinals' \textit{``The Gift of Scripture''} }
\newcounter{cardnl}%
\newcommand{\cardinals}[5]{\setlength{\columnseprule}{1pt}%
\setstretch{0.92}%
\begin{quote}\begin{multicols}{2}%
\ifthenelse{\equal{#1}{}}{}{
{\centering\fontfamily{bch}\fontsize{13pt}{15pt}\selectfont\noindent #1\
\[2mm]}
\phantomsection\addcontentsline{toc}{subsubsection}{#1}\subsectionmark
{#1}}%
%
%
\stepcounter{cardnl}\label{cardnl-\thecardnl}%
\ifthenelse{\isodd{\pageref{cardnl-\thecardnl}}}%
{\dtpvpos{159mm}{-2mm}{#5}}%
{\dtpvpos{-16mm}{2mm}{#4}}% from dtp.sty
#2
\ifthenelse{\equal{#3}{}}{\byLine{\textcolor{red}{ref needed}}}{ \byLine
{{\fontsize{9.5pt}{12pt}\selectfont#3}}}%
\end{multicols}\end{quote}
\setstretch{1}
\setlength{\columnseprule}{0pt}%
}
%\newcommand{\cardinals}[5]{#1 #2 #3 #4 #5 } % for debugging


%: goodQuote Environement
\newenvironment{goodQuote}{\definecolor{shadecolor}{named}{LightYellow1}
\begin{shaded}\begin{multicols}{2}\begin{quote}}{\end{quote}\end
{multicols}\end{shaded}}
%: various adjustment settings
%: 1. margin notes
\setlength{\marginparsep}{15mm}
\reversemarginpar
%: 2. Widows orphans
\widowpenalty=300
\clubpenalty=300
%: Shading
%:   Colours as \defines and \commands
%: table shades
\definecolor{tableiTunes}{HTML}{F1F5FA}   %iTunes
```

```
\definecolor{tableFinder}{HTML}{ECF3FE} %Finder
%\def\rangeGray{99}
\definecolor{bibleBack}{HTML}{ECF3FE}%{gray}{0.89}%0.98964
\definecolor{plainBack}{rgb}{0.94901, 0.94901, 0.92156} % HTML CCCC99
Ivory1
\definecolor{bibleParaMark}{named}{IndianRed4}
%\definecolor{bibleNumeral}{named}{DarkSeaGreen4}
\definecolor{bibleNumeral}{named}{SteelBlue4}
\definecolor{bibleHeading}{named}{DarkSeaGreen4}
\definecolor{bibleRef}{named}{DarkSeaGreen4}
\definecolor{biblePanels}{rgb}{0.91372, 0.94117, 1}
% http://www.cv-templates.info/2009/07/alternate-row-shading-latex/
\newcommand{\colourUN}{AliceBlue}
\newcommand{\colourBad}{MistyRose1}
%: \backColourBlock   #1 - Colour (named) , #2 - Text Block
\newcommand{\backColourBlock}[2]{\definecolor{shadecolor}{named}{#1}
\begin{shaded}
#2
\end{shaded}}
%
%: Logos and placement
%
%: Placement
\newcounter{placelogo}\setcounter{placelogo}{1}%
%
\newcommand{\placeLogo}[4]{% #1 odd pages logo, #2 odd x axis, #3 even
pages logo, #4 even x axis
\stepcounter{placelogo}\label{here-\theplacelogo}%
\ifthenelse{\isodd{\pageref{here-\theplacelogo}}}%
{\ifthenelse{\equal{#2}{}}{\dtpvpos{-3mm}{-12mm}{#1}}{\dtpvpos{-
8mm}{#2}{#1}}}%
{\ifthenelse{\equal{#4}{}}{\dtpvpos{-3mm}{-12mm}{#3}}{\dtpvpos{-
8mm}{#4}{#3}}}% from dtp.sty
}% end \placeLogo
%
%: Logos
\newcommand{\magCross}{\includegraphics[scale=0.8]{magStandard-Cross-
1cm.jpg}}
\newcommand{\magScroll}{\includegraphics[scale=1]{magStandard-Scroll-1cm-
WhiteBckGrnd.jpg}}
\newcommand{\magScrollxiimm}{\includegraphics[scale=0.9]{bibleScroll12mm.
png}}
\newcommand{\magScrollxiimmRev}{\reflectbox{\includegraphics[scale=0.9]
{bibleScroll12mm.png}}}
\newcommand{\magCardinal}{\includegraphics[scale=0.9]{magStandard-
Cardinal.png}}
\newcommand{\magCardinalRev}{\reflectbox{\includegraphics[scale=0.9]
{magStandard-Cardinal.png}}}
\newcommand{\magUN}{\includegraphics[scale=0.9]{UN-Flag-Thumbnail.png}}
\newcommand{\magRome}{\includegraphics[scale=1]{magStabdard-popeXIII.
jpg}}
\newcommand{\magRomeRev}{\reflectbox{\includegraphics[scale=1]
{magStabdard-popeXIII.jpg}}}
```

```
%: boite modification
\RequirePackage{pstricks}
%\newcolour{gris}{0.9}
\def\boitecoloriee{%
  \def\bkvz@before@breakbox{\ifhmode\par\fi\vskip\breakboxskip\relax}%
  \def\bkvz@set@linewidth{\advance\linewidth -2\fboxrule
    \advance\linewidth -2\fboxsep} %
  \def\bk@line{\hbox to \linewidth{%
      \ifbkcount\smash{\llap{\the\bk@lcnt\ }}\fi
      \psframebox*[framesep=0pt,fillcolor=bibleBack,linewidth=0pt]{%
        \vrule\@width\fboxrule \hskip\fboxsep
        \box\bk@bxa
        \hskip\fboxsep \vrule\@width\fboxrule
        }%
      }}%
  %\def\bkvz@top{\hrule\@height\fboxrule}
  \def\bkvz@top{\hrule height .6pt}%%% Pourquoi faut-il ajouter 0.2pt ???
  \def\bkvz@bottom{\hrule\@height\fboxrule}%
  \breakbox}
\def\endboitecoloriee{\endbreakbox}
%
%: Specific Characters etc ...
%\DeclareUnicodeCharacter{002C}{\coma}
% "Cannot define Unicode char value < 00A0."
\newcommand{\coma}{, }%
\newcommand{\kgk}{\textit{Koiné} Greek }
% http://www.technovelty.org/linux/tips/latex-tick.html
%\usepackage{amsfonts}
\newcommand{\tickYes}{\checkmark}
\usepackage{pifont}
\newcommand{\tickNo}{\hspace{1pt}\ding{55}}
%: counters etc
%: table row
\newcounter{tableRowx}
\newcommand{\startTablerowx}{\setcounter{tableRowx}{-1}}%
\newcommand{\tablerowx}{\stepcounter{tableRowx}%
\ifthenelse{\value{tableRowx} < 1}{}{% next insert space if < 10
\ifthenelse{\value{tableRowx} < 10}%
{ \ \arabic{tableRowx}. \ \ }%
{ \ \arabic{tableRowx}. \ }%
}% end of value{tableRowx} < 1
}
%: pdfcomments
%:                    \showComment
\newcommand{\showComment}[2]{#1\footnote{#1---#2} \hspace*{2mm}
\pdfcomment[color=bibleBack,icon=Comment,author={#1},hoffset={-5mm}]
{#2}%
}% /End.
%: Indexes
\RequirePackage{multind}
\makeindex{general}
\makeindex{bible}
\newcommand{\printIndicesHere}{%
```

```
\printindex{general}{General index}%

\printindex{bible}{Scripture index}%

}
%: \indexShow #1 index name e.g. bible, #2 index contents (will be shown in doc
as well))
\newcommand{\indexShow}[2]{%
#2\index{#1}{#2}%
%
}%
\newcommand{\cleanIndexEntry}[2]{
\protect\StrDel{#1}{#2}}
\newcounter{bibleRef}%
\newcommand{\checkLength}[2]{
\setcounter{bibleRef}{StrLen{#1}}%
\protect\ifthenelse{\protect\value{bibleRef} > #2}{#1}{}}
%: File Out
%: \setupKeep #1 filename
\newcommand{\setupKeep}[1]{
\newwrite\file
\immediate\openout\file=#1.keep
}
%: \finishupKeep
\newcommand{\finishupKeep}{
\closeout\file}
%: \keepTrack #1 filename no .extn, #2text
\newcommand{\keepTrack}[1]{
\immediate\write\file{#1}
}
\endinput
%
%\newcounter{verNIV}% version verse counting
%\newcounter{verWEB}%
%\newcounter{verNET}%
%\newcounter{verISV}%
%\newcounter{verAMP}%
%\newcounter{verRSV}%
%
```

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

getBibleAbrsCommaList.php

```
<?php

//$orderBy = 'title';
//$orderBy = 'cite_key';

//  $orderBy = $argv[1];

// Connecting, selecting database
$link = mysql_connect('localhost', 'root', '')
```

```
   or die('Could not connect: ' . mysql_error());
// echo 'Connected successfully';

mysql_set_charset('utf8');

mysql_select_db('bible_ref') or die('Could not select database');

// Performing SQL query
// $query = 'SELECT cite_key title FROM entries';

$query = "SELECT Abbr FROM bibles order by Abbr" ; //. $orderBy ;

$result = mysql_query($query) or die('Query failed: ' . mysql_error());

 $build = "";

while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {

   foreach ($line as $col_value) {
     $build .=  trim($col_value) .  ",";
   }
}

// Free resultset
mysql_free_result($result);
// Closing connection
mysql_close($link);

  $build = trim(substr($build, 0, -1)); // clean off last comma and cleanse

echo $build;

?>
```

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

doBibleInfo.php

```
<?php

// echo "Hi from Php for TeXworks Latex  $argv[1] $argv[2]"

  $column_to_Search      = $argv[3];
  $list_items = explode(",",$argv[2]);
  $column_to_return      = $argv[1];

  $imbetween = "\\begin{center}\n\\rule{0.71\\columnwidth}{1pt}\n\\end
{center}\n";

$link = mysql_connect('localhost', 'root', '')
   or die('Could not connect: ' . mysql_error());
// echo 'Connected successfully';
```

```
mysql_set_charset('utf8');

mysql_select_db('bible_ref') or die('Could not select database');
  $build = "";


  // Performing SQL queries

  foreach ( $list_items as $this_item)
 {

  $build .=  "\\noindent\\textbf{" .  $this_item . "} -- ";

  $query = "SELECT " . trim($column_to_return). " FROM bibles where " . trim
($column_to_Search)  . " = '"
           . trim($this_item) . "' " ;

 //      echo $query . "\n\n";
  // $query = 'SELECT CONCAT(cite_key, \',  \', title) FROM entries order by ' .
$orderBy ;

   $result = mysql_query($query) or die('Query failed: ' . mysql_error());



  while ($line = mysql_fetch_array($result, MYSQL_ASSOC))
  {

   foreach ($line as $col_value)
    {
      $build .=  trim($col_value) .  $imbetween;
    }
  }

// Free resultset
   mysql_free_result($result);

 }
// Closing connection
mysql_close($link);

 $build = trim($build); // clean off last linefeed else blank option in dropdown results

echo $build;

// Thanks to  http://www.tech-recipes.com/rx/2059/
mysql_use_concat_to_include_text_in_select_results/


?>
```

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

Filling DropDowns with directory listings.

Another example is filling a dropdown (select) box. in Qt QCombobox, with a directory box.

Under Windows this could look like this...

```
var fileList = twPan.osCmd("cmd /c dir /b \""
                + pathDir + "\\*" + fileExtension + "\""
                , true)
     removeExtnsn = new RegExp(fileExtension,"ig");
  var  fileList = fileList.replace(removeExtnsn,"").split(lineBreak);
  var chosenAnswer = twPan.select(Description, fileList);
```

Created with the Personal Edition of HelpNDoc: Free help authoring tool

Build in Update Chercking

It is possible to think through ways of including methods people can use to check for any new Update for your script if you are giving it away, or it is on an organisations local web server.

Although direct access is not given to the web, TeXworks does provide a system command for launching files by address. (There is no guarantee that a system call to wget or like will work on all systems.)

Provide an option on your form, or in a drop down box or other for the User to be able to check for a new version.

You can call a php script on your local (organisation) or www server passing variables to it in the html query, and show a page that maintains an associative array that determines
whether the script has a new version for download or not and
displays an appropriate message to the User. This can also be achieved without server side scripting, in the browser by inspecting the window.location.search property in client-side JavaScript.

```
       var thisFileName =  TW.script.fileName.substr(
                           TW.script.fileName.lastIndexOf("/") +1
                                                  );

   var checkFile = "checkFile.php?script=" + thisFileName
                   + "&version=" + TW.script.version;

   var whatHappened =  TW.launchFile(
                          "http://twscript.paulanorman.info/
downloads/"
                          + checkFile
                                    );

      if (whatHappened.status != 0) // call failed
```

```
    {
     TW.information("What Happened", whatHappened['message']);
     } //
```

Or a more basic approach is to point to a down load page with version information already shown. This would mean showing the User what the script name and version number were perhaps as part of any form, drop-down (getItem) or Information box being presented to them.

```
var thisOne = TW.getItem(null, "Please Choose",  "This is: " +
TW.script.title
                          +  " version: <b>" + TW.script.version
                          +"</b>  "+ thisFileName

//  Not available         + "<br> Dated : " + TW.script.date
                              + "<br> Please Check for Updates
at\n   <span style='color:navy; text-decoration: underline;
cursor:none' title='Choose Update from Drop Box'>twscript.
PaulANorman.info</span>"

                             +"<br><br>Please Choose..."
                             , mainList // mainList is an array
                             , 2
                             , false
                             );
```

Then something like...

```
    var whatHappened =  TW.launchFile("http://twscript.paulanorman.
info/downloads/");

     if (whatHappened.status != 0) // call failed
       {
        TW.information("What Happened", whatHappened['message']);
        } //
```

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Tw Api functions and properties

This will include additions by Qt to the standard EMCA feature set. (The EMCA standard describes how this is to be done)

Still being worked on, tree skeleton being filled out first.

Skeleton

These sections attempts to itemise by categroy, the objects, properties and fuctions appearing in the Qt script debugger, and tries to explore the useful ones.

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

File Functions, Procedures and Properties

File Functions, Procedures and Properties are distributed across the Api tree.

Top level

__FILE__ script file path and name

TW (normally requiring a prior security setting by the User in Edit/Preferences/ Script)—

- File Functions
  - .readFile()
  - .writeFile()
  - .launchFile(QString)
  - .launchFile(QString,bool)
  - TW.app.openFileFromScript(QString,TW,int,bool)

  - Return Values for Status

TW.app (not specific to the current document)

- Files
  - .getOpenFileName()
  - .getOpenFileName(QString)
  - .getOpenFileNames()
  - .getOpenFileNames(QString)
  - .getSaveFileName(QString)
  - .newFile()
  - .open()
  - *.openFile(QString,int)*
  - *.recentFileActionsChanged()*
  - .updateRecentFileActions()

TW.target (current editing document)

- File Related
  - .fileName
  - .newFile()
  - .newFromTemplate()
  - .open()
  - .removeAuxFiles()
  - .revert()
  - .save()
  - .saveAll() (all documents)
  - .saveAs()
  - .windowFilePath

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

JSON

Before anything else, but this in your script tester:

```
a = {"HOUSE":"cat", "ROOM": "kitchen"};
b = {"HOUSE":"dog"};

list = [a,b];

for (x in list)
  {
    for (z in list[x])
       {
          TW.information(null, "test", z + " = " + list[x][z]);
       }
  }
```

Here are some notes:

http://www.json.org/js.html

JSON is a subset of the object literal notation of JavaScript. Since JSON is a subset of JavaScript, it can be used in the language with no muss or fuss.

```
var myJSONObject = {"bindings": [
        {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.
*"},
        {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex":
"^delete.*"},
        {"ircEvent": "PRIVMSG", "method": "randomURI", "regex":
"^random.*"}
    ]
};
```

In this example, an object is created containing a single member "bindings", which contains an array containing three objects, each containing "ircEvent","method", and "regex" members.

Members can be retrieved using dot or subscript operators.

```
myJSONObject.bindings[0].method    // "newURI"
```

To convert a JSON text into an object, you can use the eval() function. eval () invokes the JavaScript compiler. Since JSON is a proper subset of JavaScript, the compiler will correctly parse the text and produce an object structure. The text must be wrapped in parens to avoid tripping on an ambiguity in JavaScript's syntax.

```
var myObject = eval('(' + myJSONtext + ')');
```

The eval function is very fast. However, it can compile and execute any JavaScript program, so there can be security issues. The use of eval is indicated when the source is trusted and competent. It is much safer to use a JSON parser. In web applications over XMLHttpRequest, communication is permitted only to the same origin that provide that page, so it is trusted. But it might not be competent. If the server is not rigorous in its JSON encoding, or if it does not scrupulously validate all of its inputs, then it could deliver invalid JSON text that could be carrying dangerous script. The eval function would execute the script, unleashing its malice.

To defend against this, a JSON parser should be used. A JSON parser will recognize only JSON text, rejecting all scripts. In browsers that provide native JSON support, JSON parsers are also much faster than eval. It is expected that native JSON support will be included in the next ECMAScript standard.

```
var myObject = JSON.parse(myJSONtext, reviver);
```

The optional `reviver` parameter is a function that will be called for every key and value at every level of the final result. Each value will be replaced by the result of the `reviver` function. This can be used to reform generic objects into instances of pseudoclasses, or to transform date strings into `Date` objects.

```
myData = JSON.parse(text, function (key, value) {
    var type;
    if (value && typeof value === 'object') {
        type = value.type;
        if (typeof type === 'string' && typeof window[type] ===
'function') {
            return new (window[type])(value);
        }
    }
    return value;
});
```

A JSON stringifier goes in the opposite direction, converting JavaScript data

structures into JSON text. JSON does not support cyclic data structures, so be careful to not give cyclical structures to the JSON stringifier.

```
var myJSONText = JSON.stringify(myObject, replacer);
```

If the stringify method sees an object that contains a toJSON method, it calls that method, and stringifies the value returned. This allows an object to determine its own JSON representation.

The stringifier method can take an optional array of strings. These strings are used to select the properties that will be included in the JSON text.

The stringifier method can take an optional replacer function. It will be called after the toJSON method (if there is one) on each of the values in the structure. It will be passed each key and value as parameters, and this will be bound to object holding the key. The value returned will be stringified.

Values that do not have a representation in JSON (such as functions and undefined) are excluded.

Nonfinite numbers are replaced with null. To substitute other values, you could use a replacer function like this:

```
function replacer(key, value) {
    if (typeof value === 'number' && !isFinite(value)) {
        return String(value);
    }
    return value;
}
```

Giving a corresponding reviver to JSON.parse can undo that.

Created with the Personal Edition of HelpNDoc: Full featured Help generator

parse

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

stringify

JSON.stringify(myObject, replacer)

The replacer function may be omitted. If present it needs to look something like

```
function replacer(key, value) {
    if (typeof value === 'string') {
        return '\\cite{value}';
    }
    return value;
}
```

var myJSONText = JSON.stringify(myObject, replacer);

Created with the Personal Edition of HelpNDoc: Full featured Help generator

__FILE__

__FILE__

(Not **TW.**__FILE__ no -> TW.)

(This is less reliable than Script properties (fileName etc) as otherwise it may require the User to set the Debugger to be "on" to work.)

So script property TW.script.**fileName** now depreciates __FILE__

This holds the path and file name of the current script that is running.
For e.g.

> "D:/latexportable/LaTexUtils/TeXWorks/texworks/config/scripts/Scripting - Experiments/debugger.js"

Not the document that is being edited (for that see TW.target.fileName).

You can use JavaScript string functions on this to for example get the path of the script folder that you are in.

```
__FILE__.substr(0, __FILE__.lastIndexOf("/") +1)
```

In the above example would yield:

> "D:/latexportable/LaTexUtils/TeXWorks/texworks/config/scripts/Scripting - Experiments/"

Created with the Personal Edition of HelpNDoc: Free Web Help generator

TW

The TW object

The TW object directly handles the top level interactions with the outside world, input, message and user made dialogue interfaces.

And through .findChildWidget(QWidget*,QString), .app and .target the application interface and editing windows(s)

Conceptually TW is a bit like window in a browser Api, for things like:

- window.alert() — message boxes,
- window.confirm() — message boxes,
- window.input() — get User Input and
- window.open() — createUI dialogues.

Checking project source directory under Windows with this script ...

```
cls
```

```
@echo off
echo.
echo findInCpp.bat
echo.
echo usage example: findInCpp close()
echo.
echo.
echo ----===---
echo.
echo  %1  -----=
echo.
echo.
echo.
echo.
echo.
find /i /n "%1" *.cpp
echo.
echo.
echo ::%1  -----=
echo.
find /i /n "void %1" *.h
echo.
echo.
```

Created with the Personal Edition of HelpNDoc: Free help authoring environment

Default Conversion from Qt Script to C++

Default Conversion from Qt Script to C++

http://doc.qt.nokia.com/4.4/qtscript.html#default-conversion-from-qt-script-to-c

C++ Type	Default Conversion
bool	QScriptValue::toBoolean()
int	QScriptValue::toInt32()
uint	QScriptValue::toUInt32()
float	float(QScriptValue::toNumber())
double	QScriptValue::toNumber()
short	short(QScriptValue::toInt32())
ushort	QScriptValue::toUInt16()
char	char(QScriptValue::toInt32())
uchar	unsigned char(QScriptValue:: toInt32())
qlonglong	qlonglong(QScriptValue:: toInteger())
qulonglong	qulonglong(QScriptValue::

	toInteger())
QString	QScriptValue::toString()
QDateTime	QScriptValue::toDateTime()
QDate	QScriptValue::toDateTime().date()
QRegExp	QScriptValue::toRegExp()
QObject*	QScriptValue::toQObject()
QWidget*	QScriptValue::toQObject()
QVariant	QScriptValue::toVariant()
QChar	If the QScriptValue is a string, the result is the first character of the string, or a null QChar if the string is empty; otherwise, the result is a QCharconstructed from the unicode obtained by converting the QScriptValue to a `ushort`.
QStringList	If the QScriptValue is an array, the result is a QStringList constructed from the result of QScriptValue::toString() for each array element; otherwise, the result is an empty QStringList.
QVariantList	If the QScriptValue is an array, the result is a QVariantList constructed from the result of QScriptValue::toVariant() for each array element; otherwise, the result is an empty QVariantList.
QVariantMap	If the QScriptValue is an object, the result is a QVariantMap with a (key, value) pair of the form (propertyName, propertyValue.toVariant()) for each property, using QScriptValueIterator to iterate over the object's properties.
QObjectList	If the QScriptValue is an array, the result is a QObjectList constructed from the result of QScriptValue::

	toQObject() for each array element; otherwise, the result is an empty QObjectList.
QList<int>	If the QScriptValue is an array, the result is a QList<int> constructed from the result of QScriptValue::toInt32() for each array element; otherwise, the result is an empty QList<int>.

Additionally, QtScript will handle the following cases:

- If the QScriptValue is a QObject and the target type name ends with * (i.e., it is a pointer), the QObject pointer will be cast to the target type with qobject_cast().
- If the QScriptValue is a QVariant and the target type name ends with * (i.e., it is a pointer), and the userType() of the QVariant is the type that the target type points to, the result is a pointer to theQVariant's data.
- If the QScriptValue is a QVariant and it can be converted to the target type (according to QVariant::canConvert()), the QVariant will be cast to the target type with qvariant_cast().

Default Conversion from C++ to Qt Script

The following table describes the default behavior when a QScriptValue is constructed from a C++ type:

C++ Type	Default Construction
void	QScriptEngine::undefinedValue()
bool	QScriptValue(engine, value)
int	QScriptValue(engine, value)
uint	QScriptValue(engine, value)
float	QScriptValue(engine, value)
double	QScriptValue(engine, value)
short	QScriptValue(engine, value)
ushort	QScriptValue(engine, value)
char	QScriptValue(engine, value)
uchar	QScriptValue(engine, value)
QString	QScriptValue(engine, value)
qlonglong	QScriptValue(engine, qsreal (value)). Note that the conversion may lead to loss of precision, since not all 64-bit

	integers can be represented using the qsreal type.
qulonglong	QScriptValue(engine, qsreal (value)). Note that the conversion may lead to loss of precision, since not all 64-bit unsigned integers can be represented using the qsreal type.
QChar	QScriptValue(this, value. unicode())
QDateTime	QScriptEngine::newDate(value)
QDate	QScriptEngine::newDate(value)
QRegExp	QScriptEngine::newRegExp (value)
QObject*	QScriptEngine::newQObject (value)
QWidget*	QScriptEngine::newQObject (value)
QVariant	QScriptEngine::newVariant (value)
QStringList	A new script array (created with QScriptEngine::newArray ()), whose elements are created using the QScriptValue (QScriptEngine *, QString) constructor for each element of the list.
QVariantList	A new script array (created with QScriptEngine::newArray ()), whose elements are created using QScriptEngine::newVariant () for each element of the list.
QVariantMap	A new script object (created with QScriptEngine::newObject ()), whose properties are initialized according to the (key, value) pairs of the map.
QObjectList	A new script array (created with QScriptEngine::newArray ()), whose elements are created using QScriptEngine:: newQObject() for each element

	of the list.
QList<int>	A new script array (created with QScriptEngine::newArray ()), whose elements are created using the QScriptValue (QScriptEngine *, int) constructor for each element of the list.

Other types (including custom types) will be wrapped using QScriptEngine::newVariant(). For null pointers of any type, the result is QScriptEngine::nullValue().

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

Spell Checking

http://code.google.com/p/texworks/issues/detail?id=563

Comment 1 by project member st.loeffler, Mar 4, 2012

Sounds intriguing. In r962, I have implemented a couple of functions that should be helpful:
TW.app.reloadSpellchecker() does what the name suggests - it reinitializes the spell checker for all open windows (thereby reloading the dictionaries, among other things)

TW.getDictionaryList(const bool forceReload = false) returns a map ("object" in QtScript terms) of the form "language code => array(filenames)". ATM, array(filenames) has exactly one entry, but this may change in the future. To iterate over the map, use something like

```
dicts = TW.getDictionaryList(false);
for (d in dicts) {
   var lang = d;
   var file = dicts[d][0];
}
```

If forceReload=true, the list of available dictionaries is refreshed (note that this doesn't affect any loaded spell checker, but does rebuild the "Edit > Spelling" menus).

TW.target.spellcheckLanguage is a new property that can be read from (to get the current language) and written to (to set the spell checking language). It's probably easier to use than TW.target.setSpellcheckLanguage().

TW.target.reloadSpellcheckerMenu() is new, but not really useful for scripts (it rebuilds the "Edit > Spelling" menus, but TW.getDictionaryList(true) does that automatically, and otherwise there's not much point in calling it.

Finally, a few words of caution:
 * On most *nix systems, the dictionaries are considered system resources and cannot be changed (without acquiring super-user privileges). So adding to the .dic files from a script won't be possible. As a workaround, one could maintain a copy of the dictionaries one uses in a location where they can be edited and use the TW_DICPATH environment variable to point Tw to it.
 * There currently is no persistent private word list kept by Tw (there's only the temporary, session-specific word list that is used by the "Ignore word" context menu item). This will hopefully be changed in the future

(0.8 is the target), but entails a broader rewrite of the spell checking implementation.

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

getDictionaryList(const bool forceReload = false)

See Spell Checking

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

File/System Functions with User Controlled Security

System and File Functions with User Controlled Security

These functions are controlled on the Edit/Preferences Menu

Read and Write outside a script's own (sub)-directory tree, active document, and master document (if being used) must be turned on first by the User.

There are functions for testing current settings to establish whether the User has enabled the necessary permissions already, and for presenting the permissions dialogue for the User to take any enabling action.

Once the User has turned any Read or Write functions on or off, the setting is maintained between sessions until explicitly altered by the User.

Use with care a write or system call could damage a User's system.

For current document's (being edited) path and file name see TW.target.fileName

For current script's (being run) path and file name see TW.script.fileName

Note: Script property TW.script.**fileName** now depreciates __FILE__

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

readFile()

TW.readFile(QString)

var **fileResult** = TW.readFile("path/to/file.txt");

```
relative paths are taken to be relative to the folder containing the
executing script's file
```

The return value **fileResult** is an object (a map) with three member properties.

(You can use any valid variable name and not just ***fileResult)***

status, result, and message

status

fileResult.**status**

fileResult['status']

gives you the status code (same as for writeFile).
Upon success res.status == 0

See Return Values for Status
result

fileResult.**result** contains the text of the file.
This would be undefined if there is no content.

message

fileResult.**message** will have one of the following responses in it:

1. "Reading all files is disabled (see Preferences)"
2. "The file \"%1\" could not be opened for reading"

And upon successful reading of file content:
3. undefined

Lua

In lua, ***fileResult*.status**

And so on.

Python

In Python, the members are accessed as a dictionary ***fileResult*['status']**

And so on.

Permissions

There is a setting for 'reading' permissions in the preferences dialog.

By default, global reading is disabled.

Files in the folder containing the
current script (e.g., <resources>/scripts/test/) and its sub folders are always readable, as are files in the folder containing the current target (tex/pdf) file's directory, its root (master) file's directory (if used), and their sub folders.

Reference: Stefan Löffler http://tug.org/mailman/htdig/texworks/2010q4/003471.html
Stefan Löffler http://tug.org/pipermail/texworks/2010q4/003531.html
cpp TWScriptAPI::readFile

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

writeFile()

Use with care a write or system call could damage a User's system.

TW.writeFile(QString filename, QString content)

filename

Is just that.

```
relative paths are taken to be relative to the folder containing the
executing script's file
```

Other wise you must give a fully qualified path - note if give a path outside the scripts own (sub)-directory tree, or outside the current document or its master (root) document's (sub)-directory tree, the User will need to first have turned on file writing in the Edit/Preferences/Scripting dialogue tab.

You must provide the file extension as a part of the **filename**

content

Writing is performed in utf8 encoded text-mode.

Return Value

writeFile returns an integer directly (not an object like readFile does)

The integer is the same as readFile's "status" value see Return Values for Status

Use with care a write or system call could damage a User's system.

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

launchFile(QString)

TW.launchFile(QString pathAndFileName)

Set full path and file name.
Returns an object map with two member properties:

"status" and "message".

There no arguments (options or switches) can be supplied see no variation will work.

This function is controlled by the Edit /Preferences/ Scripts - checkbox "allow scripts to run system commands".
Which is "off" by default.

Any directory (as pathAndFileName) can be opened no matter what the security setting is.

e.g. both of these work

```
var whatHappened = TW.launchFile("c:/TexProjects/writing");
var whatHappened = TW.launchFile("c:/TexProjects/writing/");
```

And local or web:—

```
var whatHappened = TW.launchFile("http://PaulANorman.info");
```

For a supplied filename, if the user has enabled System commands, then a file can be 'opened' or 'launched' by its system associated executable.

1. If a document filename is supplied, it will be opened by what ever is set to handle the file type.
2. If an executable's filename is supplied to the function the executable will be started.

1. On windows under normal default settings, *notepad.exe* might be expected to open this document.

```
var whatHappened =  launchFile("c:/to/somewherre/myFile.txt");
```

2. On windows under normal default settings, this usage would launch *notepad.exe* with a blank document showing.

```
var whatHappened =  launchFile("c:/windows/notepad.exe");
```

This type of thing will not currently work under any variation:

```
var whatHappened =  TW.launchFile("c:/windows/notepad.exe c:/hold/MyFile.txt"
```

Return Values

The function returns an object with two member properties:

"status" and "message"

The following tables list the return values for **whatHappened**

whatHappened["status"] —

Condition	Integer returned	Value
on success	SystemAccess_OK	0
on file access failure (e.g. wrong name supplied)	SystemAccess_Failed	1
if Preference Permission not set by User yet	SystemAccess_PermissionDenied	2

whatHappened["message"] —

Condition	String Returned
on success	a blank message - empty string
on file access failure (e.g. wrong name)	""c:/to/somewherre/myFile.txt" could not be opened."
if Preference Permission not set by User yet	"System command execution is disabled (see Preferences)"

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

launchFile(QString,bool)

Overlaod version with boolean, removed late 2010. See launchFile(QString)

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

TW.app.openFileFromScript(QString,TW,int,bool)

1. TW.**app**.openFileFromScript(fullFileName : String , TW)
2. TW.**app**.openFileFromScript(fullFileName : String, TW, fileLinePosition : integer)
3. TW.**app**.openFileFromScript(fullFileName : String, TW, fileLinePosition : integer, ask user - if file access permission is not set: bool)

Opens a File in Tw for editing.

Will read inside script or document directory, other wise needs File access permissions set, but can ask user on occasion if told to (True/False) in the third version of calling the function

Required - fullFileName a fully/qualified/path to the file contents wanted.

The position held by TW here is actually a scriptApiObj.

Line number position (-1 for none), in the second version of calling the function, allows the line position of the file to be set.

Returns a response object with members:

"status" as for other file open functions Return Values for Status

"result" is the opened Document's scripting object with all the properties and functions of **TW.**target

---

Example:

```
  retVal = TW.openFileFromScript("c:/somewhere/Mydocument.txt", TW, -1,
true);

 if  (retVal.status == 0) // SystemAccess_OK
     {
       var openedDocumentObject = retVal["result"] ;

       TW.information(null,"Tw Message",openedDocument.fileName); //
shows the opened document's base filename
       TW.information(null,"Tw Message",openedDocument.text); // shows
the contents of the "opened" document.

       }
```

Edit notes: Extracting from here

```
//Q_INVOKABLE
QMap<QString, QVariant> TWApp::openFileFromScript(const QString& fileName,
QObject * scriptApiObj, const int pos /*= -1*/, const bool askUser /*= false*/)
{
	QSETTINGS_OBJECT(settings);
	QMap<QString, QVariant> retVal;
	QObject * doc = NULL;
	TWScript * script;
	QFileInfo fi(fileName);
	TWScriptAPI * scriptApi = qobject_cast<TWScriptAPI*>(scriptApiObj);

	retVal["status"] = TWScriptAPI::SystemAccess_PermissionDenied;

	// for absolute paths and full reading permissions, we don't have to care
	// about peculiarities of the script; in that case, this even succeeds
	// if no valid scriptApi is passed; otherwise, we need to investigate further
	if (fi.isRelative() || !settings.value("allowScriptFileReading", false).toBool()) {
		if (!scriptApi)
			return retVal;
		script = qobject_cast<TWScript*>(scriptApi->GetScript());
		if (!script)
			return retVal; // this should never happen

		// relative paths are taken to be relative to the folder containing the
		// executing script's file
		QDir scriptDir(QFileInfo(script->getFilename()).dir());
		QString path = scriptDir.absoluteFilePath(fileName);

		if (!script->mayReadFile(path, scriptApi->GetTarget())) {
			// Possibly ask user to override the permissions
			if (!askUser)
				return retVal;
			if (QMessageBox::warning(qobject_cast<QWidget*>(scriptApi-
>GetTarget()),
				tr("Permission request"),
				tr("The script \"%1\" is trying to open the file \"%2\" without
```

```
sufficient permissions. Do you want to open the file?")\
                                .arg(script->getTitle()).arg(path),
                        QMessageBox::Yes | QMessageBox::No, QMessageBox::No
                ) != QMessageBox::Yes)
                        return retVal;
                }
        }
        doc = openFile(fileName, pos);
        retVal["result"] = QVariant::fromValue(doc);
        retVal["status"] = (doc != NULL ? TWScriptAPI::SystemAccess_OK :
TWScriptAPI::SystemAccess_Failed);
        return retVal;
}
```

Created with the Personal Edition of HelpNDoc: Free help authoring tool

Return Values for Status

status

```
        var fileResult = TW.readFile("path/to/file.txt");
or
        var status = TW.writeFile("path/to/file.txt","blah blah");
```

For TW.**read**File(the return value res is an object (a map) with three member properties, here we look at the property 'status'

```
        fileResult.status
        fileResult['status']
```

For TW.**write**File(the status value is returned directly as an integer.

```
        status
```

There are three return integer values, in QtScript test for them as numerical integers:

```
        SystemAccess_OK = 0,
        SystemAccess_Failed = 1,
        SystemAccess_PermissionDenied = 2
```

`SystemAccess_PermissionDenied` is returned when writing is disabled.

If it is enabled, though, and writing fails, there is no further distinction (e.g., whether you are trying to write to a folder that doesn't exist, or to a file that is protected by some system's permissions, etc.)

These may be expressed retrieved as a global as in pan_Tw.mod

```
 const twConst = {

                    SystemAccess_OK : 0,
                SystemAccess_Failed : 1,
```

SystemAccess_PermissionDenied : 2

}; // end twConst

Reference: Stefan Löffler http://tug.org/mailman/htdig/texworks/2010q4/003459.html

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

system(QString)

Use with care a write or system call could damage a User's system.

TW.system(cmdLine : string [, waitForResult : bool]) : object;

> This was previously placed as a command requiring no User set permissions directly under TW.app. and was moved to be immediately under TW. in November 2010, and User permissions necessitated. This move requires any existing scripts that used TW.system (...) to be redrafted to meet the new requirements.

There are two arguments, the first one is compulsory, a command line string to be executed by the OS system, and secondly optionally whether to wait for the command to finish and furnish out put from the the command line string that was executed.

User permissions must be set in Tw Edit/Preferences/Scripts

tick -> "Allow Scrips to Run System Commands"

Similar to file access functions, this function elegantly returns an object with result member values.

This result object may be accessed either in dot notation or in the apparent form of an associative array.

E.g. getting a directory listing on windows

```
var retVal = TW.system("cmd /c dir c:\*.* /ad", true);
```

Access results (see)

```
retVal.status
```

or

```
retVal["status"]
```

The form of the command string is entirely operating system dependant. Generally use what ever would return a string from the shell command line.

(We use command line php to retrieve citation key entries from an MySql

database that has been exported from JabRef. See php example)

Windows Users, note Jonathan Kew's remark:—

> "NB: Windows users will need to use "`cmd /c ...`"

The "cmd /c " preface to any other commands is essential.

From dos help "cmd /c Carries out the command specified by string and then terminates"
Termination is needed so that Tw knows that the job is done and especially if waitForResult = true, stops Tw hanging around waiting not knowing what is going on.

Result members:

Member	Purpose
retVal ["status"]	Indicating a successful attempt or otherwise
retVal ["result"]	Any code returned by the command line executed
retVal ["message"]	Indication of what happened on failure
retVal ["output"]	Any String returned by the (even partially) executed command line

"status" (= integer)	"message"	"result" Code	"output" String
SystemAccess_OK = 0 a) waitForResult = true b) waitForResult = false	a) blank b) blank	a) Available b) Empty	a) Available b) Empty
SystemAccess_Failed = 1	c) "Failed to execute system command: [cmd line details] d) "Error executing system command: [cmd line details]	c) Empty d) Available	c) Empty d) Available
SystemAccess_PermissionDenied = 2	"System command execution is disabled (see Preferences)"	Empty	Empty

---

Example:

[edit:]

Getting Cite Keys from and MySql database (exported from JabRef)

Tw Script: 'insertCitation.js'

```
// TeXworksScript
// Title: \cite from JabRef -> MySql
// Description: Make other Scripts
// Author: Paul A. Norman
// Version: 0.3
// Date: 2010-07-16
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+B, Alt+C


// see http://twscript.paulanorman.com/docs/html/files/pan_Twmod.html
// see internally: helper_twPan.mod


eval(TW.app.getGlobal("helper_twPan")); //  Comment if NOT Needed - This includes PhpJs ($P), twConst, msgBox, twPan ($tw), string.toTitleCase()



function makeSpace(howMuch)
{

 var space = new String("");
 for (x = 0; x < howMuch +7; x++) // +7 as we are dealing on Win at least with a proportional font in dropdown
    {
                space += " ";
                }

return space;
}


  var citationType = twPan.citationType;
  var insertCite = twPan.insertCite;
  var citationOrder = twPan.citationOrder;
  var orderBy = twPan.citeOrderBy;


var reBuildDisplayText = [];

var phpScript = __FILE__.substr(0,__FILE__.lastIndexOf("/") +1) + 'getCiteKeys.php';

var chosenOrderBy = twPan.select("Please Choose A Citation Order: ",
```

```
citationOrder) ;  // first and third options probably mostly do the same - for completness

if (chosenOrderBy == undefined){chosenOrderBy = "By Citation Key";}

  for (i = 0; i < citationOrder.length; i++)
            {
                          if (chosenOrderBy == citationOrder[i])
                              {var citeOrderNum = i;
                                          break;
                                          }
                          }

// Non-windows OSes won't need the cmd /c
   var citeKeysAndTitle = twPan.osCmd('cmd /c php ' + phpScript + ' ' + orderBy[citeOrderNum], true).split("\n");


        var maxCiteKeyLength = 0;

        for (descript =0; descript < citeKeysAndTitle.length; descript++) // add padding to move start point for title away from cite_key in drop down immediately below
            {

                          thisDescript = citeKeysAndTitle[descript];

                          test = thisDescript.indexOf(', ');

                          maxCiteKeyLength = (test > maxCiteKeyLength) ? test : maxCiteKeyLength;

                                reBuildDisplayText.push(
                                          [

                            citeKeysAndTitle[descript].substr(0,test +1),
                                citeKeysAndTitle[descript].substr(test +1)

                          ]
                                                                                );
                          }

                          for (descript = 0;  descript < citeKeysAndTitle.length; descript++)
                          {
                           citeKeysAndTitle[descript] = reBuildDisplayText[descript][0]
                                                                + makeSpace(maxCiteKeyLength - reBuildDisplayText[descript][0].length)

                                                          + reBuildDisplayText[descript][1]
                          }
```

```
var chosenCitation = TW.getItem( null, "Insert Citation", "Please Choose A
Citation Key: ", citeKeysAndTitle , 0, true ) ;

  if (chosenCitation != undefined)
  {

          var pageRange = TW.getText( null, "Page Range", "Please enter A
Page Range etc.\n\nE.g.>  123-127\n\nOr Cancel for None - OK to fill out
later", "");
          if (pageRange == undefined)
          {
            pageRange = "";
            }
          else
          {
            pageRange = '[P.~'+ pageRange + ']';
            }

        var citeTypeChosen = TW.getItem( null, "Citation Style", "(In
Preamble, Place: \\usepackage{natbib})\n\n\nChoose A Citation Type: ",
                              citationType , 0, true ) ;

       if (citeTypeChosen != undefined)
       {
          for (find =0; find < citationType.length; find++)
          {
           if (citationType[find] == citeTypeChosen)
           {
                 citeType = insertCite[find];
                 break;
                 }
            }

                                              chosenCitation = chosenCitation.
substr(0, chosenCitation.indexOf(", "));

                  TW.target.insertText('\\'+citeType+pageRange
+'{'+chosenCitation+'}');
                                                          TW.target.
statusTip = 'In Preamble, Need to Place: \\usepackage{natbib} ?';
                  }// /End. IF  citeType != undefined
                      } // /End. IF chosenCitation != undefined
                           null;

   // Thanks to http://www.andy-roberts.net/misc/latex/latextutorial3.html
   // Thanks to http://www.ctan.org/tex-archive/macros/latex/contrib/
natbib/natbib.pdf
```

Php file: 'getCiteKeys.php'

```
<?php

//$orderBy = 'title';
//$orderBy = 'cite_key';

 $orderBy = $argv[1];

// Connecting, selecting database
$link = mysql_connect('localhost', 'root', '')
   or die('Could not connect: ' . mysql_error());


mysql_set_charset('utf8');

// echo 'Connected successfully';
mysql_select_db('jabref') or die('Could not select database');

// Performing SQL query
// $query = 'SELECT cite_key title FROM entries';
$query = 'SELECT CONCAT(cite_key, \',  \', title) FROM entries order by ' . $orderBy ;

$result = mysql_query($query) or die('Query failed: ' . mysql_error());

 $build = "";

while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {

   foreach ($line as $col_value) {
     $build .=  $col_value .  "\n";
   }
}

// Free resultset
mysql_free_result($result);
// Closing connection
mysql_close($link);

 $build = trim($build); // clean off last linefeed else blank option in dropdown results
echo $build;

// Thanks to  http://www.tech-recipes.com/rx/2059/
mysql_use_concat_to_include_text_in_select_results/
?>
```

```
Note: Pulled these TW.system() notes out form here ...

QMap<QString, QVariant> TWScriptAPI::system(const QString& cmdline, bool
waitForResult)
{
```

```
        QMap<QString, QVariant> retVal;
        retVal["status"] = SystemAccess_PermissionDenied;
        retVal["result"] = QVariant();
        retVal["message"] = QVariant();
        retVal["output"] = QVariant();
        // Paranoia
        if (!m_script) {
                retVal["message"] = tr("Internal error");
                return retVal;
        }
        if (m_script->mayExecuteSystemCommand(cmdline, m_target)) {
                TWSystemCmd *process = new TWSystemCmd(this, waitForResult, !
waitForResult);
                if (waitForResult) {
                        process->setProcessChannelMode(QProcess::
MergedChannels);
                        process->start(cmdline);
                        // make sure events (in particular GUI update events
that should
                        // inform the user of the progress) are processed
before we make a
                        // call that possibly blocks for a considerable amount
of time
                        QCoreApplication::processEvents(QEventLoop::
ExcludeUserInputEvents, 100);
                        if (!process->waitForStarted()) {
                                retVal["status"] = SystemAccess_Failed;
                                retVal["message"] = tr("Failed to execute
system command: %1").arg(cmdline);
                                process->deleteLater();
                                return retVal;
                        }
                        QCoreApplication::processEvents(QEventLoop::
ExcludeUserInputEvents, 100);
                        if (!process->waitForFinished()) {
                                retVal["status"] = SystemAccess_Failed;
                                retVal["result"] = process->exitCode();
                                retVal["output"] = process->getResult();
                                retVal["message"] = tr("Error executing system
command: %1").arg(cmdline);
                                process->deleteLater();
                                return retVal;
                        }
                        retVal["status"] = SystemAccess_OK;
                        retVal["result"] = process->exitCode();
                        retVal["output"] = process->getResult();
                        process->deleteLater();
                }
                else {
                        process->closeReadChannel(QProcess::StandardOutput);
                        process->closeReadChannel(QProcess::StandardError);
                        process->start(cmdline);
                        retVal["status"] = SystemAccess_OK;
                }
        }
        else
                retVal["message"] = tr("System command execution is disabled
(see Preferences)");
        return retVal;
}
```

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

system(QString,bool)

See system(QString)

Was under TW.app. moved November 2010

Use with care a write or system call could damage a User's system.

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

Globals

Globals

Strings and other QtScript objects that Qt Scripting can cast directly can be stored in objects.

All global objects do not automatically persist when the User closes Tw.

If you have difficulties, the safest bet are simple things like numerical values and strings - and don't forget JSON structures.

Once stored (with a name created by the script writer), the global can be tested for (by name - does it exist already? - hasGlobal) and be re-read back into QtScript.

Globals are stored for as long as the Tw editor is still running, and should be retrieved and formatted (possibly json) and saved to disk if required for later use.

There are two types of Globals.

1. Globals that are available from and to any script while the Tw editor is still open, see Application Wide Globals

2. And globals that are assigned to a particular script, they are available again during the current Tw session, if the script that created them is used again. Other scripts do not have access to globals stored by another script, see TW.script.hasGlobal(QString) TW.script.setGlobal(QString, QVariant) and TW.script.getGlobal(QString)

hasGlobal(QString)

```
TW.app.hasGlobal("my_Globaal_01");
TW.script.hasGlobal("colour_Name");
```

Returns a boolean (true or false) showing whether the global has already been created and set.

setGlobal(QString)

```
TW.app.setGlobal("font_name", yourObject);
TW.script.setGlobal("anyThing_you_like", anotherObject);
```

The object should be one that Qt can cast (convert) from QtScript see table here: Default Conversion from Qt Script to C++

If you have difficulties, the safest bet are simple things like numerical values and strings.

getGlobal(QString)

```
var myGlobalVariable = TW.app.getGlobal("font_name");
var myScriptVariable = TW.script.getGlobal("anyThing_you_like");
```

Returns the global named, to script, and stores it in the variable name you supply. Use hasGlobal first if unsure of its existence.

Unsetting Global variables

TW.app.**unsetGlobal**("foo")

and

TW.script.**unsetGlobal**("foo")

References:
> How do I get rid of a global variable I've set

> http://tug.org/pipermail/texworks/2012q2/005453.html
>
> Oops, I think that was overlooked when the "globals" framework was implemented.
> I added the two functions TW.app.unsetGlobal("foo") and TW.script.unsetGlobal("foo") in r994. They should do just what you want.
>
> Cheers,
> Stefan

http://www.google.com/codesearch/p?hl=en#TCAOwTxChU8/trunk/src/TWApp.h&q=getGlobal%20package:http://texworks%5C.googlecode%5C.com&sa=N&cd=1&ct=rc&l=147

http://www.google.com/codesearch/p?hl=en#TCAOwTxChU8/trunk/src/TWApp.cpp&q=setGlobal%20package:http://texworks%5C.googlecode%5C.com&l=1177

```
void TWApp::setGlobal(const QString& key, const QVariant& val)
```

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

app

The TW.app object

The TW.app object handles the hosting application (TeXworks itself).

Conceptually a bit like window and window.external in a browser Api.

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

Application Wide Globals

See Globals

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

getGlobal(QString)

See Globals

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

hasGlobal(QString)

See Globals

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

setGlobal(QString,QVariant)

See Globals

Created with the Personal Edition of HelpNDoc: Free help authoring environment

unsetGlobal(Qstring)

TW.app.unsetGlobal("whatever")

See
Globals

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

clipboard

TW.app.clipboard : QString

This is a string property, it may be read or assigned (a string).

```
TW.app.clipboard ="My Text";

TW.information(null, "Clipboard - What\'s On It?", TW.app.clipboard);
```

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

clipboard

TW.app.clipboard

Read/Write plain text from/to clipboard

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

setClipboardText(QString)

setClipboardText(QString)

use clipboard property instead

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

setClipboardText(QString,QClipboard::Mode)

See QClipboard::Mode for mode details

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

clipboardText()

Use clipboard property instead

Created with the Personal Edition of HelpNDoc: Full featured Help generator

QClipboard::Mode

enum QClipboard::Mode

This enum type is used to control which part of the system clipboard is used by QClipboard::mimeData(), QClipboard::setMimeData() and related functions.

Constant	Value	Description
QClipboard::Clipboard	0	indicates that data should be stored and retrieved from the global clipboard.
QClipboard::Selection	1	indicates that data should be stored and retrieved from the global mouse selection. Support for Selection is provided only on systems with a global mouse selection (e.g. X11).
QClipboard::FindBuffer	2	indicates that data should be stored and retrieved from the Find buffer. This mode is used for holding search strings on Mac OS X

From: http://doc.trolltech.com/latest/qclipboard.html#Mode-enum

Notes:—

const QMimeData * QClipboard::mimeData (Mode *mode* = Clipboard) const

Returns a reference to a QMimeData representation of the current clipboard data.

The *mode* argument is used to control which part of the system clipboard is used.
If *mode* is QClipboard::Clipboard, the data is retrieved from the global clipboard.
If *mode* is QClipboard::Selection, the data is retrieved from the global mouse selection.

If *mode* is QClipboard::FindBuffer, the data is retrieved from the search string buffer.

void QClipboard::setMimeData (QMimeData * *src*, Mode *mode* = Clipboard)

Sets the clipboard data to *src*. Ownership of the data is transferred to the clipboard. If you want to remove the data either call clear() or call setMimeData() again with new data.

The *mode* argument is used to control which part of the system clipboard is used.
If *mode* is QClipboard::Clipboard, the data is stored in the global clipboard.
If *mode* is QClipboard::Selection, the data is stored in the global mouse selection.
If *mode* is QClipboard::FindBuffer, the data is stored in the search string buffer

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

Files

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

openFileFromScript

See openFileFromScript(QString,TW,int,bool)

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

getOpenFileName()

TW.app.getOpenFileName();

This function will cause the "open file" dialogue box to appear and return the file name chosen or an empty string if no file is chosen.

A full path name is returned.

There is an overload function that will take a string as the argument, which is a filter. See getOpenFileName(QString)

Note: The function does not itself open anything, only returns the path and name that the user selects.

See also getOpenFileNames() and getOpenFileNames(QString)

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

getOpenFileName(QString)

TW.app.getOpenFileName(QString);

This is an overload function to getOpenFileName()

It will cause the "open file" dialogue box to appear and return the file name chosen or an empty string if no file is chosen.

A full path name is returned.

This function requires a string as the argument, which is a filter.

The filter can be as simple as,

"*.jpg"

or include a message and in round brackets a file extension,

"Jpeg (*.jpg)"

include a maeesage adfn a space separated list of file extensions enclosed in round brackets,

"Image Files (*.jpg *.jpeg *.png *.pdf)"

TW.information(null,"Choose Image",TW.app.getOpenFileName("Image Files (*.jpg *.jpeg *.png *.pdf)"));

Note: The function dos not itself open anything, only returns the path and name that the user selects.

See also getOpenFileNames() and getOpenFileNames(QString)

For source code examples of filters see Filters in http://www.google.com/codesearch/p?hl=en#TCAOwTxChU8/trunk/src/TWUtils.cpp

For the C++ implementation of code see http://www.google.com/codesearch/p?hl=en#TCAOwTxChU8/trunk/src/TWApp.cpp&q=getOpenFileName%20package:http://texworks%5C.googlecode%5C.com

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

getOpenFileNames()

TW.app.getOpenFileNames()

Returns an array of chosen filenames, if no filename is chosen, a zero length array is returned.

Note: The function dos not itself open anything, only returns the path(s) and name (s) that the user selects.

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

getOpenFileNames(QString)

TW.app.getOpenFileNames(QString)

As for getOpenFileNames() and returns an array of chosen filenames, if no filename is chosen, a zero length array is returned.

This overload function requires a string representing a filter see getOpenFileName (QString) for details of filename filters.

Note: The function dos not itself open anything, only returns the path(s) and name (s) that the user selects.

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

getSaveFileName(QString)

TW.app.getSaveFileName(QString)

As for getOpenFileName(QString) but shows the File Save dialogue instead, the required string is a proposed file name (`const QString& defaultName)`

TW.app.getSaveFileName("MyList.txt")

The user will be prompted with what to do if the filename already exists.

If no filename is chosen by the User, an empty string is returned.

Note: The function does not itself save anything, only returns the path and name that the user selects.

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

newFile()

TW.app.newFile();

Creates a new file and sets the focus in the editor.

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

open()

TW.app.open();

Presents the open file dialogue.
From that point scripting looses 'contact' with the function.

Does not receive a file filter nor a suggested file name, but will not allow the dialogue "Ok" button to be enabled unless the User chooses an actual file.

Upon success, opens the chosen file in a new editor window, (returns no value — for neither success nor failure).
That editor window does not become the new TW.target.

Created with the Personal Edition of HelpNDoc: Easily create Help documents

openFile(QString,int)

Removed late 2010.

See TW.openFileFromScript(QString,int,bool)

Created with the Personal Edition of HelpNDoc: Full featured Help generator

recentFileActionsChanged()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

updateRecentFileActions()

As stated.

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

Informational

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

aboutQt()

```
TW.app.aboutQt();
```

Shows a dialogue box with information about the version of Qt being used.

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

about()

```
TW.app.about();
```

Shows the application's about box, information and links.

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

applicationName

```
TW.app.applicationName
```

A property reading, as it says the application name.

Example:

```
TW.information(null, "Name of Application", TW.app.applicationName);
```

Shows a message box with the text: TeXworks

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

applicationVersion

Created with the Personal Edition of HelpNDoc: Free Web Help generator

getVersion()

TW.app.getVersion()

Returns the application sub version number e.g. 768

The return
> value is in the form of 0xMMNNPP, with MM=major version, NN=minor
> version, PP=bugfix, so at the moment this would return . This is
> not human-readable (as applicationVersion should be), but much more
> useful in scripts for testing the current version.

Example:
TW.information(null, "Version of Application",TW.app.getVersion());

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

goToHomePage()

TW.app.goToHomePage();

Initially runs a system command to show page http://texworks.org

```
Registrant Name:karl berry
Registrant Organization:karl berry
```

Then redirects to:–

http://tug.org/texworks

A sub-domain off freefriends.org operated by Karl Berry.

Karl is the President of the Tex Users Group, and listed as one of the TeXworks developers and major sponsor.

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

objectName

TW.app.objectName();

Reads property.

Returns the application coding object name: TeXworks

Example:
TW.information(null, "Object Name",TW.app.objectName);

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

openHelpFile(QString)

TW.app.openHelpFile(QString);

This works by being supplied a local directory path (no file name) in which there must be an index.html

If there is no index.html in that local directory, or the attempt otherwise fails, then the following message is displayed:

```
"Unable to find help file."
```

As of rev 768 at least, the path can be any valid path on the PC.

Example:

If there is an index.html file in the named directory both of these will work:-

```
TW.app.openHelpFile("D:/LaTeXPortable/web_mirror/where_my_helpfile_is/");
```

Or

```
TW.app.openHelpFile("D:/LaTeXPortable/web_mirror/where_my_helpfile_is");
```

But even though an index.html is being sought, this will not work:-

```
TW.app.openHelpFile("D:/LaTeXPortable/web_mirror/where_my_helpfile_is/index.html");
```

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

organizationDomain

TW.app.organizationDomain

Read property returns String

TW.information(null, "Organization Domain",TW.app.organizationDomain);

Shows: tug.org

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

organizationName

TW.app.organizationName

Read property returns String

TW.information(null, "Organization Domain",TW.app.organizationName);

Shows: TUG

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

writeToMailingList()

TW.app.writeToMailingList();

Launches registered email client to compose an email to TeXworks developers on the mailing list.

Does not itself send anything, just starts composing a new email for the User to complete.

Includes the following kind of information by default –

```
To: texworks@tug.org <texworks@tug.org>
Subject: Message from TeXworks user


----- configuration info -----
TeXworks version : 0.3r726 (MinGWcross)
Install location : D:/latexportable/LaTexUtils/TeXWorks/texworks/TeXworks.exe
Library path : D:/latexportable/LaTexUtils/TeXWorks/texworks/config\
pdfTeX location : D:/LaTeXPortable/LatexUtils/MiTeX 2.8.3541 Portabe/miktex/
bin/pdftex.exe
Operating system : Windows XP Professional Service Pack 3
Qt4 version : 4.7.1 (build) / 4.7.1 (runtime)
------------------------------
```

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

Settings

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

applyTranslation(QString)

TW.app.applyTranslation(QString locale)

Supply a two character string for the language locale

Changes all the Menus and settings to the designated locale if available in TeXworks.

http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

Look for the column "639-1"

In TeXworks, "en" - English is available by default.

And as of July 2011 - these languages are already available.

Code	Language	Other Information
af	Afrikaans	

ar	Arabic	
ca	Catalan	
cs	Czech	
de	German	
en	**English**	**Euro style (UK) (Tw default)**
es	Spanish	Castilian
fa	Persian	Farsi or "Pârsi" (local name)
fo	Faroese *	
fr	French	
it	Italian	
ja	Japanese	
ko	Korean	
nl	Dutch	Nederlands, Vlaams
pl	Polish	
pt_BR	Portuguese	Brazilian
ru	Russian	
sl	Slovene	
tr	Turkish	
zh_CN	Chinese (Simplified)	Zhongwén

* "Support for Faroese is currently limited to Tw-specific strings. I.e., the installer doesn't include Faroese (as Inno setup doesn't provide appropriate language files), and Qt doesn't support it, either (so Qt stuff like the default "OK", "Cancel", "Save", ... buttons etc. won't be translated)." - Stefan Löffler

For TeXworks specific information on translating the interface see: http://code.google.com/p/texworks/wiki/Translating

---

Examples:

TW.app.applyTranslation("fr");

Changes all the Menus and settings to French.

TW.app.applyTranslation("en");

Changes all the Menus and settings to English.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

cursorFlashTime

TW.app.cursorFlashTime;

From: http://doc.qt.nokia.com/latest/qapplication.html#cursorFlashTime-prop

This property holds the text cursor's flash (blink) time in milliseconds.

The flash time is the time required to display, invert and restore the caret display. Usually the text cursor is displayed for half the cursor flash time, then hidden for the same amount of time, but this may vary.

The default value on X11 is 1000 milliseconds. On Windows, the **Control Panel** value is used and setting this property sets the cursor flash time for all applications.

Surprisingly on Windows Xp Pro this will not only read the setting but can also assign it
This happens with *some* Qt properties in scripting.

```
TW.app.cursorFlashTime = 100;
```

(OS wide – affecting ALL OTHER applications).

Qt documentation does not suggest this will be the case and says to use **setCursorFlashTime** (int) instead, which actually is not available to scripting.

As setting this is not yet officially supported by Qt, set it under advisement if at all, otherwise only read it.

Example:

```
TW.information(null, "cursor Flash Time",TW.app.cursorFlashTime);
```

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

doubleClickInterval

TW.app.doubleClickInterval

From: http://doc.qt.nokia.com/latest/qapplication.html#doubleClickInterval-prop

This property holds the time limit in milliseconds that distinguishes a double click from two consecutive mouse clicks.

The default value on X11 is 400 milliseconds. On Windows and Mac OS, the operating system's value is used. However, on Windows and Symbian OS, calling this function sets the double click interval for all applications.

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

keyboardInputInterval

TW.app.keyboardInputInterval

Property that in QtScript appears to both set and read the keyboardInputInterval in milli seconds.

This property holds the time limit in milliseconds that distinguishes a key press from two consecutive key presses.

The default value on X11 is 400 milliseconds. On Windows and Mac OS, the operating system's value is used.

See http://doc.qt.nokia.com/latest/qapplication.html#keyboardInputInterval-prop

Under Xp setting is also 400 `twPan.alert(TW.app.keyboardInputInterval);`

```
TW.app.keyboardInputInterval = 300;
twPan.alert(TW.app.keyboardInputInterval);
TW.app.keyboardInputInterval = 400;
```

... Appears to work.

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

layoutDirection

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

quitOnLastWisdowClosed - don't use!

TW.app.quitOnLastWindowClosed

In Script this property reads and is writable. It does not quite do what some might expect.

```
TW.app.quitOnLastWindowClosed = false;
twPan.alert(TW.app.quitOnLastWindowClosed);
```

Works, and when the last window is closed Tw remains as a process with no Application interface.

Basically **don't use** - User would need to find the process and ill it if this were called as above, otherwise they would not be able to restart TeXworks until they rebooted.

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

startDragDistance

Created with the Personal Edition of HelpNDoc: Free Web Help generator

startDragTime

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

wheelScrollLines

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Spelling

See Spell Checking

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

reloadSpellchecker()

See Spell Checking

Created with the Personal Edition of HelpNDoc: Full featured Help generator

Windows

Created with the Personal Edition of HelpNDoc: Easily create Help documents

getOpenWindows()

TW.app.getOpenWindows();

Returns an array of TW.target objects currently open.
That is an array of current Tw edit objects.

Example:-

```
  //  eval(TW.app.getGlobal("helper_twPan"));  // using longhand so that libraries can be independently loaded, PAN libraries must be set up

  //  eval(TW.app.getGlobal("helper_PhpJs"));  // rem out if PAN libraries not set up


  var windows =  TW.app.getOpenWindows();
  var titleList = [];

    for (editor in windows)
        {

          var targetDocument = windows[editor];
          titleList.push(targetDocument.windowTitle);
          }

  chosenWindow = TW.getItem(null, "Please Choose Document", "Please Choose Document", titleList);

  TW.information(null, "Choosen Document", chosenWindow);


                // requires helper_PhpJs
 //  arrayPosition = PhpJs.array_search(chosenWindow, titleList);
```

```
                                    // needle, Array to search
// If   helper_PhpJs not in use do this longahand

  for (count in titleList)
      {

     var arrayPosition = -1;
       if ( titleList[count] == chosenWindow)
          {
          arrayPosition = count;
          break;
         }
       }

//  TW.information(null, "Array Position", arrayPosition); // requires helper_PhpJs

    TW.information(null, "Choosen Document File Name", windows[arrayPosition].fileName);
```

Created with the Personal Edition of HelpNDoc: Free Web Help generator

activatedWindow(QWidget*)

TW.app.activatedWindow(QWidget*)

If the User has pulled out various panes like the Tag WIndow, this will hide them all and show the edit window.

May need to us with findWindow

Reference Code:

http://www.google.com/codesearch/p?hl=en#TCAOwTxChU8/trunk/src/TWApp.cpp&q=hideFloatersExcept%20package:http://texworks%5C.googlecode%5C.com&l=985

```
void TWApp::activatedWindow(QWidget* theWindow)
{
        emit hideFloatersExcept(theWindow);
}
```

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

closeAllWindows()

As it says.

Exits whole Tw application.

Created with the Personal Edition of HelpNDoc: Full featured Help generator

stackWindows()

As stated.

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

tileWindows()

As stated.

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

windowListChanged()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

updateWindowMenus()

```
// called by windows when they open/close/change name
void updateWindowMenus()
```

Possibly of no use.

Created with the Personal Edition of HelpNDoc: Free help authoring tool

maybeQuit()

```
TW.app.maybeQuit();
```

Checks first to see if any document needs saving (dirty). If so prompts the User for Save instruction and provides the option then to cancel the close/quit process.

Created with the Personal Edition of HelpNDoc: Full featured Help generator

quit()

```
TW.app.quit();
```

May close Tw forcibly, potentially causing data loss.

In all normal cases, use TW.app.maybeQuit() instead. [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

showScriptsFolder()

```
TW.app.showScriptsFolder()
```

Helpful procedure, opens the Users Script folder in default file explorer. One folder up is the home of other configuration sub folders as well. See also .

Created with the Personal Edition of HelpNDoc: Free help authoring tool

syncPdf(QString,int,bool)

TW.app.syncPdf(**fullFileName** : QString, **lineNumber**: int, **showPreview**: bool)

This procedure appears to require that the document is open already.

As you may have more than one document open, the first parameter is the **full**

file path of the document that you wish to move to **line number** (as shown in the source editor) in the previewer, then the last parameter says whether to bring the **previewer** forward or not **true/false**.

If the correct document is already the active document then `TW.target.fileName` could be used for **full file name**

Possible uses could include a choice made by the User in a drop down box which maps to an array of already gathered and assigned line numbers.

From Tw source

```
syncPdf(const QString& sourceFile, int lineNo, bool activatePreview);
```

http://www.google.com/codesearch#TCAOwTxChU8/trunk/src/TWApp.h&q=syncPdf%20package:http://texworks%5C.googlecode%5C.com&l=215

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

updateScriptsList()

TW.app.updateScriptsList()

Will rescan the Scripts folder and sub-directories and reset the Scripts Menu.

Must be used (even just from Script menu) in current Tw session, if you want new scripts to show before next restart which will fond them anyway.

Used in this script which is a helper for making new scripts

```
// TeXworksScript
// Title:  Make Script
// Description: Creates a New Script
// Author:  Paul A. Norman
// Version: 0.31
// Date:  201-12-21
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+N, Alt+S

    // Fist Make a New Blank Document manually

    // updated 2011 07 23 around line 230 to include TW.app.updateScriptsList();

     // TW.app.newFile();   // doesn't always get the focus, and last active open
document is often the target

 var  varNames = ["scriptName", "scriptDescription", "scriptAuthor", "scriptVersion",
"scriptDate",
                "scriptType", "scriptContext", "scriptShortcut", "scriptHook"] ;
```

```
  var scriptParts = []; // New array to hold our script parts

// First create objects

  var availableScriptLanguages = ["QtScript","Python","Lua"];  // <-- Add any new
Script languages and specific needs here -->

  var  scriptSpecific = {

                      "QtScript" : {
                                  "headerPrefix" : "",
                                    "linePrefix" : "// ",
                                  "headerSuffix" : "",
                                      "fileExt" : ".js"
                                 },

                       "Python" : {
                                  "headerPrefix" : "",
                                    "linePrefix" : "# ",
                                  "headerSuffix" : "",
                                      "fileExt" : ".py"
                                 },

                          "Lua" : {
                                  "headerPrefix" : "--[[",
                                    "linePrefix" : "",
                                  "headerSuffix" : "]]",
                                      "fileExt" : ".lua"
                                 }
                       }


    for (upto in varNames)
        {

         eval(
              "var " + varNames[upto]
              + " = {"
              + "  'partName': " +  varNames[upto] // for error debuging
              + ", 'useThis' : 'dont use'" // no script part is used unless it is in the User's
decision flow path.
              + ", 'preFixText' :""
              + "    }"
             );

         eval ("scriptParts.push("+varNames[upto]+")"); // Add to array holding our
script parts

         }


     with (TW) with (TW.app) with (TW.target)
     {

      scriptName.useThis =    getText(null, 'Script Name (Title)', 'Script Name (Menu
Title)');
        scriptName.preFixText = "Title:  ";

      scriptDescription.useThis = getText(null, 'Script Description', 'Script Description
```

```
(Shows in Script Manager)');
       scriptDescription.preFixText = "Description: ";

     var fileResult = readFile('authorName.txt'); // retrieve any previously used Author
Name

         if (fileResult.status == 0)
              { var namePrompt = fileResult.result}
            else
              { var namePrompt = 'Your Name'}

     scriptAuthor.useThis =    getText(null, 'Script Author', 'Script Author (Shows in
Script Manager)', namePrompt);
       scriptAuthor.preFixText = "Author:   ";

        if ((namePrompt != scriptAuthor.useThis) & (scriptAuthor.useThis !=
undefined)) // Save Author Name
          {
                writeFile('authorName.txt', scriptAuthor.useThis);
           }

     scriptVersion.useThis =    getText(null, 'Script Version', 'Script Version (Shows in
Script Manager)', '0.1');
       scriptVersion.preFixText = "Version: ";

     // Altered from: http://www.tizag.com/javascriptT/javascriptdate.php
         var currentTime = new Date();
         var month = currentTime.getMonth() + 1;
                var day = currentTime.getDate() + ''; // force cast for length test
                var year = currentTime.getFullYear();

                if (day.length < 2){day = "0" + day;}
                    month = month + '';  // force cast for length test
                if (month.length < 2){month = "0" + month;}

     scriptDateBuild = year + '-' + month + '-' + day;

     scriptDate.useThis =    getText(null, 'Script Date', 'Script Date (Shows in Script
Manager)', scriptDateBuild);
       scriptDate.preFixText = "Date: ";

     scriptType.useThis = getItem(null, 'Script Type', 'Choose Script Type - \n\n 1.
User initiated - \"standalone\",\n    or\n 2.  Automatic (Event/Signal Driven) - \"hook
\"', ['standalone','hook']);
       scriptType.preFixText = "Script-Type: ";


    switch(scriptType.useThis)
    {
        case 'hook':
      scriptHook.useThis = getItem(null, 'Script Hook Type', 'Script Hook Activated
by ...', ['NewFile','NewFromTemplate','LoadFile','AfterTypeset', 'TeXworksLaunched']);
         scriptHook.preFixText = "Hook: ";

            break;

        case 'standalone':
                default:

         nameParts = scriptName.useThis.split(" ");
```

```
          var shortCutKeysPrompt = '';

          if (nameParts.length > 0)
            {
              for (var xJ in nameParts)
                 {
                  shortCutKeysPrompt += "Alt+" + nameParts[xJ][0].toUpperCase();

                    if (xJ < nameParts.length -1)
                       { shortCutKeysPrompt += ", ";
                       }
                 }
              }
              else
              {
               shortCutKeysPrompt =  'Alt+X, Alt+X'} // for completion


    scriptShortcut.useThis  = getText(null, 'Script Shortcut', 'Script Shortcut Keys
(Activates Script) Change the Xs', shortCutKeysPrompt);
      scriptShortcut.preFixText = "Shortcut: ";

           scriptContext.useThis = getItem(null, 'Script Context', 'Script Context (Use in
Editor or Pdf Viewer)', ['TeXDocument','PDFDocument']);
            scriptContext.preFixText = "Context: ";

                  break;

     }  // End. switch  scriptType


      var scriptLanguage = getItem(null, "Please Choose Script Language", "Script
Language: \n", availableScriptLanguages);

            if (scriptLanguage == undefined){scriptLanguage = "QtScript"; } // got to
be something


          insertText(scriptSpecific[scriptLanguage].headerPrefix + scriptSpecific
[scriptLanguage].linePrefix + 'TeXworksScript');
                          // new line added later

          for (upto in scriptParts)
          {

            if (scriptParts[upto].useThis == "dont use") // set at object creation,
presence here means script part not chosen in User's decision flow path (e.g. see
switch(scriptType.useThis) above)
                {continue;}

                          //add new line here so we can add final new line after
scriptSpecific[scriptLanguage].headerSuffix below
              var thisText = "\n" +scriptSpecific[scriptLanguage].linePrefix
                            + scriptParts[upto].preFixText; // build Header skeleton
whether content added below or not

           if (scriptParts[upto].useThis != undefined)  // If undefined, User cancelled
input box
              {
                  thisText += scriptParts[upto].useThis;  // add content User chose/
```

```
wrote
                }
                    insertText(thisText) ;
            }

            insertText(scriptSpecific[scriptLanguage].headerSuffix + "\n");


          if (scriptLanguage == 'QtScript')
          {

 var use_twPan = false;

     if ((scriptAuthor.useThis == "Paul Norman")||( TW.question(null, "Include panTw
Module?", "Include panTw Module?", 0x00004000 | 0x00010000, 0x00004000) ==
0x00004000       ))
          {

           insertText('\n\n\n  eval(TW.app.getGlobal("helper_twPan")); //  Comment if
NOT Needed - This includes PhpJs ($P), twConst, msgBox, twPan ($tw), string.
toTitleCase() \n');
              }
            else
             {
     if ( TW.question(null, "Include PhpJs Module?", "Include PhpJs Module?",
0x00004000 | 0x00010000, 0x00004000) == 0x00004000      )
              {
           insertText('\n\n\n  eval(TW.app.getGlobal("helper_PhpJs")); //Comment if NOT
Needed, This only includes PhpJs ($P) \n');
              }
                  }

           } // End. if (scriptLanguage == 'QtScript')

           setSyntaxColoringMode(scriptLanguage); /* no error produced if Syntax Type
not present in syntax-patterns.txt  -
                                       could rem out if not using a syntax colouration
model for [QtScript] [Python] [Lua]
                                       see /config/configuration  syntax-patterns.txt and
                                       http://twscript.paulanorman.com/docs/html/files/
syntaxpatternstxt.html */


             statusTip = "Proposed File Name is on Clipboard";   // set first so that it
shows in time
             yield();

                insertText('\n\n');


          var fileSaveName = scriptName.useThis.replace(/ /g,"") + scriptSpecific
[scriptLanguage].fileExt;

          fileSaveName = fileSaveName.substr(0,1).toLowerCase() + fileSaveName.substr
(1);

          var holdClipboard = clipboard;
              clipboard =  fileSaveName;

            var doSave =  question(null,"Do You Wish to Save File Now?","File Name is on
```

```
Clipboard \n\nChange File Extension to All Files (*)\n\n Do You Wish to Save File
Now?", 0x00004000 | 0x00010000, 0x00004000); // Yes | No, prefer to use msgBox in
helper_twPan.mod

        if (doSave == 0x00004000) // if Yes
         {
          var fileResult  = saveAs(); // catch return value

          if( fileResult == true)
            {
             TW.app.updateScriptsList();
            }

                // dispose of return value - it was appearing in a message box

            fileResult  = null;
          }



            } // End. with (TW) with (TW.app) with (TW.target)

          clipboard = holdClipboard;     // restore clipboard

          null; // stop extra Qt framework generated message box appearing
```

Created with the Personal Edition of HelpNDoc: Free help authoring environment

Dont Use

Created with the Personal Edition of HelpNDoc: Easily create Help documents

Changed ...

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

scriptListChanged()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

focusChanged(QWidget*,QWidget*)

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

engineListChanged()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring tool

fontDatabaseChanged()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

set This ()

Created with the Personal Edition of HelpNDoc: Full featured Help generator

setStyleSheet(QString)

Possibly not available to Scripting see http://doc.qt.nokia.com/latest/qapplication.html#styleSheet-prop

Created with the Personal Edition of HelpNDoc: Easily create Help documents

saveStateRequest(QSessionManager&)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

setAutoSipEnabled(bool)

[X] - Dont Use - Not Available/Not Useful (4.5+ later; only on WindowsCE and Symbian, which we don't support) [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring environment

highlightLineOptionChanged()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

launchAction()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring environment

globalStrut

Created with the Personal Edition of HelpNDoc: Full featured Help generator

styleSheet

See http://doc.qt.nokia.com/latest/qapplication.html#styleSheet-prop

> This property holds the application style sheet.
>
> By default, this property returns an empty string unless the user specifies the `-stylesheet` option on the command line when running the application.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

unixSignal(int)

[?] [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring environment

updatedTranslators()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

aboutToQuit()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

autoSipEnabled (property)

[X] - Dont Use - Not Available/Not Useful

(4.5+ later; only on WindowsCE and Symbian, which we don't support [remarks - Stefan Löffler])

Created with the Personal Edition of HelpNDoc: Free help authoring tool

autoSipEnabled()

[X] - Dont Use - Not Available/Not Useful

(4.5+ later; only on WindowsCE and Symbian, which we don't support [remarks - Stefan Löffler])

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

commitDataRequest(QSessionManager&)

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring tool

deleteLater()

[X] - Dont Use - Not Available/Not Useful destroys the app object (i.e., crashes Tw) [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

destroyed()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring tool

destroyed(QObject*)

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

hideFloatersExcept(QWidget*)

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

lastWindowClosed()

Qt signal - (event)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring environment

quitOnLastWindowClosed

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

windowIcon

TW.app.windowIcon

Not completely cast no functions or properties directly available to Script.

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

target

The TW.target object

The TW.target object handles the current Editing window.

Conceptually mostly a bit like window.document in a browser Api.

For an array of open documents (target objects) see: TW.app.getOpenWindows()

TeXDocument(name = "TeXDocument")

Created with the Personal Edition of HelpNDoc: Free help authoring environment

TW.target template

TwScript: TW.target.();

See Qt 4.4 below.

Example:

TW.target.();

.cpp

Provided in Qt framework.

Qt 4.4 ()

.

.

reference:

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

Informational

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

accessibleDescription

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

accessibleName

Created with the Personal Edition of HelpNDoc: Full featured Help generator

consoleOutput

read-Only - area where for example the (error messages) results of typesetting appear.

Created with the Personal Edition of HelpNDoc: Full featured Help generator

setWindowTitle(QString)

[X] - Dont Use - Not Available/Not Useful use windowTitle property instead (may be removed at some point) [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

statusTip

Text that appears in the pane/window's bottom status bar, can be read or written.

This message may only show when the cursor is over the document.

Examples

```
TW.target.statusTip = "Document processed";

TW.information(null,  TW.target.statusTip, TW.target.statusTip);
```

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

toolTip (file menu?)

As appears in file menu?

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

windowTitle

Mouseover text, like a Browser html title property on an html tag.

Appears anywhere that there is not a specific tool tip set, over status bar, button area etc.

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Editor Commands Properties

Most of these are self explanatory.

All start:

```
TW.target.
```

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

documentMode

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

enabled

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

focus

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

focusPolicy

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

setAutoIndentMode(QString)

setAutoIndentMode(QString)

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

setLineNumbers(bool)

setLineNumbers(bool)

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

setSmartQuotesMode(QString)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

setSpellcheckLanguage(QString)

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

setSyntaxColoring(int)

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

setSyntaxColoringMode(QString)

```
// TeXworksScript
// Title: Syntax for Scripting
// Description: Colourises  .js .qs .mod files
```

```
// Author: Paul A Norman
// Version: 0.3
// Date: 2010-12-20
// Script-Type: hook
// Hook: LoadFile


 var  fileName = TW.target.fileName;

 var   extPos = fileName.lastIndexOf('.') +1;

if (extPos > 0)
 {
  var ext = fileName.substr(extPos);

  // updated 2011 07 24

  switch(ext)
        {

        case 'html':
        case  'htm':
        case  'xml':
         TW.target.setSyntaxColoringMode('HTML');   // make sure you have an HTML section!
        break;

        case   'qs':
        case   'js':
        case 'mod':
        case  'php':
         TW.target.setSyntaxColoringMode('QtScript');
        break;


        default:
         TW.target.setSyntaxColoringMode('LaTeX');
        break;
       }

  }  //End. if (extPos > 0)
```

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

setWrapLines(bool)

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

showSelection()

As stated, will also scroll editor to the position of the cursor if nothing is selected, which is useful if you have inserted text from Script.

TW.target.showSelection();

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

syncClick(int)

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

syncFromSource(QString,int,bool)

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

tagListUpdated()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

toggleConsoleVisibility()

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

typeset()

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

untitled

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

windowModified

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

Editor Dialogues

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

doFontDialog()

Created with the Personal Edition of HelpNDoc: Free Web Help generator

doHardWrapDialog()

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

doLineDialog()

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

doReplaceAgain()

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

text

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

balanceDelimiters()

TwScript: TW.target.balanceDelimiters();

In the editor, from whee the cursor is, any text between code "delimiters" will be selcted,
and the delimeters will be as well,
(highlighted) .

Same as the user taping Crtl B, or using Menu/ Edit /Balance delimiters

Works on:—

( )
[ ]
{ }

Presntly not on:—

" "
' '
< >
` '

Example:

```
TW.target.balanceDelimiters();

var selText = TW.target.selection;
    selText = selText.substr(1, selText.length -2);

// would return just the contents of the brackets.
```

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

clear()

TwScript: TW.target.clear();

Any text selected (highlighted) in the editor will be cleared.

Example:

```
TW.target.clear();
```

TeXDocument.cpp

```
void TeXDocument::clear()
{
        textEdit->textCursor().removeSelectedText();
}
```

Qt 4.4 void QTextCursor::removeSelectedText ()

If there is a selection, its content is deleted; otherwise does nothing.

See also hasSelection().

reference: http://doc.trolltech.com/4.4/qtextcursor.html#removeSelectedText

---

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

cursor

QVariant(QCursor) -- not useful?

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

doComment()

TwScript: TW.target.doComment()

Inserts % at the begiing of current/selected line(s)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

doIndent()

TwScript: TW.target.doIndent()

Indents current/selected line(s) of text

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

doUncomment()

TwScript: TW.target.doUncomment()

Removes % at the beging of current/selected line(s)

Created with the Personal Edition of HelpNDoc: Full featured Help generator

doUnindent()

TwScript: TW.target.doUnindent()

Un-indents current/selected lines of text

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

font

TwScript: TW.target.font

```
TW.information(null, "Font Properties", TW.target.font);
```

Returns font information in Qt format.

Possibly read only - experiment - check out the Qt font documentation some functions properties may be available:—

http://doc.qt.nokia.com/4.7/qfont.html

Created with the Personal Edition of HelpNDoc: Full featured Help generator

insertText(QString)

TwScript: TW.target.insertText(QString)

Any text selected in the editor will be replaced by supplied string.
If no text is sleceted the string is inserted at the cursor position.

Example:

```
TW.target.insertText("hello");
```

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

layoutDirection

layoutDirection

possible not useful

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

locale

QVariant(QLocale)

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

lower()

unknown as yet.

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

modified

TwScript: TW.target.modified

A property that lets you set or read whether the document has been modified.

In script you can make the document "dirty" forcing the user to be prompted for save option beofre the document is closed.

```
TW.target.modified = true;

TW.information(null, "Document Modified?", TW.target.modified);
```

Created with the Personal Edition of HelpNDoc: Easily create Help documents

selectAll()

TwScript: TW.target.selectAll();

as stated

To replace all text, perhaps something like,

```
TW.target.selectAll();

TW.target.insertText("hello");
```

Created with the Personal Edition of HelpNDoc: Free help authoring environment

selectRange(int)

TwScript: TW.target.selectRange(int)

By it self will merely position the cursor at the int position from the beginning of the document.

To actually select available text see selectRange(int,int)

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

selectRange(int,int)

TwScript: TW.target.selectRange(int, int)

Position the cursor at the int position from the beginning of the document, and then selects as much text as indicated by the second integer.

```
TW.target.selectRange(100, 20);
```

If the text is available, would select twenty characters starting ant the 100th character of the document.

To merely position the cursor at a position from the beginning of the document see selectRange(int)

Example Using Propriety Latex Macro

```
txt = TW.target.selection;

if (txt == "")
  {txt = "Write Heading"}

TocLevel =  TW.getItem( null, "TOC Level", "0 - none,\n1 - section,\n2 - subsection ,\n3 - subsubsection", ["0","1","2","3"] , 1 , true ) ;

if (TocLevel != undefined)
 {
  TW.target.insertText("\n%{fontsize} {fontleading} {text} {Toc Level (1 = section) - 1,2,3}\n\
\centreHeadToc{13} {15} {" + txt + "} {" + TocLevel + "}");

 TW.target.selectRange(TW.target.selectionStart - 4 - txt.length, txt.length);

  }

  null;
```

LaTeX

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

selection

TwScript: TW.target.selection

Returns selected text in TW.target or an empty string if nothing is selected.

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

selectionLength

TwScript: TW.target.selectionLength;

Gives length of selection in TW.target, or 0 if no text is selected.

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

selectionStart

TwScript: TW.target.selectionStart;

Gives position of selection start in TW.target, or the cursor if no text is selected.

Read only.

Created with the Personal Edition of HelpNDoc: Free help authoring environment

cursorPosition()

TwScript: TW.target.cursorPosition();

This appears in revision 1019, and I have not had an executable to try it out on yet.

From Stefan Löffler's comments on http://code.google.com/p/texworks/source/detail?r=1019

> ... Works like TW.target.selectionStart.
>
> If there is no selection, returns the same
> value as TW.taget.selectionStart.
>
> Otherwise, typically returns either
> TW.target.selectionStart or (TW.target.selectionStart + TW.target.selectionEnd),
> depending on where the cursor is positioned.

TW.target.selectionStart + TW.target.selectionLength is probably meant.

Read only.

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

setWindowModified(bool)

possibly not useful?

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

text

TwScript: TW.target.text

returns all the text in the editor (read only)

```
TW.information(null, "Example", TW.target.text);
```

To replace all text, perhaps something like,

```
TW.target.selectAll();

TW.target.insertText("hello");
```

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

toLowercase()

TwScript: TW.target.toLowercase()

As stated, works on sletected text.

Created with the Personal Edition of HelpNDoc: Full featured Help generator

toUppercase()

TwScript: TW.target.toUppercase()

As stated, works on sletected text.

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

toggleCase()

TwScript: TW.target.toggleCase()

As stated, works on sletected text.

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Qt (not relevant and yet to be classified)

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

baseSize

TwScript: TW.target.baseSize

Read only.

See Qt 4.4 below.

Example:

```
var baseSize = TW.target.baseSize;
```

.cpp

Provided in Qt framework.

Qt 4.4 : QSize

This property holds the base size of the widget.

The base size is used to calculate a proper widget size if the widget defines sizeIncrement().

Access functions:

- QSize baseSize () const
- void setBaseSize (const QSize &)
- void setBaseSize (int *basew*, int *baseh*)()

See also setSizeIncrement().

Reference: http://www.greyc.ensicaen.fr/ensicaen/Docs/Qt4/qwidget.html#baseSize-prop

Created with the Personal Edition of HelpNDoc: Free help authoring tool

childrenRect

TwScript: TW.target.childrenRect QVariant(QRect)

Read only.

See Qt 4.4 below.

Example:

```
var childrenRect = TW.target.childrenRect;
```

.cpp

Provided in Qt framework.

Qt 4.4 childrenRect : const QRect

This property holds the bounding rectangle of the widget's children.

Hidden children are excluded.

Access functions:

- QRect childrenRect () const

See also childrenRegion() and geometry().

Reference: http://www.greyc.ensicaen.fr/ensicaen/Docs/Qt4/qwidget.html#childrenRect-prop

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

childrenRegion

TwScript: TW.target.childrenRegion QVariant(QRegion)

Read only.

See Qt 4.4 below.

Example:

```
var mYchildrenRegion = TW.target.childrenRegion;
var howMany = mYchildrenRegion.numRects;

TW.information(null, "childrenRegion Rectangles", howMany);
```

Will show a blank message Box -

- illustrating that sometimes (seemingly randomly you can access properties and functions from Script and sometimes you can not dig deeper into Qt's underlying framework.

.cpp

Provided in Qt framework.

Qt 4.4 childrenRect : const QRect

This property holds the bounding rectangle of the widget's children.

Hidden children are excluded.

Access functions:

- QRect childrenRect () const

See also childrenRegion() and geometry().

Reference: http://www.greyc.ensicaen.fr/ensicaen/Docs/Qt4/qwidget.

html#childrenRect-prop

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

deleteLater()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler] use other methods like close() or quit() or something instead

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

destroyed()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring tool

destroyed(QObject*)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

inputMethodHints

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

interrupt()

[?]

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

mouseTracking

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

openAt(QAction*)

openAt(QAction*)

Created with the Personal Edition of HelpNDoc: Free Web Help generator

runHooks(QString)

runHooks(QString)

(Treat as internal and leave alone for now)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

runScript(QObject*)

runScript(QObject*)

(Treat as internal and leave alone for now)

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

runScript(QObject*,TWScript::ScriptType)

(Treat as internal and leave alone for now)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

setAnimated(bool)

setAnimated(bool)

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

tabShape

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

toolButtonStyle

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

toolButtonStyleChanged(Qt::ToolButtonStyle)

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

unifiedTitleAndToolBarOnMac

Created with the Personal Edition of HelpNDoc: Easily create Help documents

update()

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

updateMicroFocus()

Created with the Personal Edition of HelpNDoc: Full featured Help generator

whatsThis

Created with the Personal Edition of HelpNDoc: Free help authoring tool

System Related

Created with the Personal Edition of HelpNDoc: Free help authoring environment

acceptDrops

It appears that this property can be read to determine if the current editor window will accept objects dropped onto it.

For example Tw will accept images and place a LaTeX \includegraphics{real image path} in the document at the cursor.

This property is read only.

Qt acceptDrops : bool

This property holds whether drop events are enabled for this widget.

Setting this property to true announces to the system that this widget may be able to accept drop events.

If the widget is the desktop (QWidget::(windowType() == Qt::Desktop)), this may fail if another application is using the desktop; you can call acceptDrops() to test if this occurs.

Warning: Do not modify this property in a Drag&Drop event handler.

Access functions:

bool acceptDrops () const
void setAcceptDrops (bool on)

Reference: http://www.greyc.ensicaen.fr/ensicaen/Docs/Qt4/qwidget.html#acceptDrops-prop

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

Menu Related

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

contextMenuPolicy

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

customContextMenuRequested(QPoint)

customContextMenuRequested(QPoint)

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

File Related

Some of these are as for the File Menu items and while actionable, some do not return anything to script at all.

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

fileName

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

newFile()

TW.target.newFile()

No return value;

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

newFromTemplate()

TW.target.newFromTemplate()

Brings up dialogue box.

No return value;

Created with the Personal Edition of HelpNDoc: Easily create Help documents

open()

TW.target.open()

No return value;

Created with the Personal Edition of HelpNDoc: Full featured Help generator

removeAuxFiles()

TW.target.removeAuxFiles()

Brings up dialogue for managing the cleansing of no longer needed .aux and related files from TW.target document's directory.

Returns nothing to script.

Created with the Personal Edition of HelpNDoc: Free help authoring environment

revert()

TW.target.revert();

Shows a dialogue box offering to retrieve last saved version of document from disk and discard recent edits in document.

Returns nothing to Script.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

save()

TW.target.save();

Seems to always return true - not sure why or if it could ever return false

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

saveAll()

TW.target.saveAll();

Seems to always return true - not sure why or if it could ever return false

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

saveAs()

TW.target.saveAs();

Presents the Save As dialogue.

Returns True fro a sucessful save, and False on Cancellation.

Created with the Personal Edition of HelpNDoc: Free help authoring tool

windowFilePath

TW.target.windowFilePath

Returns nothing.

Not writable.

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

find - replace

As per Menu Items.

Mostly nothing returned to script.

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

copyToFind()

TW.target.copyToFind()

Whatever is selected in the document will be copied to the find dialogue find field.

Created with the Personal Edition of HelpNDoc: Free help authoring tool

copyToReplace()

TW.target.copyToReplace()

Whatever is selected in the document will be copied to the replace dialogue find field.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

doFindAgain()

TW.target.doFindAgain();

As stated.

Noting returned to script.

Created with the Personal Edition of HelpNDoc: Free Web Help generator

doFindAgain(bool)

TW.target.doFindAgain(bool)

If in a document editor this seems to behave no differently than the doFindAgain() which by default is actually set to doFindAgain(false)

Also the actual settings in the Find Dialogue will effect what happens, for example if "Find All Occurrences" is ticked, that may mean that instead of moving the cursor selection to the found word, the found list pane will move to it's result line, without directly selecting the actual discovery in the Document editor.

In the pdf previewer if the script is called, doFindAgain(true) may be meant to start form the top again?

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

doFindDialog()

TW.target.doFindDialog()

As stated.

Created with the Personal Edition of HelpNDoc: Free help authoring environment

doReplaceDialog()

TW.target.doReplaceDialog()

As stated.

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

findSelection()

TW.target.findSelection()

As stated, looks for the next occurrence of the selected text. If nothing is selected it appears that the previously found item is looked for again.

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

Window

A lot of these window functions and procedures, would be useful when working with —

TW.app.getOpenWindows()
and
TW.app.openFileFromScript(QString,TW,int,bool)

Most of these Window routines are well enough named to describe themselves. Some may be redundant or not accessible to Script depending on Qt releases.

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

close()

TW.target.close();

Closes the current document. This would normally be the document that had the focus when the script was called.

---------- TEXDOCUMENT.CPP
[173] connect(actionClose, SIGNAL(triggered()), this, SLOT(close()));

Created with the Personal Edition of HelpNDoc: Full featured Help generator

goToPreview()

Created with the Personal Edition of HelpNDoc: Easily create Help documents

selectWindow()

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

selectWindow(bool)

Created with the Personal Edition of HelpNDoc: Free help authoring environment

setFocus()

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

setHidden(bool)

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

setVisible(bool)

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

show()

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

showFullScreen()

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

showMaximized()

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

showMinimized()

Created with the Personal Edition of HelpNDoc: Free help authoring tool

showNormal()

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

sideBySide()

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

maximized

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

isActiveWindow

Created with the Personal Edition of HelpNDoc: Free help authoring environment

maximumSize

maximumSize
QVariant(QSize)

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

minimumWidth

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

placeOnLeft()

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

placeOnRight()

Created with the Personal Edition of HelpNDoc: Free help authoring environment

animated

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

autoFillBackground

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

dockNestingEnabled

Created with the Personal Edition of HelpNDoc: Free Web Help generator

dockOptions

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

frameGeometry

QVariant(QRect)

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

frameSize

QVariant(QSize)

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

fullScreen

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

geometry

QVariant(QRect)

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

height

Created with the Personal Edition of HelpNDoc: Free help authoring environment

hide()

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

setDisabled(bool)

setDisabled(bool)

Created with the Personal Edition of HelpNDoc: Free help authoring tool

setDockNestingEnabled(bool)

setDockNestingEnabled(bool)

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

setShown(bool)

setShown(bool)

void QWidget::setShown (bool *shown*) `[slot]`

Use setVisible(*shown*) instead.

See also isShown().

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

setStyleSheet(QString)

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

visible

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

width

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

Don't Use - Or Possibly Not Useful

Created with the Personal Edition of HelpNDoc: Full featured Help generator

iconSize

QVariant(QSize)

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

iconSizeChanged(QSize)

iconSizeChanged(QSize)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

maximumHeight

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

minimumSizeHint

QVariant(QSize)

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

normalGeometry

QVariant(QRect)

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

palette

QVariant(QPalette)

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

pos

QVariant(QPoint)

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

raise()

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

rect

QVariant(QRect)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

repaint()

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

setEnabled(bool)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

size

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

sizeHint

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

sizeIncrement

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

sizePolicy

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

styleSheet

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

windowIcon

Created with the Personal Edition of HelpNDoc: Easily create Help documents

windowIconText

Created with the Personal Edition of HelpNDoc: Free help authoring environment

windowModality

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

windowOpacity

Created with the Personal Edition of HelpNDoc: Free Web Help generator

x

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

y

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

activatedWindow(QWidget*)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

updateScriptsMenu()

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

updatesEnabled

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

Spelling

See Spell Checking

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

spellcheckLanguage

See Spell Checking

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

reloadSpellcheckerMenu()

See Spell Checking

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

setSpellcheckLanguage()

See Spell Checking

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

createUI dialogues

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

createUI(QString)

createUI(QString : filepath)

Make a form in Qt Creator and save to your script directory, and load in your script.

If you base the form on the name of your .js script, you could do someting like this

```
var ans = TW.createUI(__FILE__.replace(".js",".ui"));
```

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

createUI(QString,QWidget*)

createUI(QString,QWidget*)

filename, owning object

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

createUIFromString(QString)

createUIFromString(QString)

Qt Qdialog XML

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

createUIFromString(QString,QWidget*)

TW.createUIFromString(QString,QWidget*)

Qt Qdialog XML

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Free help authoring tool

findChildWidget(QWidget*,QString)

TW.findChildWidget(QWidget*,QString)

Owning Form Script Object, Object Name

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

makeConnection(QObject*,QString,QObject*,QString)

[?] again, use with care; probably an "internal" method as well remarks - Stefan Löffler]

See UI Dialogue/Form Scripts

Created with the Personal Edition of HelpNDoc: Full featured Help generator

message boxes

See msgBox for button constants used in message boxes.

```
(2) Message boxes: as of r592, there are four methods that present simple alert boxes:
  TW.information( parent, title, text [, buttons [, default ] ] )
  TW.question( parent, title, text [, buttons [, default ] ] )
  TW.warning ( parent, title, text [, buttons [, default ] ] )
  TW.critical( parent, title, text [, buttons [, default ] ] )
All take the same parameters:
  - parent: either null, or a reference to a TeXworks window (normally TW.target) that
the alert should be
associated with
  - title: string to use as the window title. NOTE that this may or may not actually be
displayed, depending on
the OS, so it should not be used for important content!
  - text: string to display in the message box
You can optionally specify one or more buttons to display (default is just an "OK"
button), and also which
button should be the default. These are specified via numeric flags; see the
QMessageBox documentation for
values.
The four "flavors" of message box may or may not differ significantly in visual
appearance, depending on the
platform (and possibly Qt style) in use. Choose the appropriate one for the situation,
but don't rely on precise
appearance.
```

http://code.google.com/p/texworks/issues/detail?id=221

default is to be one of the button constants

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

critical

```
TW.critical( parent, title, text [, buttons [, default ] ] )
```

See msgBox for button constants used in message boxes.

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

critical(QWidget*,QString,QString)

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

critical(QWidget*,QString,QString,int)

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

critical(QWidget*,QString,QString,int,int)

Created with the Personal Edition of HelpNDoc: Free help authoring environment

information

```
TW.information( parent, title, text [, buttons [, default ] ] )
```

See msgBox for button constants used in message boxes.

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Free help authoring environment

information(QWidget*,QString,QString)

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

information(QWidget*,QString,QString,int)

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

information(QWidget*,QString,QString,int,int)

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

question

```
TW.question( parent, title, text [, buttons [, default ] ] )
```

See msgBox for button constants used in message boxes.

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

question(QWidget*,QString,QString)

Created with the Personal Edition of HelpNDoc: Free Web Help generator

question(QWidget*,QString,QString,int)

Created with the Personal Edition of HelpNDoc: Free help authoring environment

question(QWidget*,QString,QString,int,int)

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

warning

```
TW.warning ( parent, title, text [, buttons [, default ] ] )
```

See msgBox for button constants used in message boxes.

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Free Web Help generator

warning(QWidget*,QString,QString,int,int)

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

warning(QWidget*,QString,QString,int)

Created with the Personal Edition of HelpNDoc: Free Web Help generator

warning(QWidget*,QString,QString)

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

progressDialog(QWidget*)

TW.progressDialog(QWidget*)

```
"The attached patch adds a progress dialog to the dialogs available to scripts. Create
it by:
 d = TW.progressDialog( parent )
Then, you can set the range by
 d.setRange( min, max )
the label (indicating what the script is currently doing) by
 d.setLabelText( str )
and the current value by
 d.setValue( val )

The dialog will open when it is created, and will close when the max value of the
range is reached (you can also close it prematurely by d.reset()).
Note: Right now, the cancel button is removed from the dialog as there is no default
way to determine whether it was clicked or not from a script.
```

http://code.google.com/p/texworks/issues/detail?id=221

N.B. This dialogue needs to call .deleteLater() when you are finished with it.
e.g.
d.deleteLater();

See bottom of http://tug.org/mailman/htdig/texworks/2011q2/004415.html

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

get User Input

```
"(3) Input dialogs: there are also four simple input dialogs:
  TW.getInt( parent, title, label [, value [, min [, max [, step ] ] ] ] )
  TW.getDouble( parent, title, label [, value [, min [, max [, decimals ] ] ] ] )
  TW.getItem( parent, title, label, items [, currentIndex [, editable? ] ] )
  TW.getText( parent, title, label [, initialText ] )

Again, parent must be null or a TW window; and title may or may not be displayed. Label
should be used as the primary prompt string. All will return null (which is distinct
from zero or an empty string) if the user cancels the dialog.
```

http://code.google.com/p/texworks/issues/detail?id=221

I've found the undefined object being returned on Cancel - Paul

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

get Text

```
TW.getText( parent, title, label [, initialText ] )
```

See: getText(QWidget*,QString,QString)

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

getText(QWidget*,QString,QString)

TW.getText(QWidget widget, QString boxTitle, QString infoText [, suggestionPromptText QString]);

Returns text entered by User in an input area.

for widget generally enter null.

boxTitle (at the top of the box) should not be relied on as it may not show in all OS-es

infoText appears close to the text input area

Ok button returns whatever is in the text input area, which can optionally be primed by suggestionPromptText.

Cancel button returns the object undefined

```
var returned = TW.getText(null, "This Might Not be Seen", "Your Name: ", "Mr., Mrs... ");

    if (returned != undefined)
      {
       TW.information(null, "Answer", returned);
      }

   var returned = TW.getText(null, "This Might Not be Seen", "Hair colour: ");

    if (returned != undefined)
      {
       TW.information(null, "Hair Colour: ", returned);
      }
```

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

getText(QWidget*,QString,QString,QString)

Overload form of getText(QWidget*,QString,QString)

The extra string is a suggestion that will appear in the editable input text area.

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

TW.getItem - get List Choice

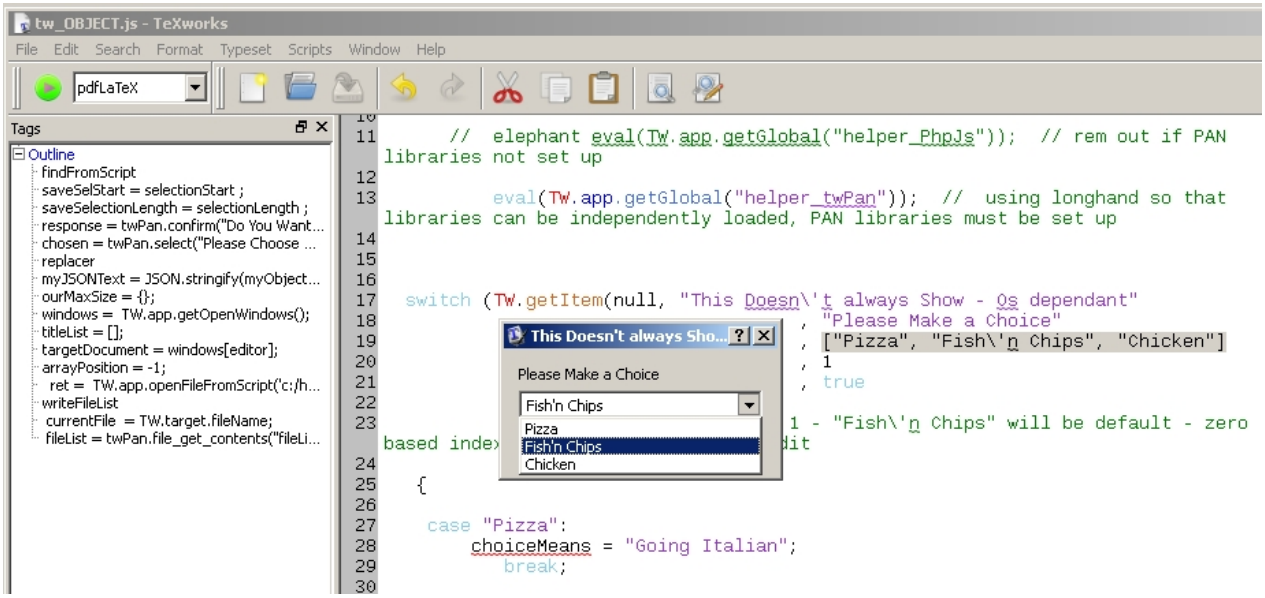
```
TW.getItem( parent, title, label, items [, currentIndex [, editable? ] ] )
```

http://code.google.com/p/texworks/issues/detail?id=221

A cancel by the User returns the object: undefined

items is an array and could be directly entered in the function.

```
 switch (TW.getItem(null, "This Doesn\'t always Show - Os dependant"
                          , "Please Make a Choice"
                          , ["Pizza", "Fish\'n Chips", "Chicken"]
                          , 1
                          , true

                       )//  1 - "Fish\'n Chips" will be default - zero based index, true
- here means can edit
         )
  {

   case "Pizza":
      choiceMeans = "Going Italian";
        break;

   case "Fish\'n Chips":
      choiceMeans = "Going British";
        break;

  case "Chicken":
      choiceMeans = "Universal";
        break;

 case undefined:
      choiceMeans = "Going Hungry";
        break;

default :  // edited by User
       choiceMeans = "Personal Choices Not Allowed";
        break;

  } // End.  switch (TW.getItem(

TW.information(null,"Tw Message", choiceMeans);
```



Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

getItem(QWidget*,QString,QString,QStringList)

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

getItem(QWidget*,QString,QString,QStringList,int)

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

getItem(QWidget*,QString,QString,QStringList,int,bool)

Created with the Personal Edition of HelpNDoc: Easily create Help documents

get Numerical Input

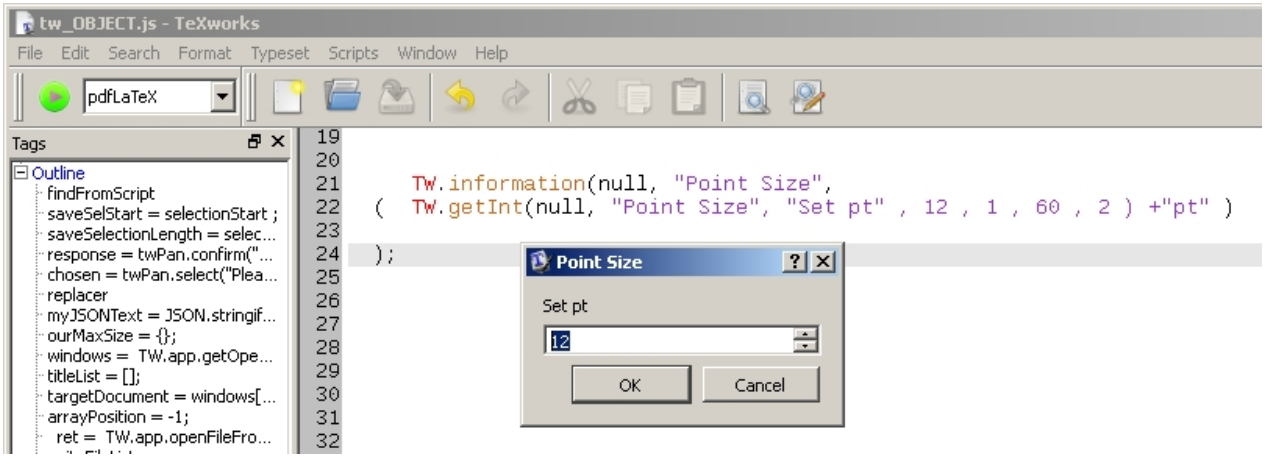
```
TW.getInt( parent, title, label [, value [, min [, max [, step ] ] ] ] )
TW.getDouble( parent, title, label [, value [, min [, max [, decimals ] ] ] ] )
```

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

getInt(QWidget*,QString,QString,int,int,int,int)

```
TW.getInt( parent, title, label [, value [, min [, max [, step ] ] ] ] )
```



Cancel returns: undefined

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

getInt(QWidget*,QString,QString,int,int,int)

Created with the Personal Edition of HelpNDoc: Full featured Help generator

getInt(QWidget*,QString,QString,int,int)

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

getInt(QWidget*,QString,QString,int)

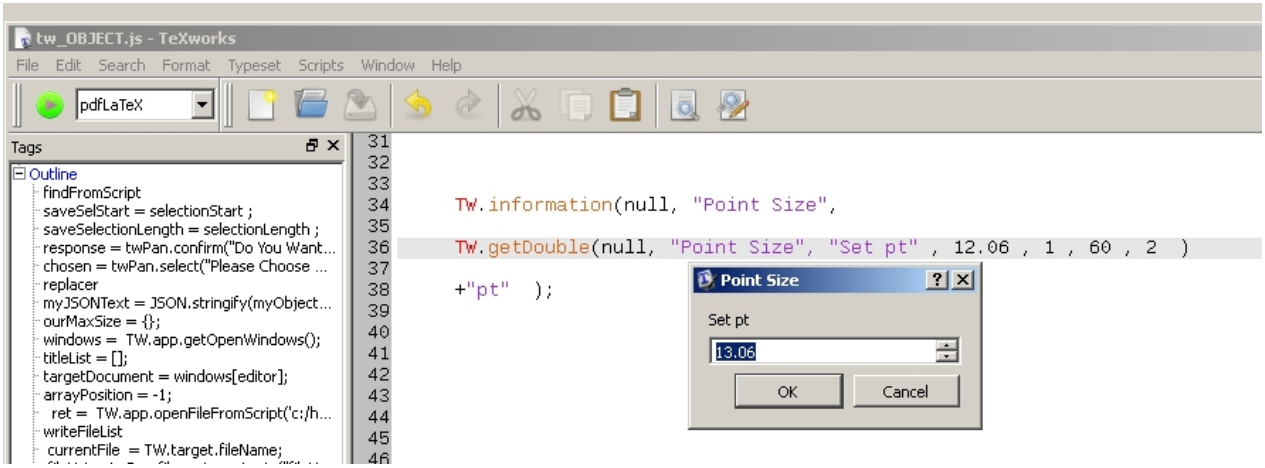
Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

getInt(QWidget*,QString,QString)

Created with the Personal Edition of HelpNDoc: Free help authoring tool

getDouble(QWidget*,QString,QString)

```
TW.getDouble( parent, title, label [, value [, min [, max [, decimals ] ] ] ] )
```



Picture shows the result after one upward click on the top small arrow, the interval is one by default here.

Cancel returns: undefined

http://code.google.com/p/texworks/issues/detail?id=221

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

getDouble(QWidget*,QString,QString,double,double,double,int)

Created with the Personal Edition of HelpNDoc: Full featured Help generator

getDouble(QWidget*,QString,QString,double,double,double)

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

getDouble(QWidget*,QString,QString,double,double)

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

getDouble(QWidget*,QString,QString,double)

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

other top level elements

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

destroyed()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Free help authoring tool

destroyed(QObject*)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

deleteLater()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

platform()

TW.platform()

returns a String one of,

"MacOSX" "Windows" "X11" "unknown"

Which could be used as--

In QTScript

```
switch(TW.platform())
  {
   case "MacOSX":

     // do someting Mac Specific
       break;

   case "Windows":

     // do someting Windows specific
       break;

   case  "X11":

     // do someting Linux specific
   break;

   case  "unknown":
    // what ever
   break;

   // probably no need for default: here

  } // /End. switch(TW.platform())
```

Created with the Personal Edition of HelpNDoc: Full featured Help generator

result

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

strlen(QString)

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

yield()

Allows Tw to process any application messages - in essence sort of keeps things form freezing up, perhaos use in side any loops that you know will take a while.

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

script

For script properties readible in script
.fileName .title .description .author .version

see Script properties (fileName etc)

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Script properties (fileName etc)

TwScript: TW.script.*propertyName*

Returns a string.

propertyNames—

.fileName .title .description .author .version

These are as stated for the property names, for the script,
(and not for the TeXworks document that is being edited).

These properties are read only, there is no persistence for any changes attempted in script.

Note: TW.script.fileName is more reliable than __FILE__

TW.script.fileName

Not for the TeXworks document that is being edited, (for that see TW.target.fileName).

It is the path and file name for the actual script that is reading the property —

```
/path/scriptName.js

c:/texworks/myconfig/scripts/text/scriptName.js
```

.title through to .version are set by the script author in the header of the script.

```
// TeXworksScript
// Title:  Shortcuts for Latex text
// Description: User abbreviations for text
// Author:  Paul A Norman
// Version: 0.1
```

```
TW.script.title
TW.script.description
TW.script.author
TW.script.version
```

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

deleteLater()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

destroyed()

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

destroyed(QObject*)

[X] - Dont Use - Not Available/Not Useful [remarks - Stefan Löffler]

Created with the Personal Edition of HelpNDoc: Easily create Help documents

getGlobal(QString)

See Globals

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

hasGlobal(QString)

See Globals

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

objectName

Created with the Personal Edition of HelpNDoc: Free help authoring environment

setGlobal(QString,QVariant)

See Globals

Created with the Personal Edition of HelpNDoc: Full featured Help generator

unsetGlobal(Qstring)

TW.script.unsetGlobal("whatever")

See
Globals

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

Introductory QtScript (ECMA)

QtScript very much conforms to any experience that you may have had with JavaScript, with additions from the designers of the Qt framework.

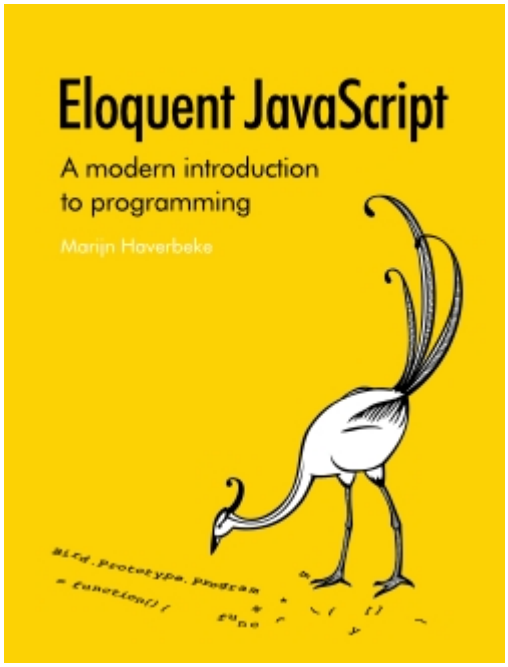
I had intended to put some notes here introducing core JavaScript concepts and details, but believe that these links will be a far better start off, in all the JavaScript links in this document you'll need to separate out any information that is intended to be browser specific.
The core of ECMA is not browser specific, but of course many available tutorials are.
Perhaps this following tutorial may be very good to start with for any one either starting to use QtScript EMCA (JavaScript) or wanting to reacquaint themselves with it, and recent advances.

The online and downloadable digital versions are currently (2011/07) free and a print version is available. Of particular help may be the book's (online as well) page on regular expressions http://eloquentjavascript.net/chapter10.html
(For another ground up explanation also see http://www.websina.com/bugzero/kb/regexp.html)

Eloquent JavaScript
A Modern Introduction to Programming
by Marijn Haverbeke

Eloquent JavaScript is a book providing an introduction to the JavaScript programming language and programming in general.

> "A concise and balanced mix of principles and pragmatics. I loved the tutorial-style game-like program development. This book rekindled my earliest joys of programming.
> Plus, JavaScript!"
>
> —Brendan Eich, the man who gave us JavaScript



http://eloquentjavascript.net/

Here are two links (with a bit of cross-over) to a few more free online and downloadable books http://www.readwriteweb.com/hack/2010/12/6-free-javascript-e-books.php

from there special mention of a book still being written - (may be you can help?) —

> **Wiki Books JavaScript**
>
> This book is a guide to JavaScript, a scripting language widely used in web pages and web applications such as email applications. JavaScript is not to be confused with Java, which is quite a separate language for creating stand-alone applications.

and another list at http://www.e-booksdirectory.com/listing.php?category=424, of special mention from that site are are first again —

> **Eloquent JavaScript: An opinionated guide to programming**
> by **Marijn Haverbeke** , **2008**
> This ebook provides a comprehensive introduction to the JavaScript programming language. It contains plenty of example programs, and an environment to try them out and play with them. The book is aimed at the beginning programmer.
>
> **Core JavaScript Guide**
> - **Netscape Communications Corp.** , **2000**
> JavaScript is a cross-platform, object-based scripting language. This book explains everything you need to know about using core JavaScript. It is assumed that you have a general understanding of the WWW, and a good working knowledge of HTML.
>
> **Building A JavaScript Framework**
> by **Alex Young** - **DailyJS** , **2010**
> This book is a guide to building a JavaScript framework. It'll teach you how to build a framework and draw on real-world code from projects like jQuery. Along the way we'll explore some fundamental parts of modern JavaScript.
>
> **Up and Running with Node.js**
> by **Tom Hughes-Croucher** - **O'Reilly Media** , **2010**
> Node.js, the new web development framework written in JavaScript, is many things, but mostly it's a way of running JavaScript outside the web browser. This book will cover why that's important, and what benefits Node.js provides.

(Also see note about Qt Quick and QML below.)

Here in this document under JavaScript Garden are some notes copied from http://bonsaiden.github.com/JavaScript-Garden/

Mozilla foundation and related groups provide much online material.

> https://developer.mozilla.org/en/JavaScript/Guide
>
> https://developer.mozilla.org/en/javascript
>
> and for a complete list of Mozilla resources/knowledge: https://developer.mozilla.org/Special:Tags?tag=JavaScript&language=en

Devguru.com provide a free pdf version of a useful reference.

A particularly useful resource

> http://www.readwriteweb.com/hack/2010/12/6-free-javascript-e-books.php

... lists these free tutorials as available for download or use on line:

Eloquent JavaScript is a JavaScript book by Marijn Haverbeke. The digital form is free http://eloquentjavascript.net/

Sams Teach Yourself JavaScript in 24 Hours by Michael Moncur is part of the well known series of Sams books.

W3Schools is one of the most venerable and respected resources for online tutorials, and of course it has its own JavaScript tutorial. It also has a stash of JavaScript examples, including some advanced scripts.

SitePoint is another well respected online source for tutorials and books. It has a few free tutorials, mostly as previews for its books. Its JavaScript Guru List is a good place to start.

Well known JavaScript expert and ***JavaScript: The Good Parts*** author Douglas Crockford has a series of free lectures available on history of JavaScript, its features, and its use.

Introduction to Core JavaScript/ECMA-262

"Welcome to the Web Site built specifically for JavaScript practitioners ...

This site addresses the Standard ECMA-262 (ECMAScript Language Specification). The core of JavaScript is based on the ECMAScript Language Specification. Our purpose is to explain core JavaScript."

http://rx4ajax-jscore.com/index.html

EMCA-262

by Dmitry A. Soshnikov - September 2nd, 2010

http://dmitrysoshnikov.com/tag/ecmascript/

http://dmitrysoshnikov.com/ecmascript/javascript-the-core/

ECMA Quick Reference

http://www.devguru.com/technologies/JavaScript/home.asp
"Download free PDF version ... "

http://www.devguru.com/technologies/JavaScript/JavaScript_PDF806.zip

http://www.devguru.com/technologies/ecmascript/quickref/javascript_index.html

A more comparative overview can be found here:—

Using JavaScript as a Real Programming Language

Tommi Mikkonen and Antero Taivalsaari

"Software systems and applications that were conventionally written using a static programming language such as C, C++ or Java™, are now built with dynamic languages that were originally designed for scripting rather than fullscale, general-purpose application development."
http://labs.oracle.com/techrep/2007/abstract-168.html

http://labs.oracle.com/techrep/2007/smli_tr-2007-168.pdfQt Quick - QML

Qt Nokia have also released a new development paradigm, in Qt 4.7, utilising JavaScript heavily.

Called Qt Quick, it creates a new declarative language, the QML language, this may in the future create modules accessible to QtScript developers. See QML, Elements and the Qt Declarative Module and http://doc.qt.nokia.com/4.7-snapshot/qml-intro.html for more information.

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

Language Fundamentals

Language Fundamentals

Overview

This is from the Qt site: http://doc.trolltech.com/4.4/ecmascript.html

(The latest currently is: http://doc.trolltech.com/4.7/ecmascript.html)

Licensees holding valid Qt Commercial licenses may use this document in accordance with the Qt Commercial License Agreement provided with the Software or, alternatively, in accordance with the terms contained in a written agreement between you and Nokia.
Alternatively, this document may be used under the terms of the GNU Free Documentation License version 1.3
as published by the Free Software Foundation.) © 2008-2010 Nokia Corporation and/or its subsidiaries. Nokia, Qt and their respective logos are trademarks of Nokia Corporation in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.

ECMAScript Reference

This reference contains a list of objects, functions and properties supported by QtScript.

-

The Global Object

Value Properties

- NaN
- Infinity
- undefined
- Math

Function Properties

- eval(x)
- parseInt(string, radix)
- parseFloat(string)
- isNaN(number)
- isFinite(number)
- decodeURI(encodedURI)
- decodeURIComponent(encodedURIComponent)
- encodeURI(uri)
- encodeURIComponent(uriComponent)

Constructor Properties

- Object
- Function
- Array
- String
- Boolean
- Number
- Date
- RegExp
- Error
- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Object Objects

Object Prototype Object

- toString()
- toLocaleString()
- valueOf()
- hasOwnProperty(V)
- isPrototypeOf(V)
- propertyIsEnumerable(V)

Function Objects

Function Prototype Object

Function Properties

- toString()
- apply(thisArg, argArray)
- call(thisArg [, arg1 [, arg2, ...]])

Array Objects

Array Prototype Object

Function Properties

- toString()
- toLocaleString()
- concat([item1 [, item2 [, ...]]])
- join(separator)
- pop()
- push([item1 [, item2 [, ...]]])
- reverse()
- shift()
- slice(start, end)
- sort(comparefn)
- splice(start, deleteCount[, item1 [, item2 [, ...]]])
- unshift([item1 [, item2 [, ...]]])

String Objects

String Prototype Object

Function Properties

- toString()
- valueOf()
- charAt(pos)
- charCodeAt(pos)
- concat([string1 [, string2 [, ...]]])
- indexOf(searchString ,position)
- lastIndexOf(searchString, position)
- localeCompare(that)
- match(regexp)
- replace(searchValue, replaceValue)
- search(regexp)
- slice(start, end)
- split(separator, limit)
- substring(start, end)
- toLowerCase()
- toLocaleLowerCase()
- toUpperCase()
- toLocaleUpperCase()

Boolean Objects

Boolean Prototype Object

Function Properties

- toString()
- valueOf()

Number Objects

Number Prototype Object

Function Properties

- toString(radix)
- toLocaleString()
- toFixed(fractionDigits)

- toExponential(fractionDigits)
- toPrecision(precision)

The Math Object

Value Properties

- E
- LN10
- LN2
- LOG2E
- LOG10E
- PI
- SQRT1_2
- SQRT2

Function Properties

- abs(x)
- acos(x)
- asin(x)
- atan(x)
- atan2(y, x)
- ceil(x)
- cos(x)
- exp(x)
- floor(x)
- log(x)
- max([value1 [, value2 [, ...]]])
- min([value1 [, value2 [, ...]]])
- pow(x, y)
- random()
- round(x)
- sin(x)
- sqrt(x)
- tan(x)

Date Objects

Date Prototype Object

Function Properties

- toString()
- toDateString()
- toTimeString()
- toLocaleString()
- toLocaleDateString()
- toLocaleTimeString()
- valueOf()
- getTime()
- getFullYear()
- getUTCFullYear()
- getMonth()
- getUTCMonth()
- getDate()
- getUTCDate()
- getDay()
- getUTCDay()
- getHours()
- getUTCHours()
- getMinutes()
- getUTCMinutes()
- getSeconds()
- getUTCSeconds()
- getMilliseconds()
- getUTCMilliseconds()
- getTimeZoneOffset()
- setTime(time)
- setMilliseconds(ms)
- setUTCMilliseconds(ms)
- setSeconds(sec [, ms])
- setUTCSeconds(sec [, ms])
- setMinutes(min [, sec [, ms]])
- setUTCMinutes(min [, sec [, ms]])
- setHours(hour [, min [, sec [, ms]]])
- setUTCHours(hour [, min [, sec [, ms]]])
- setDate(date)
- setUTCDate(date)
- setMonth(month [, date])

- setUTCMonth(month [, date])
- setFullYear(year [, month [, date]])
- setUTCFullYear(year [, month [, date]])
- toUTCString()

RegExp Objects

RegExp Prototype Object

Function Properties

- exec(string)
- test(string)
- toString()

Error Objects

Error Prototype Object

Value Properties

- name
- message

Function Properties

- toString()

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

JavaScript Garden

Note: Qt have their own modifications to the Object prototype http://doc.qt.nokia.com/latest/scripting.html and specifically http://doc.qt.nokia.com/latest/scripting.html#prototype-objects-and-shared-properties

JavaScript Garden

Intro	Objects	The prototype
`hasOwnProperty`	Functions	How `this` works
Closures and references	The `arguments` object	Scopes and namespaces

Intro

JavaScript Garden is a growing collection of documentation about the most quirky parts of the JavaScript programming language. It gives advice to avoid common mistakes, subtle bugs, as well as performance issues and bad practices that non-expert JavaScript programmers may encounter on their endeavours into the depths of the language.

JavaScript Garden does **not** aim to teach you JavaScript. Former knowledge of the language is strongly recommended in order to understand the topics covered in this guide. In order to learn the basics of the language, please head over to the excellent guide on the Mozilla Developer Network.

The authors

This guide is the work of two lovely Stack Overflow users, Ivo Wetzel (Writing) and Zhang Yi Jiang (Design).

Contributors

- Caio Romão (Spelling corrections)
- Andreas Blixt (Language corrections)

License

JavaScript Garden is published under the MIT license and hosted on GitHub. If you find errors or typos please file an issue or a pull request on the repository. You can also find us in the JavaScript room on Stack Overflow chat.

http://bonsaiden.github.com/JavaScript-Garden

Copyright © 2011.

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

Objects

Closures and references	The `arguments` object	Scopes and namespaces
Constructors	Equality and comparisons	Arrays
The `Array` constructor	The `for in` loop	The `typeof` operator
The `instanceof` operator	Type casting	`undefined` and `null`
Reasons against `eval`	`setTimeout` and `setInterval`	Automatic semicolon insertion

Note: Qt have their own modifications to the Object prototype http://doc.qt.nokia.com/latest/scripting.html and specifically http://doc.qt.nokia.com/latest/scripting.html#prototype-objects-and-shared-properties

Objects

Everything in JavaScript acts like an object, with the only two exceptions being `null` and `undefined`.

```
false.toString() // 'false'
[1, 2, 3].toString(); // '1,2,3'
function Foo(){}
Foo.bar = 1;
Foo.bar; // 1
```

A common misconception is that number literals cannot be used as objects. That is because a flaw in JavaScript's parser tries to parse the *dot notation* on a number as a floating point literal.

```
2.toString(); // raises SyntaxError
```

There are a couple of workarounds which can be used in order make number literals act as objects too.

```
2..toString(); // the second point is correctly recognized
2 .toString(); // note the space left to the dot
(2).toString(); // 2 is evaluated first
```

Objects as a data type

Objects in JavaScript can also be used as a *Hashmap*, they mainly consist of named properties mapping to values.

Using a object literal - `{}` notation - it is possible to create a plain object. This new object inherits from `Object.prototype` and has no own properties defined on it.

```
var foo = {}; // a new empty object
// a new object with a property called 'test' with value 12
var bar = {test: 12};
```

Accessing properties

The properties of an object can be accessed in two ways, via either the dot

notation, or the square bracket notation.

```
var foo = {name: 'Kitten'}
foo.name; // kitten
foo['name']; // kitten
var get = 'name';
foo[get]; // kitten
foo.1234; // SyntaxError
foo['1234']; // works
```

Both notations are identical in their workings, with the only difference being that the square bracket notation allows for dynamic setting of properties, as well as the use of property names that would otherwise lead to a syntax error.

Deleting properties

The only way to actually remove a property from an object is to use the `delete` operator; setting the property to `undefined` or `null` does **only** remove the value associated with the property, but not the key.

```
var obj = {
    bar: 1,
    foo: 2,
    baz: 3
};
obj.bar = undefined;
obj.foo = null;
delete obj.baz;
for(var i in obj) {
    if (obj.hasOwnProperty(i)) {
        console.log(i, '' + obj[i]);
    }
}
```

The above outputs both `bar undefined` and `foo null` - only `baz` got actually removed and is therefore missing from the output.

Notation of keys

```
var test = {
    'case': 'I am a keyword so I must be notated as a string',
    delete: 'I am a keyword too so me' // raises SyntaxError
};
```

Object properties can be both notated as plain characters and as strings. Due to another mis-design in JavaScript's parser, the above will throw a `SyntaxError`prior to ECMAScript 5.

This error arises from the fact that `delete` is a *keyword* of the language; therefore, it must be notated as a *string literal* in order to ensure working code under older JavaScript engines.

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

The prototype #top

JavaScript does not feature a classical inheritance model, instead it uses a *prototypical* one.

While this is often considered to be one of JavaScript's weaknesses, the prototypical inheritance model is in fact more powerful than the classic model. It is for example fairly trivial to build a classic model on top of it, while the other way around is a far more difficult task.

Due to the fact that JavaScript is basically the only widely used language that features prototypical inheritance, it takes some time to adjust to the differences between the two models.

The first major difference is that inheritance in JavaScript is done by using so called *prototype chains*.

Note: Simply using `Bar.prototype = Foo.prototype` will result in both objects sharing the **same** prototype. Therefore, changes to either object's prototype will affect the prototype of the other as well, which in most cases is not the desired effect.

```
function Foo() {
    this.value = 42;
}
Foo.prototype = {
    method: function() {}
};
```

```
function Bar() {}
// Set Bar's prototype to a new instance of Foo
Bar.prototype = new Foo();
Bar.prototype.foo = 'Hello World';
// Make sure to list Bar as the actual constructor
Bar.prototype.constructor = Bar;
var test = new Bar() // create a new bar instance
// The resulting prototype chain
test [instance of Bar]
    Bar.prototype [instance of Foo]
        { foo: 'Hello World' }
        Foo.prototype
            { method: ... }
            Object.prototype
                { toString: ... /* etc. */ }
```

In the above, the object `test` will inherit from both `Bar.prototype` and `Foo.prototype`; hence, it will have access to the function `method` that was defined on`Foo`. It will also have access to the property `value` of the **one** `Foo` instance that is its prototype. It is important to note that `new Bar()` does **not** create a new `Foo`instance, but reuses the one assigned to its prototype; thus, all `Bar` instances will share the **same** `value` property.

Note: Do **not** use `Bar.prototype = Foo`, since it will not point to the prototype of `Foo` but rather to the function object `Foo`. So the prototype chain will go over`Function.prototype` and not `Foo.prototype`; therefore, `method` will not be on the prototype chain.

Property lookup

When accessing the properties of an object, JavaScript will traverse the prototype chain **upwards** until it finds a property with the requested name.

When it reaches the top of the chain - namely `Object.prototype` - and still hasn't found the specified property, it will return the value undefined instead.

The prototype property

While the prototype property is used by the language to build the prototype chains, it is still possible to assign **any** given value to it. Although primitives will simply get ignored when assigned as a prototype.

```
function Foo() {}
Foo.prototype = 1; // no effect
```

Assigning objects, as shown in the example above, will work, and allows for dynamic creation of prototype chains.

Performance

The lookup time for properties that are high up on the prototype chain can have a negative impact on performance critical sections of code. Additionally, trying to access non-existent properties will always traverse the full prototype chain.

Also, when iterating over the properties of an object **every** property that is on the prototype chain will get enumerated.

Extension of native prototypes

One mis-feature that is often used is to extend `Object.prototype` or one of the other built in prototypes.

This technique is called monkey patching and breaks *encapsulation*. While used by widely spread frameworks such as Prototype, there is still no good reason for cluttering built in types with additional *non-standard* functionality.

The **only** good reason for extending a built-in prototype is to backport the features of newer JavaScript engines; for example, `Array.forEach`.

In conclusion

It is a **must** to understand the prototypical inheritance model completely before writing complex code which makes use of it. Also, watch the length of the prototype chains and break them up if necessary to avoid possible performance issues. Further, the native prototypes should **never** be extended unless it is for the sake of compatibility with newer JavaScript features.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

hasOwnProperty

	erval	insertion

hasOwnProperty

In order to check whether a object has a property defined on *itself* and **not** somewhere on its prototype chain, it is necessary to use the `hasOwnProperty` method which all objects inherit from `Object.prototype`.

Note: It is **not** enough to check whether a property is `undefined`. The property might very well exist, but its value just happens to be set to `undefined`.

`hasOwnProperty` is the only thing in JavaScript which deals with properties and does **not** traverse the prototype chain.

```
// Poisoning Object.prototype
Object.prototype.bar = 1;
var foo = {goo: undefined};
foo.bar; // 1
'bar' in foo; // true
foo.hasOwnProperty('bar'); // false
foo.hasOwnProperty('goo'); // true
```

Only `hasOwnProperty` will give the correct and expected result, this is essential when iterating over the properties of any object. There is **no** other way to exclude properties that are not defined on the object *itself*, but somewhere on its prototype chain.

`hasOwnProperty` as a property

JavaScript does **not** protect the property name `hasOwnProperty`; thus, if the possibility exists that an object might have a property with this name, it is necessary to use an *external* `hasOwnProperty` in order to get correct results.

```
var foo = {
    hasOwnProperty: function() {
        return false;
    },
    bar: 'Here be dragons'
};
foo.hasOwnProperty('bar'); // always returns false
// Use another hasOwnProperty and call it with 'this' set to foo
{}.hasOwnProperty.call(foo, 'bar'); // true
```

In conclusion

When checking for the existence of a property on a object, `hasOwnProperty` is the **only** method of doing so. It is also recommended to make `hasOwnProperty`part of **every** `for in` loop, this will avoid errors from

extended native prototypes.

Created with the Personal Edition of HelpNDoc: Free help authoring tool

Functions

Functions in JavaScript are first class objects, that means that they can be passed around like any other value. One common use of this feature is to pass an*anonymous function* as a callback to another, possibly asynchronous function.

The `function` declaration

```
function foo() {}
```

The above function gets hoisted before the execution of the program starts; thus, it is available *everywhere* in the scope it was *defined* in, even if called before the actual definition in the source.

```
foo(); // Works because foo was created before this code runs
function foo() {}
```

The `function` expression

```
var foo = function() {};
```

This example assigns the unnamed and *anonymous* function to the variable `foo`.

```
foo; // 'undefined'
foo(); // this raises a TypeError
var foo = function() {};
```

Due to the fact that `var` is a declaration, that hoists the variable name `foo` before the actual execution of the code starts, `foo` is already defined when the script gets executed.

But since assignments only happens at runtime, the value of `foo` will default to undefined before the corresponding code is executed.

Named function expression

Another special case is the assignment of named functions.

```
var foo = function bar() {
    bar(); // Works
}
bar(); // ReferenceError
```

Here `bar` is not available in the outer scope, since the function only gets assigned to `foo`; however, inside of `bar` it is available. This is due to how name resolution in JavaScript works, the name of the function is *always* made available in the local scope of the function itself.

Created with the Personal Edition of HelpNDoc: Free help authoring environment

How `this` works #top

JavaScript has a different concept of what the special name `this` refers to than most other programming languages do. There are exactly **five** different ways in which the value of `this` can be bound in the language.

The global scope

```
this;
```

When using `this` in global scope, it will simply refer to the *global* object.

Calling a function

```
foo();
```

Here `this` will again refer to the *global* object.

ES5 Note: In strict mode, the global case **no longer** exists. `this` will instead have the value of `undefined` in that case.

Calling a method

```
test.foo();
```

In this example `this` will refer to `test`.

Calling a constructor

```
new foo();
```

A function call that is preceded by the `new` keyword acts as a constructor. Inside the function `this` will refer to a *newly created* `Object`.

Explicit setting of `this`

```
function foo(a, b, c) {}
var bar = {};
foo.apply(bar, [1, 2, 3]); // array will expand to the below
foo.call(bar, 1, 2, 3); // results in a = 1, b = 2, c = 3
```

When using the `call` or `apply` methods of `Function.prototype`, the value of `this` inside the called function gets **explicitly set** to the first argument of the corresponding function call.

As a result, the above example the *method case* does **not** apply, and `this` inside of `foo` will be set to `bar`.

Note: `this` **cannot** be used to refer to the object inside of an `Object` literal. So `var obj = {me: this}` will **not** result in `me` referring to `obj`, since `this` only gets bound by one of the five listed cases.

Common pitfalls

While most of these cases make sense, the first one is to be considered another mis-design of the language, as it **never** has any practical use.

```
Foo.method = function() {
    function test() {
        // this is set to the global object
```

```
    }
    test();
}
```

A common misconception is that `this` inside of `test` refers to `Foo`, while in fact it **does not**.

In order to gain access to `Foo` from within `test` it is necessary to create a local variable inside of `method` which refers to `Foo`.

```
Foo.method = function() {
    var that = this;
    function test() {
        // Use that instead of this here
    }
    test();
}
```

`that` is just a normal name, but it is commonly used for the reference to an outer `this`. In combination with closures, it can also be used to pass `this` values around.

Assigning methods

Another thing that does **not** work in JavaScript is function aliasing, that is, **assigning** a method to a variable.

```
var test = someObject.methodTest;
test();
```

Due to the first case `test` now acts like like a plain function call; therefore, `this` inside it will no longer refer to `someObject`.

While the late binding of `this` might seem like a bad idea at first, it is in fact what makes prototypical inheritance work.

```
function Foo() {}
Foo.prototype.method = function() {};
function Bar() {}
Bar.prototype = Foo.prototype;
new Bar().method();
```

When `method` gets called on a instance of `Bar`, `this` will now refer to that very instance.

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

Closures and references

Intro	Objects	The prototype

Closures and references

One of JavaScript's most powerful features is the availability of *closures*, this means that scopes **always** keep access to the outer scope they were defined in. Since the only scoping that JavaScript has is function scope, all functions, by default, act as closures.

Emulating private variables

```
function Counter(start) {
    var count = start;
    return {
        increment: function() {
            count++;
        },
        get: function() {
            return count;
        }
    }
}
var foo = Counter(4);
foo.increment();
foo.get(); // 5
```

Here, `Counter` returns **two** closures. The function `increment` as well as the function `get`. Both of these functions keep a **reference** to the scope of `Counter` and, therefore, always keep access to the `count` variable that was defined in that very scope.

Why private variables work

Since it is not possible to reference or assign scopes in JavaScript, there is **no** way of accessing the variable `count` from the outside. The only way to interact with it is via the two closures.

```
var foo = new Counter(4);
foo.hack = function() {
```

```
    count = 1337;
};
```

The above code will **not** change the variable `count` in the scope of `Counter`, since `foo.hack` was not defined in **that** scope. It will instead create - or override - the*global* variable `count`.

Closures inside loops

One often made mistake is to use closures inside of loops, as if they were copying the value of the loops index variable.

```
for(var i = 0; i < 10; i++) {
    setTimeout(function() {
        console.log(i);
    }, 1000);
}
```

The above will **not** output the numbers `0` through `9`, but will simply print the number `10` ten times.

The *anonymous* function keeps a **reference** to `i` and at the time `console.log` gets called, the `for loop` has already finished and the value of `i` as been set to `10`.

In order to get the desired behavior, it is necessary to create a **copy** of the value of `i`.

Avoiding the reference problem

In order to copy the value of the loop's index variable, it is best to use an anonymous wrapper.

```
for(var i = 0; i < 10; i++) {
    (function(e) {
        setTimeout(function() {
            console.log(e);
        }, 1000);
    })(i);
}
```

The anonymous outer function gets called immediately with `i` as its first argument and will receive a copy of the **value** of `i` as its parameter `e`.

The anonymous function that gets passed to `setTimeout` now has a reference to `e`, whose value does **not** get changed by the loop.

There is another possible way of achieving this; that is to return a function from the anonymous wrapper, that will then have the same behavior as the code above.

```
for(var i = 0; i < 10; i++) {
    setTimeout((function(e) {
        return function() {
            console.log(e);
```

```
        }
    })(i), 1000)
}
```

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

The `arguments` object

Every function scope in JavaScript can access the special variable `arguments`. This variable holds a list of all the arguments that were passed to the function.

Note: In case `arguments` has already been defined inside the function's scope either via a `var` statement or being the name of a formal parameter, the `arguments`object will not be created.

The `arguments` object is **not** an `Array`. While it has some of the semantics of an array - namely the `length` property - it does not inherit from `Array.prototype`and is in fact an `Object`.

Due to this, it is **not** possible to use standard array methods like `push`, `pop` or `slice` on `arguments`. While iteration with a plain `for` loop works just fine, it is necessary to convert it to a real `Array` in order to use the standard `Array` methods on it.

Converting to an array

The code below will return a new `Array` containing all the elements of the `arguments` object.

```
Array.prototype.slice.call(arguments);
```

This conversion is **slow**, it is **not recommended** to use it in performance critical sections of code.

Passing arguments

The following is the recommended way of passing arguments from one function to another.

```
function foo() {
    bar.apply(null, arguments);
}
function bar(a, b, c) {
    // do stuff here
}
```

Another trick is to use both `call` and `apply` together to create fast, unbound wrappers.

```
function Foo() {}
Foo.prototype.method = function(a, b, c) {
    console.log(this, a, b, c);
};
// Create an unbound version of "method"
// It takes the parameters: this, arg1, arg2...argN
Foo.method = function() {
    // Result: Foo.prototype.method.call(this, arg1, arg2... argN)
    Function.call.apply(Foo.prototype.method, arguments);
};
```

Formal parameters and arguments indexes

The `arguments` object creates *getter* and *setter* functions for both its properties as well as the function's formal parameters.

As a result, changing the value of a formal parameter will also change the value of the corresponding property on the `arguments` object, and the other way around.

```
function foo(a, b, c) {
    arguments[0] = 2;
    a; // 2
    b = 4;
    arguments[1]; // 4
    var d = c;
    d = 9;
    c; // 3
}
foo(1, 2, 3);
```

Performance myths and truths

The `arguments` object is always created with the only two exceptions being the cases where it is declared as a name inside of a function or one of its formal

parameters. It does not matter whether it is used or not.

Both *getters* and *setters* are **always** created; thus, using it has nearly no performance impact at all, especially not in real world code where there is more than a simple access to the `arguments` object's properties.

ES5 Note: These *getters* and *setters* are not created in strict mode.

However, there is one case which will drastically reduce the performance in modern JavaScript engines. That case is the use of `arguments.callee`.

```
function foo() {
    arguments.callee; // do something with this function object
    arguments.callee.caller; // and the calling function object
}
function bigLoop() {
    for(var i = 0; i < 100000; i++) {
        foo(); // Would normally be inlined...
    }
}
```

In the above code, `foo` can no longer be a subject to inlining since it needs to know about both itself and its caller. This not only defeats possible performance gains that would arise from inlining, it also breaks encapsulation since the function may now be dependent on a specific calling context.

It is **highly recommended** to **never** make use of `arguments.callee` or any of its properties.

ES5 Note: In strict mode, `arguments.callee` will throw a `TypeError` since its use has been deprecated.

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

Scopes and namespaces

		insertion

Scopes and namespaces

Although JavaScript deals fine with the syntax of two matching curly braces for blocks, it does **not** support block scope; hence, all that is left is in the language is*function scope*.

```
function test() { // a scope
    for(var i = 0; i < 10; i++) { // not a scope
        // count
    }
    console.log(i); // 10
}
```

Note: When not used in an assignment, return statement or as a function argument, the `{...}` notation will get interpreted as a block statement and **not** as an object literal. This, in conjunction with automatic insertion of semicolons, can lead to subtle errors.

There are also no distinct namespaces in JavaScript, that means that everything gets defined in one *globally shared* namespace.

Each time a variable is referenced, JavaScript will traverse upwards through all the scopes until it finds it. In the case that it reaches the global scope and still has not found the requested name, it will raise a `ReferenceError`.

The bane of global variables

```
// script A
foo = '42';
// script B
var foo = '42'
```

The above two scripts do **not** have the same effect. Script A defines a variable called `foo` in the *global* scope and script B defines a `foo` in the *current* scope.

Again, that is **not** at all the *same effect*, not using `var` can have major implications.

```
// global scope
var foo = 42;
function test() {
    // local scope
    foo = 21;
}
test();
foo; // 21
```

Leaving out the `var` statement inside the function `test` will override the value of `foo`. While this might not seem like a big deal at first, having thousands of lines of JavaScript and not using `var` will introduce horrible and hard to track down

bugs.

```
// global scope
var items = [/* some list */];
for(var i = 0; i < 10; i++) {
    subLoop();
}
function subLoop() {
    // scope of subLoop
    for(i = 0; i < 10; i++) { // missing var statement
        // do amazing stuff!
    }
}
```

The outer loop will terminate after the first call to `subLoop`, since `subLoop` overwrites the global value of `i`. Using a `var` for the second `for` loop would have easily avoided this error. The `var` statement should **never** be left out unless the *desired effect* is to affect the outer scope.

Local variables

The only source for local variables in JavaScript are function parameters and variables that were declared via the `var` statement.

```
// global scope
var foo = 1;
var bar = 2;
var i = 2;
function test(i) {
    // local scope of the function test
    i = 5;
    var foo = 3;
    bar = 4;
}
test(10);
```

While `foo` and `i` are local variables inside the scope of the function `test`, the assignment of `bar` will override the global variable with the same name.

Hoisting

JavaScript **hoists** declarations. This means that both `var` statements and `function` declarations will be moved to the top of their enclosing scope.

```
bar();
var bar = function() {};
var someValue = 42;
test();
function test(data) {
    if (false) {
        goo = 1;
    } else {
        var goo = 2;
    }
    for(var i = 0; i < 100; i++) {
```

```
        var e = data[i];
    }
}
```

The above code gets transformed before any execution is started. JavaScript moves the `var` statements as well as the `function` declarations to the top of the nearest surrounding scope.

```
// var statements got moved here
var bar, someValue; // default to 'undefined'
// the function declartion got moved up too
function test(data) {
    var goo, i, e; // missing block scope moves these here
    if (false) {
        goo = 1;
    } else {
        goo = 2;
    }
    for(i = 0; i < 100; i++) {
        e = data[i];
    }
}
bar(); // fails with a TypeError since bar is still 'undefined'
someValue = 42; // assignments are not affected by hoisting
bar = function() {};
test();
```

Missing block scoping will not only move `var` statements out of loops and their bodies, it will also make the results of certain `if` constructs non-intuitive.

In the original code the `if` statement seemed to modify the *global variable* `goo`, while actually it modifies the *local variable* - after hoisting has been applied.

Without the knowledge about *hoisting*, below code might seem to raise a `ReferenceError`.

```
// check whether SomeImportantThing has been initiliazed
if (!SomeImportantThing) {
    var SomeImportantThing = {};
}
```

But of course, the above works due to the fact that the `var` statement is being moved to the top of the *global scope*.

```
var SomeImportantThing;
// other code might initiliaze SomeImportantThing here, or not
// make sure it's there
if (!SomeImportantThing) {
    SomeImportantThing = {};
}
```

Name resolution order

All scopes in JavaScript, including the *global scope*, have the special name `this` defined in them, which refers to the *current object*.

Function scopes also have the name `arguments` defined, which contains the

arguments that were passed to a function.

For example, when trying to access a variable named `foo` inside the scope of a function, JavaScript will lookup the name in the following order:

1. In case there is a `var foo` statement in the current scope use that.
2. If one of the function parameters is named `foo` use that.
3. If the function itself is called `foo` use that.
4. Go to the next outer scope and start with **#1** again.

Note: Having a parameter called `arguments` will **prevent** the creation of the default `arguments` object.

Namespaces

A common problem of having only one global namespace is the likeliness of running into problems where variable names clash. In JavaScript, this problem can easily be avoided with the help of *anonymous wrappers*.

```
(function() {
    // a self contained "namespace"
    window.foo = function() {
        // an exposed closure
    };
})(); // execute the function immediately
```

Unnamed functions are considered expressions; so in order to being callable, they must first be evaluated.

```
( // evaluate the function inside the paranthesis
function() {}
) // and return the function object
() // call the result of the evaluation
```

There are other ways for evaluating and calling the function expression; which, while different in syntax, do behave the exact same way.

```
// Two other ways
+function(){}();
(function(){}());
```

In conclusion

It is recommended to always use an *anonymous wrapper* for encapsulating code in its own namespace. This does not only protect code against name clashes, it also allows for better modularization of programs.

Additionally, the use of global variables is considered **bad practice**. **Any** use of them indicates badly written code that is prone to errors and hard to maintain.

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

Constructors

Constructors in JavaScript are yet again different from many other languages. Any function call that is preceded by the `new` keyword acts as a constructor.

Inside the constructor - the called function - the value of `this` refers to a newly created `Object`. The `prototype` of this **new** object is set to the `prototype` of the function object that was invoked as the constructor.

If the function that was called has no explicit `return` statement, then it implicitly returns the value of `this` - the new object.

```
function Foo() {
    this.bla = 1;
}
Foo.prototype.test = function() {
    console.log(this.bla);
};
var test = new Foo();
```

The above calls `Foo` as constructor and sets the `prototype` of the newly created object to `Foo.prototype`.

In case of an explicit `return` statement the function returns the value specified that statement, **but only** if the return value is an `Object`.

```
function Bar() {
    return 2;
}
new Bar(); // a new object
function Test() {
    this.value = 2;
    return {
        foo: 1
    };
}
```

```
new Test(); // the returned object
```

When the `new` keyword is omitted, the function will **not** return a new object.

```
function Foo() {
    this.bla = 1; // gets set on the global object
}
Foo(); // undefined
```

While the above example might still appear to work in some cases, due to the workings of `this` in JavaScript, it will use the *global object* as the value of `this`.

Factories

In order to be able to omit the `new` keyword, the constructor function has to explicitly return a value.

```
function Bar() {
    var value = 1;
    return {
        method: function() {
            return value;
        }
    }
}
Bar.prototype = {
    foo: function() {}
};
new Bar();
Bar();
```

Both calls to `Bar` return the exact same thing, a newly create object which has a property called `method` on it, that is a Closure.

It is also to note that the call `new Bar()` does **not** affect the prototype of the returned object. While the prototype will be set on the newly created object, `Bar` never returns that new object.

In the above example, there is no functional difference between using and not using the `new` keyword.

Creating new objects via factories

An often made recommendation is to **not** use `new` since forgetting its use may lead to bugs.

In order to create new object, one should rather use a factory and construct a new object inside of that factory.

```
function Foo() {
    var obj = {};
    obj.value = 'blub';
    var private = 2;
    obj.someMethod = function(value) {
        this.value = value;
```

```
    }
    obj.getPrivate = function() {
        return private;
    }
    return obj;
}
```

While the above is robust against a missing `new` keyword and certainly makes the use of private variables easier, it comes with some downsides.

1. It uses more memory since the created objects do **not** share the methods on a prototype.
2. In order to inherit the factory needs to copy all the methods from another object or put that object on the prototype of the new object.
3. Dropping the prototype chain just because of a left out `new` keyword somehow goes against the spirit of the language.

In conclusion

While omitting the `new` keyword might lead to bugs, it is certainly **not** a reason to drop the use of prototypes altogether. In the end it comes down to which solution is better suited for the needs of the application, it is especially important to choose a specific style of object creation **and stick** with it.

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

Equality and comparisons

JavaScript has two different ways of comparing the values of objects for equality.

The equals operator

The equals operator consists of two equal signs: ==

JavaScript features *weak typing*, that means, that the equals operator **coerces** types in order to compare them.

```
""           ==   "0"           // false
0            ==   ""            // true
0            ==   "0"           // true
false        ==   "false"       // false
false        ==   "0"           // true
false        ==   undefined     // false
false        ==   null          // false
null         ==   undefined     // true
" \t\r\n"    ==   0             // true
```

The above table shows the results of the type coercion and it is the main reason why the use of == is widely regarded as bad practice, it introduces hard to track down bugs due to its complicated conversion rules.

Additionally there is also a performance impact when type coercion is in play; for example, a string has to be converted to a number before it can be compared to another number.

The strict equals operator

The strict equals operator consists of **three** equal signs: ===

Other than the normal equals operator, the strict equals operator does **not** perform type coercion between its operands.

```
""           ===   "0"           // false
0            ===   ""            // false
0            ===   "0"           // false
false        ===   "false"       // false
false        ===   "0"           // false
false        ===   undefined     // false
false        ===   null          // false
null         ===   undefined     // false
" \t\r\n"    ===   0             // false
```

The above results are a lot clearer and allow for early breakage of code. This hardens code to a certain degree and also gives performance improvements in case the operands are of different types.

Comparing objects

While both == and === are stated as **equality** operators, they behave different

when at least one of their operands happens to be an `Object`.

```
{} === {};                   // false
new String('foo') === 'foo'; // false
new Number(10) === 10;       // false
var foo = {};
foo === foo;                 // true
```

Here both operators compare for **identity** and **not** equality; that is, they will compare for the same **instance** of the object, much like `is` in Python and a pointer comparison in C do.

In conclusion

It is highly recommended to only use the **strict equals** operator. In cases where types need to be coerced, it should be done explicitly and not left to the language's complicated coercion rules.

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

Arrays

Although arrays in JavaScript are objects, there are no good reasons to use the `for in loop` in for iteration on them. In fact there are a number of good reasons**against** the use of `for in` on arrays.

Note: JavaScript arrays are **not** *associative arrays*. JavaScript only has objects for mapping keys to values. And while associative arrays **preserve** order, objects**do**

not.

Since the `for in` loop enumerates all the properties that are on the prototype chain and the only way to exclude those properties is to use `hasOwnProperty`, it is already up to **twenty times** slower than a normal `for` loop.

Iteration

In order to achieve the best performance when iterating over arrays, it is best to use the classic `for` loop.

```
var list = [1, 2, 3, 4, 5, ..... 100000000];
for(var i = 0, l = list.length; i < l; i++) {
    console.log(list[i]);
}
```

There is one extra catch in the above example, that is the caching of the length of the array via `l = list.length`.

Although the `length` property is defined on the array itself, there is still an overhead for doing the lookup on each iteration of the loop. And while recent JavaScript engines **may** apply optimization in this case, there is no way of telling whether the code will run on one of these newer engines or not.

In fact, leaving out the caching may result in the loop being only **half as fast** as with the cached length.

The `length` property

While the *getter* of the `length` property simply returns the number of elements that are contained in the array, the *setter* can be used to **truncate** the array.

```
var foo = [1, 2, 3, 4, 5, 6];
foo.length = 3;
foo; // [1, 2, 3]
foo.length = 6;
foo; // [1, 2, 3]
```

Assigning a smaller length does truncate the array, but increasing the length does not have any effect on the array.

In conclusion

For the best performance it is recommended to always use the plain `for` loop and cache the `length` property. The use of `for in` on an array is a sign of badly written code that is prone to bugs and bad performance.

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

The `Array` constructor

Since the `Array` constructor is ambiguous in how it deals with its parameters, it is highly recommended to always use the array literals - `[]` notation - when creating new arrays.

```
[1, 2, 3]; // Result: [1, 2, 3]
new Array(1, 2, 3); // Result: [1, 2, 3]
[3]; // Result: [3]
new Array(3); // Result: []
new Array('3') // Result: ['3']
```

In cases when there is only one argument passed to the `Array` constructor, and that argument is a `Number`, the constructor will return a new *sparse* array with the`length` property set to the value of the argument. It should be noted that **only** the `length` property of the new array will be set this way, the actual indexes of the array will not be initialized.

```
var arr = new Array(3);
arr[1]; // undefined
1 in arr; // false, the index was not set
```

The behavior of being able to set the length of the array upfront only comes in handy in a few cases, like repeating a string, in which it avoids the use of a `for loop`code.

```
new Array(count + 1).join(stringToRepeat);
```

In conclusion

The use of the `Array` constructor should be avoided as much as possible. Literals are definitely preferred. They are shorter and have a clearer syntax; therefore, they also increase the readability of the code.

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

Intro	Objects	The prototype
`hasOwnProperty`	Functions	How `this` works
Closures and references	The `arguments` object	Scopes and namespaces
Constructors	Equality and comparisons	Arrays
The `Array` constructor	The `for in` loop	The `typeof` operator
The `instanceof` operator	Type casting	`undefined` and `null`
Reasons against `eval`	`setTimeout` and `setInterval`	Automatic semicolon insertion

The `for in` loop

Just like the `in` operator, the `for in` loop also traverses the prototype chain when iterating over the properties of an object.

Note: The `for in` loop will **not** iterate over any properties that have their `enumerable` attribute set to `false`; for example, the `length` property of an array.

```
// Poisoning Object.prototype
Object.prototype.bar = 1;
var foo = {moo: 2};
for(var i in foo) {
    console.log(i); // prints both bar and moo
}
```

Since it is not possible to change the behavior of the `for in` loop itself, it is necessary to filter out the unwanted properties inside the loop body , this is done by using the `hasOwnProperty` method of `Object.prototype`.

Note: Since the `for in` always traverses the complete prototype chain, it will get slower with each additional layer of inheritance added to an object.

Using `hasOwnProperty` for filtering

```
// still the foo from above
for(var i in foo) {
    if (foo.hasOwnProperty(i)) {
```

```
        console.log(i);
    }
}
```

This version is the only correct one to use. Due to the use of `hasOwnPropery` it will **only** print out `moo`. When `hasOwnProperty` is left out, the code is prone to errors in cases where the native prototypes - e.g. `Object.prototype` - have been extended.

One widely used framework which does this is Prototype. When this framework is included, `for in` loops that do not use `hasOwnProperty` are guaranteed to break.

Best practices

It is recommended to **always** use `hasOwnProperty`. Never should any assumptions be made about the environment the code is running in, or whether the native prototypes have been extended or not.

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

The `typeof` operator

The `typeof` operator (together with `instanceof`) is probably the biggest design flaw of JavaScript, as it is near of being **completely broken**.

Although `instanceof` still has its limited uses, `typeof` really has only one practical use case, which does **not** happen to be checking the type of an object.

Note: While `typeof` can also be called with a function like syntax i.e. `typeof (obj)`, this is not a function call. The two parenthesis will behave like normal and the return value will be used as the operand of the `typeof` operator. There is **no** `typeof` function.

The JavaScript type table

```
Value               Class      Type
-------------------------------------
"foo"               String     string
new String("foo")   String     object
1.2                 Number     number
new Number(1.2)     Number     object
true                Boolean    boolean
new Boolean(true)   Boolean    object
new Date()          Date       object
new Error()         Error      object
[1,2,3]             Array      object
new Array(1, 2, 3)  Array      object
new Function("")    Function   function
/abc/g              RegExp     object (function in Nitro/V8)
new RegExp("meow")  RegExp     object (function in Nitro/V8)
{}                  Object     object
new Object()        Object     object
```

In the above table *Type* refers to the value, that the `typeof` operator returns. As can be clearly seen, this value is anything but consistent.

The *Class* refers to the value of the internal `[[Class]]` property of an object.

From the Specification: The value of `[[Class]]` can be one of the following strings. `Arguments`, `Array`, `Boolean`, `Date`, `Error`, `Function`, `JSON`, `Math`, `Number`, `Object`, `RegExp`, `String`.

In order to retrieve the value of `[[Class]]` one has to make use of the `toString` method of `Object.prototype`.

The Class of an object

The specification gives exactly one way of accessing the `[[Class]]` value, with the use of `Object.prototype.toString`.

```
function is(type, obj) {
    var clas = Object.prototype.toString.call(obj).slice(8, -1);
    return obj !== undefined && obj !== null && clas === type;
}
is('String', 'test'); // true
is('String', new String('test')); // true
```

In the above example, `Object.prototype.toString` gets called with the value of this being set to the object whose `[[Class]]` value should be retrieved.

ES5 Note: For convenience the return value of `Object.prototype.`

`toString` for
both `null` and `undefined` was **changed** from `Object` to `Null` and`Undefined` in ECMAScript 5.

Testing for undefined variables

```
typeof foo !== 'undefined'
```

The above will check whether `foo` was actually declared or not; just referencing it would result in a `ReferenceError`. This is the only thing `typeof` is actually useful for.

In conclusion

In order to check the type of an object, it is highly recommended to use `Object.prototype.toString`; as this is the only reliable way of doing so. As shown in the above type table, some return values of `typeof` are not defined in the specification; thus, they can differ across various implementations.

Unless checking whether a variable is defined, `typeof` should be avoided at **all costs**.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

The `instanceof` operator #top

The `instanceof` operator compares the constructors of its two operands. It is only useful when comparing custom made objects. Used on built-in types, it is nearly as useless as the typeof operator.

Comparing custom objects

```
function Foo() {}
function Bar() {}
Bar.prototype = new Foo();
new Bar() instanceof Bar; // true
new Bar() instanceof Foo; // true
// This just sets Bar.prototype to the function object Foo
// But not to an actual instance of Foo
Bar.prototype = Foo;
new Bar() instanceof Foo; // false
```

Using `instanceof` with native types

```
new String('foo') instanceof String; // true
new String('foo') instanceof Object; // true
'foo' instanceof String; // false
'foo' instanceof Object; // false
```

One important thing to note here is, that `instanceof` does not work on objects that origin from different JavaScript contexts (e.g. different documents in a web browser), since their constructors will not be the exact same object.

In conclusion

The `instanceof` operator should **only** be used when dealing with custom made objects that origin from the same JavaScript context. Just like the `typeof` operator, every other use of it should be **avoided**.

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

Type casting

JavaScript is a *weakly typed* language, so it will apply *type coercion* **wherever** possible.

```
// These are true
new Number(10) == 10; // Number.toString() is converted
                      // back to a number
10 == '10';           // Strings gets converted to Number
10 == '+10 ';         // More string madness
10 == '010';          // And more
isNaN(null) == false; // null converts to 0
                      // which of course is not NaN
// These are false
10 == 010;
10 == '-10';
```

ES5 Note: Number literals that start with a `0` are interpreted as octal (Base 8). Octal support for these has been **removed** in ECMAScript 5 strict mode.

In order to avoid the above, use of the strict equal operator is **highly** recommended. Although this avoids a lot of common pitfalls, there are still many further issues that arise from JavaScript's weak typing system.

Constructors of built-in types

The constructors of the built in types like `Number` and `String` behave differently when being used with the `new` keyword and without it.

```
new Number(10) === 10;     // False, Object and Number
Number(10) === 10;         // True, Number and Number
new Number(10) + 0 === 10; // True, due to implicit conversion
```

Using a built-in type like `Number` as a constructor will create a new `Number` object, but leaving out the `new` keyword will make the `Number` function behave like a converter.

In addition, having literals or non-object values in there will result in even more type coercion.

The best option is to cast to one of the three possible types **explicitly**.

Casting to a string

```
'' + 10 === '10'; // true
```

By prepending a empty string a value can easily be casted to a string.

Casting to a number

```
+'10' === 10; // true
```

Using the **unary** plus operator it is possible to cast to a number.

Casting to a boolean

By using the **not** operator twice, a value can be converted a boolean.

```
!!'foo';   // true
!!'';      // false
!!'0';     // false
!!'1';     // true
!!'-1'     // true
!!{};      // true
!!true;    // true
```

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

`undefined` and `null`

JavaScript has two distinct values for `nothing`, the more useful of these two being `undefined`.

The value `undefined`

`undefined` is a type with exactly one value: `undefined`.

The language also defines a global variable that has the value of `undefined`, this variable is also called `undefined`. But this variable is **not** a constant, nor is it a keyword of the language. This means that its *value* can be easily overwritten.

ES5 Note: `undefined` in ECMAScript 5 is **no longer** *writable* in strict mode, but its name can still be shadowed by for example a function with the name`undefined`.

Some examples for when the value `undefined` is returned:

- Accessing the (unmodified) global variable `undefined`.
- Implicit returns of functions due to missing `return` statements.
- `return` statements which do not explicitly return anything.
- Lookups of non-existent properties.
- Function parameters which do not had any explicit value passed.
- Anything that has been set to the value of `undefined`.

Handling changes to the value of `undefined`

Since the global variable `undefined` only holds a copy of the actual *value* of `undefined`, assigning a new value to it does **not** change the value of the *type*`undefined`.

Still, in order to compare something against the value of `undefined` it is necessary to retrieve the value of `undefined` first.

In order to protect code against a possible overwritten `undefined` variable, a common technique used is to add an additional parameter to an anonymous wrapper, that gets no argument passed to it.

```
var undefined = 123;
(function(something, foo, undefined) {
    // undefined in the local scope does
    // now again refer to the value
})('Hello World', 42);
```

Another way to achieve the same effect would be to use a declaration inside the wrapper.

```
var undefined = 123;
(function(something, foo) {
    var undefined;
    ...
})('Hello World', 42);
```

The only difference being here, that this version results in 4 more bytes being used in case it is minified and there is no other `var` statement inside the anonymous wrapper.

Uses of `null`

While `undefined` in the context of the JavaScript language is mostly used in the sense of a traditional *null*, the actual `null` (both a literal and a type) is more or less just another data type.

It is used in some JavaScript internals (like declaring the end of the prototype chain by setting `Foo.prototype = null`), but in almost all cases it can be replaced by `undefined`.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

(Some of this is more browser specific)

Reasons against `eval`

The `eval` function will execute a string of JavaScript code in the local scope.

```
var foo = 1;
function test() {
    var foo = 2;
    eval('foo = 3');
    return foo;
}
test(); // 3
foo; // 1
```

But `eval` only executes in local scope when it is being called **directly** *and* the name of the called function is actually `eval`.

```
var foo = 1;
function test() {
    var foo = 2;
    var bar = eval;
    bar('foo = 3');
    return foo;
}
test(); // 2
foo; // 3
```

The use of `eval` should be avoided at **all costs**. 99.9% of its "uses" can be achieved **without** it.

`eval` in disguise

The timeout functions `setTimeout` and `setInterval` can both take a string as

their first argument. This string will **always** get executed in the global scope since`eval` is not being called directly in that case.

Security issues

`eval` also is a security problem as it executes **any** code given to it, it should **never** be used with strings of unknown or untrusted origins.

In conclusion

`eval` should never be used, any code that makes use of it is to be questioned in its workings, performance and security. In case something requires `eval` in order to work, its design is to be questioned and should **not** be used in the first place, a *better design* should be used, that does not require the use of `eval`.

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

This is Implementation Specific

`setTimeout` and `setInterval`

Since JavaScript is asynchronous, it is possible to schedule the execution of a function by using the `setTimeout` and `setInterval` functions.

Note: Timeouts are **not** part of the ECMAScript Standard, they are implemented as

part of the DOM.

```
function foo() {}
var id = setTimeout(foo, 1000); // returns a Number > 0
```

When `setTimeout` gets called, it will return the ID of the timeout and schedule `foo` to run in **approximately** one thousand milliseconds in the future. `foo` will then get executed exactly **once**.

Depending on the timer resolution of the JavaScript engine that is running the code, as well as the fact that JavaScript is single threaded and other code that gets executed might block the thread, it is by **no means** a safe bet that one will get the exact delay that was specified in the `setTimeout` call.

The function that was passed as the first parameter will get called by the *global object*, that means, that `this` inside the called function refers to that very object.

```
function Foo() {
    this.value = 42;
    this.method = function() {
        // this refers to the global object
        console.log(this.value); // will log undefined
    };
    setTimeout(this.method, 500);
}
new Foo();
```

Note: As `setTimeout` takes a **function object** as its first parameter, an often made mistake is to use `setTimeout(foo(), 1000)`, which will use the **return value** of the call `foo` and **not** `foo`. This is, most of the time, a silent error, since when the function returns `undefined` `setTimeout` will **not** raise any error.

Stacking calls with `setInterval`

While `setTimeout` only runs the function once, `setInterval` - as the name suggests - will execute the function **every** `X` milliseconds. But its use is discouraged.

When code that is being executed blocks the timeout call, `setInterval` will still issue more calls to the specified function. This can, especially with small intervals, result in function calls stacking up.

```
function foo(){
    // something that blocks for 1 second
}
setInterval(foo, 100);
```

In the above code `foo` will get called once and will then block for one second.

While `foo` blocks the code `setInterval` will still schedule further calls to it. Now, when `foo` has finished, there will already be **ten** further calls to it waiting for execution.

Dealing with possible blocking code

The easiest as well as most controllable solution, is to use `setTimeout` within the function itself.

```
function foo(){
    // something that blocks for 1 second
    setTimeout(foo, 100);
}
foo();
```

Not only does this encapsulate the `setTimeout` call, but it also prevents the stacking of calls and it gives additional control.`foo` itself can now decide whether it wants to run again or not.

Manually clearing timeouts

Clearing timeouts and intervals works by passing the respective ID to `clearTimeout` or `clearInterval`, depending which `set` function was used in the first place.

```
var id = setTimeout(foo, 1000);
clearTimeout(id);
```

Clearing all timeouts

As there is no built-in method for clearing all timeouts and/or intervals, it is necessary to use brute force in order to achieve this functionality.

```
// clear "all" timeouts
for(var i = 1; i < 1000; i++) {
    clearTimeout(i);
}
```

There might still be timeouts that are unaffected by this arbitrary number; therefore, is is instead recommended to keep track of all the timeout IDs, so they can be cleared specifically.

Hidden use of `eval`

`setTimeout` and `setInterval` can also take a string as their first parameter. This feature should **never** be used, since it internally makes use of `eval`.

Note: Since the timeout functions are **not** specified by the ECMAScript standard, the exact workings when a string is passed to them might differ in various JavaScript implementations. As a fact, Microsoft's JScript makes use of the `Function` constructor in place of `eval`.

```
function foo() {
    // will get called
}
```

```
function bar() {
    function foo() {
        // never gets called
    }
    setTimeout('foo()', 1000);
}
bar();
```

Since `eval` is not getting called directly in this case, the string passed to `setTimeout` will get executed in the *global scope*; thus, it will not use the local variable `foo`from the scope of `bar`.

It is further recommended to **not** use a string for passing arguments to the function that will get called by either of the timeout functions.

```
function foo(a, b, c) {}
// NEVER use this
setTimeout('foo(1,2, 3)', 1000)
// Instead use an anonymous function
setTimeout(function() {
    foo(a, b, c);
}, 1000)
```

Note: While it is also possible to use the syntax `setTimeout(foo, 1000, a, b, c)`, it is not recommended, as its use may lead to subtle errors when used withmethods.

In conclusion

Never should a string be used as the parameter of `setTimeout` or `setInterval`. It is a clear sign of **really** bad code, when arguments need to be supplied to the function that gets called. An *anonymous function* should be passed that then takes care of the actual call.

Further, the use of `setInterval` should be avoided since its scheduler is not blocked by executing JavaScript.

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

Automatic semicolon insertion

Although JavaScript has C style syntax, it does **not** enforce the use of semicolons in the source code, it is possible to omit them.

But JavaScript is not a semicolon-less language, it in fact needs the semicolons in order to understand the source code. Therefore the JavaScript parser**automatically** inserts them whenever it encounters a parse error due to a missing semicolon.

```
var foo = function() {
} // parse error, semicolon expected
test()
```

Insertion happens, and the parser tries again.

```
var foo = function() {
}; // no error, parser continues
test()
```

The automatic insertion of semicolon is considered to be one of **biggest** design flaws in the language, as it *can* change the behavior of code.

How it works

The code below has no semicolons in it, so it is up to the parser to decide where to insert them.

```
(function(window, undefined) {
    function test(options) {
        log('testing!')
        (options.list || []).forEach(function(i) {
        })
        options.value.test(
            'long string to pass here',
            'and another long string to pass'
        )
        return
        {
            foo: function() {}
        }
    }
    window.test = test
})(window)
(function(window) {
    window.someLibrary = {}
})(window)
```

Below is the result of the parser's "guessing" game.

```
(function(window, undefined) {
    function test(options) {
        // Not inserted, lines got merged
        log('testing!')(options.list || []).forEach(function(i) {
        }); // <- inserted
        options.value.test(
            'long string to pass here',
            'and another long string to pass'
        ); // <- inserted
        return; // <- inserted, breaks the return statement
        { // treated as a block
            // a label and a single expression statement
            foo: function() {}
        }; // <- inserted
    }
    window.test = test; // <- inserted
// The lines got merged again
})(window)(function(window) {
    window.someLibrary = {}; // <- inserted
})(window); //<- inserted
```

Note: The JavaScript parser does not "correctly" handle return statements which are followed by a new line, while this is not neccessarily the fault of the automatic semicolon insertion, it can still be an unwanted side-effect.

The parser drastically changed the behavior of the code above, in certain cases it does the **wrong thing**.

Leading parenthesis

In case of a leading parenthesis, the parser will **not** insert a semicolon.

```
log('testing!')
(options.list || []).forEach(function(i) {})
```

This code gets transformed into one line.

```
log('testing!')(options.list || []).forEach(function(i) {})
```

Chances are **very** high that `log` does **not** return a function; therefore, the above will yield a `TypeError` stating that `undefined is not a function`.

In conclusion

It is highly recommended to **never** omit semicolons, it is also advocated to keep braces on the same line with their corresponding statements and to never omit them for one single-line `if` / `else` statements. Both of these measures will not only improve the consistency of the code, they will also prevent the JavaScript parser from changing its behavior.

Created with the Personal Edition of HelpNDoc: Easily create Web Help sites

Standard global objects (alphabetically)

Created with the Personal Edition of HelpNDoc: Free help authoring tool

Infinity

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

isFinite()

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

undefined

You cannot declare a constant with the same name as a function or variable in the same scope. For example:

```
view plainprint?
1.  // THIS WILL CAUSE AN ERROR
2.  function f() {};
3.  const f = 5;
4.
5.  // THIS WILL CAUSE AN ERROR ALSO
6.  function f() {
7.    const g = 5;
8.    var g;
9.
10.   //statements
11. }
```

Literals

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Additional Js/EMCA Scripting Information

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

Js/EMCA Best Practice and Tips

Code Conventions for the JavaScript Programming Language

with Statement Considered Harmful

"If you can't read a program and be confident that you know what it is going to do, you can't have confidence that it is going to work correctly. For this reason, the with statement should be avoided."

http://yuiblog.com/blog/2006/04/11/with-statement-considered-harmful/

Global variables are evil.

"Global variables are a source of unreliability and insecurity. Fortunately, JavaScript includes tools for allowing us to drastically minimize our use of globals, which makes our programs more robust. This becomes increasingly important as our programs get bigger, and as we mix in and mash up program components from multiple authors

http://yuiblog.com/blog/2006/06/01/global-domination/

Seven JavaScript Things I Wish I Knew Much Earlier In My Career

"I've been writing JavaScript code for much longer than I care to remember. I am very excited about the language's recent success; it's good to be a part of that success story. I've written dozens of articles, book chapters and one full book on the matter, and yet I keep finding new things. Here are some of the "aha!" moments I've had in the past, which you can try out rather than waiting for them to come to you by chance.
http://www.smashingmagazine.com/2010/04/20/seven-javascript-things-i-wish-i-knew-much-earlier-in-my-career/

Five things to do to a script before handing it over to the next developer

"Let's face fact folks: not too many developers plan their JavaScripts. Instead we quickly write something that works, and submit it. We come up with variable and function names as we go along and in the end know that we'll never have to see this little bit of script ever again.

"The problems start when we do see our script again, or we get scripts from other developers, that were built the same way. That's why it is good to keep a few extra steps in mind when it comes to saying "this is done, I can go on"."

http://www.wait-till-i.com/2008/02/07/five-things-to-do-to-a-script-before-handing-it-over-to-the-next-developer/

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

JavaScript Objects

JavaScript Objects

There are many useful features in JavaScript (QtScript) which are worth scratching just below the surface to find.

This first link is to a good summary of many aspects. It is a framework build up, step by step, for browser bound JavaScript, but it is done in a very useful introductory fashion showing concepts universal to JavaScript implementation any where.

Let's Make a Framework: Free eBook

By Alex Young | 02 Dec 2010

"I've collected and edited the Let's Make a Framework articles into a book that suitable for e-readers.
Consider this a Christmas present!"

- build-a-javascript-framework.pdf
- build-a-javascript-framework.epub
- build-a-javascript-framework.mobi (Kindle)

The other following links are very good explanations of the features they focus on.

Learning Javascript with Object Graphs

By Tim Caswell

One of the secrets to being a super effective JavaScript developer is to truly understand the semantics of the language. This article will explain the basic elemental parts of JavaScript using easy to follow diagrams.

http://howtonode.org/object-graphs

Learning Javascript with Object Graphs (Part II)

By Tim Caswell

The first article using graphs to describe JavaScript semantics was so popular that I've decided to try the technique with some more advanced ideas. In this article I'll explain three common techniques for creating objects. They are constructor with prototype, pure prototypal, and object factory.

My goal is that this will help people understand the strengths and weaknesses of each technique and understand what's really going on.

http://howtonode.org/object-graphs-2

Learning Javascript with Object Graphs (Part III)

By Tim Caswell

Part I of this series explained basic object graphs and visually described references, closures, and basic inheritance in JavaScript. Part II compared different styles for doing object-oriented programming in JavaScript. Now in Part III we'll get creative and look as Ruby's object model and compare it to how JavaScript works. Also I'll show how to implement some Ruby style classes. JavaScript is a very flexible language and can support about any object model you want with enough understanding and creativity.

http://howtonode.org/object-graphs-3

Creating Objects

By Mark "Tarquin" Wilton-Jones

Any function in JavaScript can be used to create custom object classes, simply by calling it using the keyword `new`. When called in this way, the special variable `this` inside the function references the new object that is being constructed (it normally refers to the 'current' object, which is usually `window`, except inside methods). The function should not return a value.

http://www.howtocreate.co.uk/tutorials/javascript/objects

My Favorite JavaScript Design Pattern

James Edwards

"I thought it might be interesting to look at a JavaScript design pattern that I use a great deal. I settled on it gradually, over a period of time, absorbing and adapting influences from various sources, until reaching a pattern that offers the flexibility I need.

Let me show you an overview, and then look at how it comes together ..."

http://blogs.sitepoint.com/2010/11/30/my-favorite-javascript-design-pattern/

JavaScript Module Pattern: In-Depth

Ben Cherry

"The module pattern is a common JavaScript coding pattern. It's generally well understood, but there are a number of advanced uses that have not gotten a lot of attention. In this article, I'll review the basics and cover some truly remarkable advanced topics, including one which I think is original.

We'll start out with a simple overview of the module pattern, which has been well-known since Eric Miraglia (of YUI) firstblogged about it three years ago. If you're already familiar with the module pattern, feel free to skip ahead to 'Advanced Patterns' ..."

http://www.adequatelygood.com/2010/3/JavaScript-Module-Pattern-In-Depth

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Learning Javascript with Object Graphs (Part I)

Learning Javascript with Object Graphs

One of the secrets to being a super effective JavaScript developer is to truly understand the semantics of the language. This article will explain the basic elemental parts of JavaScript using easy to follow diagrams.

References Everywhere

A variable in JavaScript is simply a label that references a value in memory somewhere. These values can be primitives like strings, numbers, and booleans. They can also be objects or functions.

Local Variables

In the following example, we will create four local variables in the top-level scope and point them to some primitive values:

variables.js

```
// Let's create some local variables in the top scope
var name = "Tim Caswell";
```

```
var age = 28;
var isProgrammer = true;
var likesJavaScript = true;
// Test to see if the two variables reference the same value
isProgrammer === likesJavaScript;
```

=> true

Output

Notice that the two boolean variables point to the same value in memory. This is because primitives are immutable and so the VM can optimize and share a single instance for all references to that particular value.

In the code snippet we checked to see if the two references pointed to the same value using `===` and the result was `true`.

The outer box represents the outermost closure scope. These variables are top-level local variables, not to be confused with properties of the global/window object.

Objects and Prototype Chains

Objects are just collections of more references to new objects and prototypes. The only special thing they add is the prototype chain for when you try to access a property that's not in the local object, but is in a parent object.

objects.js

```
// Create a parent object
var tim = {
  name: "Tim Caswell",
  age: 28,
  isProgrammer: true,
  likesJavaScript: true
}
// Create a child object
var jack = Object.create(tim);
// Override some properties locally
jack.name = "Jack Caswell";
jack.age = 4;
// Look up stuff through the prototype chain
jack.likesJavaScript;
```

=> true

Output

Here we have one object with four properties referenced by the `tim` variable. Also we created a new object that inherits from the first object and referenced it from `jack`. Then we overrode two properties in the local object.

Now when looking up `jack.likesJavaScript`, we first find the object that `jack` references. Then we

look for the`likesJavaScript` property. Since it's not there, we look at the parent object and find it there. Then we find the `true` value it references.

The Global Object

Ever wondered why tools like **jslint** always tell you to not forget to put `var` statements before your variables. Well, here is what happens if you forget.

globals.js

```
var name = "Tim Caswell";
var age = 28;
var isProgrammer = true;
// Oops we forgot a var
likesJavaScript = true;
```

 Notice that `likesJavaScript` is now a property of the global object instead of a free variable in the outer closure. This only really matters if you're going to be mixing several scripts. But in any real program that's exactly what you're going to be doing.

Always remember to put those `var` statements in there to keep your variable's scope to the current closure and it's children. You'll be much happier by following this simple rule.

If you must put something on the global object, do it explicitly with `window.woo` in the browser or`global.goo` in node.js.

Functions and Closures

JavaScript isn't just a series of chained data structures. It contains executable, callable code known as functions. These functions create chained scopes and closures.

Visualizing Closures

Functions can be drawn as special objects that contain executable code as well as properties. Every function has a special`[scope]` property that represents the environment it was in when it was defined. If a function is returned from another function then this reference to the old environment is closed over by the new function in a "closure".

In this example we will create a simple factory method that generates a closure and returns a function.

closure.js

```
function makeClosure(name) {
  return function () {
    return name;
  };
}
```

```
var description1 = makeClosure("Cloe the Closure");
var description2 = makeClosure("Albert the Awesome");
console.log(description1());
console.log(description2());
```

Output

```
Cloe the Closure Albert the Awesome
```

When we call `description1()`, the VM looks up the function that it references and executes it. Since that function looks for a local variable named `name`, it finds it in the closure scope. This factory method is nice since each generated function has it's own space for local variables.

See the article **why use closure** for more in-depth reading on this topic and it's many uses.

Shared Functions and `this`

Sometimes for performance reasons, or because you just plain prefer the style, JavaScript provides a `this` keyword that allows you to reuse a function object in different scopes depending on how it was called.

Here we'll create a few objects that all share a common function. This function will reference `this` internally to show how it changes from call to call.

functions.js

```
var Lane = {
  name: "Lane the Lambda",
  description: function () {
    return this.name;
  }
};
var description = Lane.description;
var Fred = {
  description: Lane.description,
  name: "Fred the Functor"
};
// Call the function from four different scopes
console.log(Lane.description());
console.log(Fred.description());
console.log(description());
console.log(description.call({
  name: "Zed the Zetabyte"
}));
```

Output

```
Lane the Lambda Fred the Functor undefined Zed the Zetabyte
```

In the diagram, we see that even though `Fred.description` was set to `Lane.description`, it's really only referencing the function. Thus all three references have equal ownership of the anonymous function. This is why I try to not call functions on constructor prototypes "methods", because that implies some sort of binding of the function to the constructor and it's "class". *(see **what is***

this *for more details on the dynamic nature of* `this`*)*

Conclusion

I've had tons of fun using diagrams to visualize these data structures. My hope is that this helps those of us that are visual learners to get a better grasp of JavaScript semantics. I have past experience as both a front-end designer/developer and as a server-side architect. I hope my unique perspective is useful to those coming from the world of design and learning the innards of this wonderful language known as JavaScript.

(NOTE, all the diagrams are ***graphviz*** *dot files and can be seen* ***here****)*

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

Extending Classes In Object Oriented Javascript

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

User 'Library' Modules and helper objects

Brief Instructions ...

All these files go in one directory under configuration/scripts/*modualibraries*
... any directory there will do, it doesn't have to be called *moduaibraries*
(See notes below for descriptions)

phpjs.namespaced.mod
helper_PhpJs.mod
helper_twPan.mod

mod_general.js
mod_reload.js

There is a zip of the latest versions available at:
Coming soon!

The basic set up will happen on TeXworks startup. The necessary files are pre loaded i to global objects. This is done as otherwise if read access has not been turned on by the User, subsequent scripts would not be able to load the libraries.

During a Tw session, if you alter and need to re- initialise, from your script sub-menu run the mod_reload.js script (Menu item: **Re-load Globals**) to load the altered libraries (the current short cut for doing that would be **Alt+R Alt+L** if you have not already used that combination for something else).

For any script that you wish to use the constants and object functions found in twPan, set it up in this way,

```
eval(TW.app.getGlobal("helper_twPan"));
```

This will create twConst and msgBox, and twPan with all of its members for use.

Then Try

```
twPan.alert('Hi');

var response = twPan.confirm("Do You Want That");
    twPan.alert("User Happy: " + response);

var chosen = twPan.select(Please Choose One", ["One","Two","Three"]);
    twPan.alert("User Chose: " + chosen);
```

And in any script you (also) wish to use the functionality of PhpJs (scores of predefined functions) set it up in this way,

```
eval(TW.app.getGlobal("helper_PhpJs"));
```

This will create PhpJs and all its members for use. See PhpJs Function List

Either helper object () can be used with, or with out the other. They are

independent of each other.

Notes:
phpjs.namespaced.mod (exposes JavaScript versions of numerous php array, string handling, and other functions)

Libarary Files:

phpjs.namespaced.mod (object source)
helper_PhpJs.mod (loader)

helper_twPan.mod (object source and loader)

Hook Script:
mod_general.js (establishes .mod files on Tw startup as application wide globals available to any script)

Stand Alone Script:
mod_reload.js (re-establishes edited .mod files as application wide globals available to any script)

At Application startup the necessary modules will be pre-loaded automatically.

Whenever during your current Tw session you change any of phpjs.namespaced.mod, helper_twPan.mod, helper_PhpJs.mod or any you have added to mod_general.js or mod_reload.js, run the mod_reload.js script to reload the altered libraries (the current short cut for doing that would be Alt+R Alt+L if you have not already used that combination).

Unless you are introducing new or improved functionality to phpjs.namespaced.mod you will probably be leaving it alone.

helper_PhpJs.mod is used to load phpjs.namespaced.mod and create the necessary helper object PhpJs.

helper_twPan.mod has everything it needs in the one file, and when evaluated it creates the twPan helper object and various constant objects, and functions.

twConst

These will be added to as releveant constants grow in the Tw api. Preently the following are incorporated and relate to the C++ code.

```
SystemAccess_OK : 0,
SystemAccess_Failed : 1,
SystemAccess_PermissionDenied : 2
```

msgBox

twPan

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

helper_twPan.mod

```
/*This file is read into a Tw global at startup */

// original phpjs.namespaced.mod from http://phpjs.org/ * Dual licensed under the MIT (MIT-LICENSE.txt) * and GPL (GPL-LICENSE.txt) licenses.

var window = {};  // fix for not being in a browser environment

 eval(TW.app.getGlobal("phpjs_namespaced"));


  var   PhpJs = new PHP_JS();
  var      $P = PhpJs;

// TW.information(null, 'test', 'Setting up');

  const twConst = {
          SystemAccess_NotAttempted : -1,
                SystemAccess_OK : 0,
             SystemAccess_Failed : 1,
      SystemAccess_PermissionDenied : 2

            }; // end twConst


 /* from http://doc.trolltech.com/4.7/qmessagebox.html#StandardButton-enum and discussions on thread http://tug.org/mailman/htdig/texworks/2010q2/002662.html */

 const msgBox = {

 Ok              : 0x00000400, // An "OK" button defined with the AcceptRole.
 Open            : 0x00002000, // A "Open" button defined with the AcceptRole.
 Save            : 0x00000800, // A "Save" button defined with the AcceptRole.
 Cancel          : 0x00400000, // A "Cancel" button defined with the RejectRole.
 Close           : 0x00200000, // A "Close" button defined with the RejectRole.
 Discard         : 0x00800000, // A "Discard" or "Don't Save" button, depending on the platform, defined with the DestructiveRole.
 Apply           : 0x02000000, // An "Apply" button defined with the ApplyRole.
 Reset           : 0x04000000, // A "Reset" button defined with the ResetRole.
 RestoreDefaults  : 0x08000000, // A "Restore Defaults" button defined with the ResetRole.
 Help            : 0x01000000, // A "Help" button defined with the HelpRole.
 SaveAll         : 0x00001000, // A "Save All" button defined with the AcceptRole.
```

```
Yes             : 0x00004000, // A "Yes" button defined with the YesRole.
YesToAll        : 0x00008000, // A "Yes to All" button defined with the YesRole.
No              : 0x00010000, // A "No" button defined with the NoRole.
NoToAll         : 0x00020000, // A "No to All" button defined with the NoRole.
Abort           : 0x00040000, // An "Abort" button defined with the RejectRole.
Retry           : 0x00080000, // A "Retry" button defined with the AcceptRole.
Ignore          : 0x00100000, // An "Ignore" button defined with the AcceptRole.
NoButton        : 0x00000000  // An invalid button.

  };


/* A function tag window entry of this form
  //%:    ##file_get_contents## : string;

  Will use the function in Alt+R, Alt+F  - 'Run twPan Stand Alone Functions'
  under moduleAndObjects menu

*/

var  twPan = {

//%: callingScriptPath() : string ; // Path = dirctory with trailing / of calling script

        callingScriptPath : function (scriptFile)  // pass __FILE__ in here
                        {

                    return  scriptFile.substr(0,scriptFile.lastIndexOf("/") +1);
                        },


//%: Biblio     - citationType : array of string ; // Layout of citation format - keep in sync with:  insertCite

 citationType : ["[1. page 2-3]", "Goossens et al. (1993)","(Goossens et al., 1993)","Goossens,
Mittlebach, and Samarin (1993)","(Goossens, Mittlebach, and Samarin, 1993)", "Jones et al.","Jones,
Baker, and Williams","1990","(1990)"],

//%:     Biblio - insertCite : array of string ; // Command for  citation format - keep in sync with:
citationType
 insertCite : [ "cite", "citet", "citep", "citet*", "citep*", "citeauthor", "citeauthor*", "citeyear",
"citeyearpar"],


//%:     Biblio - citationOrder : array of string ; // for MySql query dropbox  - keep in sync with
citeOrderBy
 citationOrder : ["By Citation Key", "By Title", "By Author - (on first Name)", "Year", "Journal",
"Editor"],

//%:     Biblio - citeOrderBy : array of string ; // for MySql query  extract dropbox - keep in sync with
citationOrder
```

```
 citeOrderBy : ["cite_key", "title", "author", "year", "journal", "editor"],

//%: bibleNamesAbrvs : array of string ; // Names and Abbreviations of current Bibles
            bibleNamesAbrvs : function()
            {

             var bibleList = this.osCmd('cmd /c getBibleNamesAbrs.php.bat', true);

               return bibleList.split('\n');

             },

              /* See panMagUseAa.sty

                 %:                Count Verses - now loaded directly in .sty

                 \global\def\versionList{NIV,WEB,NET,ISV,AMP,RSV,KJV,NKJV,UNSP}%

                 And

                 %:               \showBibleVerseCount

             */

              os : "", // prefer to do these in initialise()
//%: initialise : void;

          initialise : function() {
                            this.os = TW.platform() ;
                           // this.alert("Tw loaded");

                 },  // End. pan_Tw.initialise();

//%: Web Like Equivalents

//%:     select(message, optionsArray) : string

           select : function(message, optionsArray){

         chosen = TW.getItem(null, message, message, optionsArray, 0, true); // true - editable

            if (chosen === null)
              {return "";}

                 return  chosen;

                                                                                  }, // End.  twPan.select
()

 //%:     chooseFromDirectoryListing(Description, pathDir, fileExtension) : string
```

```
   chooseFromDirectoryListing : function(Description, pathDir, fileExtension)
   {
    // pathDir (no trailing slash), fileExtension with . E.g.  .txt

   /*   TW.app.clipboard = "cmd /c dir /b  \""
                    +  pathDir + "\\*" + fileExtension + "\"";

                    */


    var fileList = this.osCmd("cmd /c dir /b  \""
                     +  pathDir + "\\*" + fileExtension + "\""
                       , true)


       removeExtnsn = new RegExp(fileExtension,"ig");

      fileList = fileList.replace(removeExtnsn,"").split(this.LineBreakIs(fileList));

    var chosenAnswer =
           this.select(Description, fileList);

      if (this.emptyString(chosenAnswer) == false)
        {
        chosenAnswer = chosenAnswer + fileExtension;
         }

    return chosenAnswer;
   },  // End. twPan.chooseFromDirectoryListing



//%:    prompt(message) : string
         prompt : function(message){

                        response = TW.getText(null, message, message);

                           if( response == null)
                             {response = -1};

                           return response;
                           },  // End. pan_Tw.input(message)

//%:    confirm(message) : boolean
         confirm : function(message){

                        response = TW.question(null, message, message, msgBox.Yes | msgBox.No,
msgBox.Yes);
                        if (response == msgBox.Yes)
```

```
                    {return true;}
                   else
                    {return false;}

                        },  // End. pan_Tw.confirm(message)
//%:    alert(text) : void;

        alert : function(text){
                    TW.information(null, "TeXworks Message", text);

                    }, // End. pan_Tw.alert();

//%:    ##file_get_contents## : string;
file_get_contents : function(){


          if (this.file_get_contents.arguments.length == 0)
              {
              fileNameFullPath = TW.app.getOpenFileName();
              }
            else
              {
               fileNameFullPath = this.file_get_contents.arguments[0];
               }

                  var res = TW.readFile(fileNameFullPath);
                    switch (res.status)
                    {
                      case twConst.SystemAccess_PermissionDenied:
                      case twConst.SystemAccess_Failed:
                                                                                this.
alert(res.message);
                                                                                var text
= undefined;
                                                                          break;

                       case twConst.SystemAccess_OK:

                                                                                var text
= res.result;
                                                                          break;
                       }

                    return text;

                        }, // End. twPan.file_get_contents

//%:    file_put_contents : void;

      file_put_contents : function(fileNameFullPath , fileContents){
```

```
                        var  status = TW.writeFile(fileNameFullPath, fileContents); // pathFileName,
Contents (- utf-8)
                        switch (status)
                        {
                         case twConst.SystemAccess_PermissionDenied:
                         case twConst.SystemAccess_Failed:
                                                                this.alert('Please Turn on File
Writing in \n\ Edit Menu / Preferences / Scripts');
                                                                    break;

                        } // End. switch (status)



                              }, // End. twPan.file_put_contents

//%: getRelativePath(mainPath, wantedPath)
       getRelativePath : function(mainPath, wantedPath){

            // require - assume full file paths including filenames

         var mainPathLessFile = mainPath.substr(0, mainPath.lastIndexOf('/')+1);
      var wantedPathLessFile = wantedPath.substr(0, wantedPath.lastIndexOf('/')+1);

         var mainPathJustFile = mainPath.substr(mainPath.lastIndexOf('/')+1);
       var wantedPathJustFile = wantedPath.substr(wantedPath.lastIndexOf('/')+1);


           if (wantedPathLessFile == mainPathLessFile) // TeX file and graphic in same directory
                {return wantedPathJustFile;}

 if (wantedPathLessFile.indexOf(mainPathLessFile) > -1) // graphic (2nd) file in lower directory
           {return wantedPathLessFile.replace(mainPathLessFile, '')+wantedPathJustFile;}

 if (mainPathLessFile.indexOf(wantedPathLessFile) > -1) // graphic (2nd) file in higher directory
                    {
                     remainderPath =   mainPathLessFile.replace(wantedPathLessFile, '')

                      var buildSlash = '';

                       for (chr in remainderPath)
                         {
                           if (remainderPath[chr] == '/')
                             {buildSlash += '../'; }
                          }

                        return buildSlash + wantedPathJustFile;
                     }
                    else  // higher then lower
                    {
```

```
        mainPathLessFileArray =    mainPathLessFile.split('/');
        wantedPathLessFileArray =  wantedPathLessFile.split('/');

      if (mainPathLessFileArray[0] != wantedPathLessFileArray[0]) // different file tree(?)
          { return wantedPath;  // full path of graphic (2nd) file
                    }

         for (var count = 0; count < mainPathLessFileArray.length; count++ )
           {
          if (mainPathLessFileArray[count] != wantedPathLessFileArray[count])
                          {
                           break;
                           }
                        }
          neededSlashes =  mainPathLessFileArray.length - (count + 1);

                          var buildSlash = '';

                          for (J = 0; J < neededSlashes; J++)
                              {
                              buildSlash += '../';
                               }

                          var buildPath ='';

           for (K = count; K < (wantedPathLessFileArray.length -1); K++)
                                                          // -1 remove trailing slash
                    {
                       buildPath += wantedPathLessFileArray[K] + '/';
                                }
                  return  buildSlash +  buildPath +  wantedPathJustFile;
                }



                }, // End. twPan.getRelativePath(mainPath, wantedPath)


//%: finalise : void;


      finalise : function(){
                  // nothing to finalise at the moment
                  },// End. twPan.finalise

//%: Bible Related

//%:    getBibleVersion : object : (abbrv : string | full : string) ; if a string is passed it is a specific for the
occasion  from User
```

```
        getBibleVersion : function(){

         if (this.getBibleVersion.arguments.length > 0) // User has supplied text
             {
               var userText = this.getBibleVersion.arguments[0]; // get User's default text
             choiceIndex = 0;} // use User's text at top
             else
              { // User supplied no text, use standard list with gap at top for User to write in
               var userText = '';
               choiceIndex = 3;  // NET, most frequently used at present
               }

//  this.bibleNamesAbrvs at top of twPan

            var bibleList = this.bibleNamesAbrvs();

            var getNamesAbrvs = [userText].concat(bibleList);

            userChoice  = TW.getItem( null,  "Bible Version ?", "Bible Version : ",
                                  getNamesAbrvs ,  choiceIndex , true ) ;

      var result = [];

       if( userChoice == undefined)
          {
         result.abbrv= '';
         result.full= '';
           }
        else
        {
         var abrvStart = userChoice.lastIndexOf(' ');

          result.abbrv = userChoice.substr(abrvStart + 1);
          result.full  = userChoice.substr(0, abrvStart );
         }

                      return result;
                        },// End. twPan.getBibleVersion

//%:     getBibleVersionName : string // no abrev at end as getBibleVersionFull does
//##getBibleVersionName##  - available for use in runtwPanStandaloneFunctions.js

        getBibleVersionName : function(){

          return this.getBibleVersion().full;

                            }, // End.  getBibleVersionName()


//%:     getBibleVersionAbrv : string // no Name at start as getBibleVersionFull does
```

```
//##getBibleVersionAbrv##  - available for use in runtwPanStandaloneFunctions.js

        getBibleVersionAbrv : function(){

          return this.getBibleVersion().abbrv;

                        }, // End.  getBibleVersionName()


//%:     ##getBibleVersionFull## : object : (abbrv : string | full : string) ; if a string is passed it is a specific
for the occasion  from User

        getBibleVersionFull : function(){

         if (this.getBibleVersionFull.arguments.length > 0) // User has supplied text
             {
              var userText = this.getBibleVersionFull.arguments[0]; // get User's default text
            choiceIndex = 0;} // use User's text at top
            else
             { // User supplied no text, use standard list with gap at top for User to write in
              var userText = '';
              choiceIndex = 3;  // NET, most frequently used at present
              }

//  this.bibleNamesAbrvs at top of twPan

            var bibleList = this.bibleNamesAbrvs();

            var getNamesAbrvs = [userText].concat(bibleList);

            userChoice  = TW.getItem( null,  "Bible Version ?", "Bible Version : ",
                              getNamesAbrvs ,  choiceIndex , true ) ;

                    return userChoice;

                      }, // End. twPan.getBibleVersionFull

//%: Date Related


//%:     ##getSuffixDateTh## : string : (dateNumerals) : string ;

        getSuffixDateTh : function(dateNumerals){

        var suffices = ['st','nd','rd'];

                            if (dateNumerals.length > 1)
                              {
                               lastDigit = dateNumerals.substr(1);
                                }
```

```
                    else
                     {
                     lastDigit = dateNumerals.substr(0);
                       }



                    if (lastDigit < 1 || lastDigit > 3 || (dateNumerals > 9 & dateNumerals < 21))
                     {
                       return  'th';
                     }
                     else
                     {
                       return suffices[lastDigit -1];
                       }

                 },  // End. twPan.getSuffixDateTh(dateNumerals)


//%:    ##insertSuperscriptDateTh## : string : (dateNumerals) : string ;

        insertSuperscriptDateTh : function(dateNumerals){

            return   dateNumerals
                    + "\\superscript{"
                    + this.getSuffixDateTh(dateNumerals)
                    +"}";

                    },  // End. twPan.insertSuperscriptDateTh(dateNumerals)

//%:    ##getDateWords## : () ; // can accept an argument for date format 'dmy' or 'ydm' or 'ymd', if not present dropdown box is shown.


         getDateWords : function(){

    var months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'Novemeber', 'December'];

    var days = ['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday'];



                    // Altered from: http://www.tizag.com/javascriptT/javascriptdate.php

        var currentTime = new Date();

        var month = months[currentTime.getMonth()];
                var day = days[currentTime.getDay()];
```

```
                var dayNum = currentTime.getDate() + '';
                var year = currentTime.getFullYear();


        dayNumTh = dayNum + this.getSuffixDateTh(dayNum);

    if (this.getDateWords.arguments.length > 0)
      {

       wantedFormat = this.getDateWords.arguments[0].toLowerCase();

     switch (wantedFormat)
        {
        case 'dmy': // could just use default below HERE FOR COMPLETENESS
           result = day + ' ' + dayNumTh + ', ' + month + ', '   + year;
         break;

        case 'ydm':
           result = year + ', ' + day +  ' ' + dayNumTh +', ' + month;
         break;

        case 'ymd':
           result = year + ', ' + month + ', ' + day + ' ' + dayNumTh ;
         break;

        default:
           result = day + ' ' + dayNumTh + ', ' + month + ', '   + year;
        }

      }
     else
     {
       var result = this.select('Select Format',
                          [ day + ' ' + dayNumTh + ', ' + month + ', '   + year,
                           year + ', ' + day +  ' ' + dayNumTh + ', ' + month,
                           year + ', ' + month + ', ' + day + ' ' + dayNumTh
                           ]);
     }

      return result;

                    }, // End twPan.getDateWords


//%:     ##getDate## : () ; // can accept an argument for date format 'dmy' or 'ydm' or 'ymd', if not present dropdown box is shown.

        getDate : function(){
```

```
                    // Altered from: http://www.tizag.com/javascriptT/javascriptdate.php
      var currentTime = new Date();
      var month = currentTime.getMonth() + 1;
                var day = currentTime.getDate() + ''; // force cast for length test
                var year = currentTime.getFullYear();
                var hour = currentTime.getHours()+ '';
   var minute = currentTime.getMinutes()+ '';
   var second = currentTime.getSeconds()+ '';


                if (day.length < 2){day = "0" + day;}
                   month = month + '';  // force cast for length test
                if (month.length < 2){month = "0" + month;}
   if (hour.length < 2){hour = "0" + hour;}
   if (minute.length < 2){minute = "0" + minute;}
   if (second.length < 2){second = "0" + second;}


if (this.getDate.arguments.length > 0)
  {

    wantedFormat = this.getDate.arguments[0].toLowerCase();

    switch (wantedFormat)
     {
     case 'dmy': // could just use default below HERE FOR COMPLETENESS
       result = day + '-' + month + '-'   + year;
      break;

     case 'ydm':
       result = year + '-' + day + '-' + month;
      break;

     case 'ymd':
       result = year + '-' + month + '-' + day;
      break;

     case 'ymdhms':
       result = year + '-' + month + '-' + day
             + '--' + hour + '-' + minute + "-" + second;
      break;

     default:
       result = day + '-' + month + '-'  + year;
     }

  }
 else
 {
   var result = this.select('Select Format',
```

```
                        [ day + '-' + month + '-'   + year,
                         year + '-' + day + '-' + month,
                         year + '-' + month + '-'    + day
                         ]);
    }

      return result;

                    }, // End twPan.getDate

//%:  String Handling

//%:      trim ltrim rtrim
// http://www.somacon.com/p355.php  Created 2006-10-19, Last Modified 2007-06-07, © Shailesh N. Humbad Disclaimer: This content is provided as-is. The information may be incorrect.


 trim : function(stringToTrim)
       {
       if(stringToTrim === false)
        {return stringToTrim;}

            return stringToTrim.replace(/^\s+|\s+$/g,"");
        },

 ltrim : function(stringToTrim)
       {

      f(stringToTrim === false)
        {return stringToTrim;}

          return stringToTrim.replace(/^\s+/,"");
        },

 rtrim : function(stringToTrim)
      {

       if(stringToTrim === false)
        {return stringToTrim;}

          return stringToTrim.replace(/\s+$/,"");
      },


//%:    ##convertSpeechMarks##(text) : string ;  // gets around most fun and games when pasting in form other applications

          convertSpeechMarks : function(text){

  aposSingle = new RegExp("\u2018"|"\u2019","g");
```

```
    textChanged = text.replace(aposSingle,"'");
  //aposSingle = new RegExp("\u2019","g");
  //   textChanged = textChanged.replace(aposSingle,"'");

  textChanged = textChanged.replace(/\s\'/g,"'`");

   textChanged = textChanged.replace(/\n\'/g,"\n`");
  textChanged = textChanged.replace(/?/g,"'`");
  textChanged = textChanged.replace(/?/g,"'");
  textChanged = textChanged.replace(/' /g,"'");
  textChanged = textChanged.replace(/\s\?/g," ``");
  textChanged = textChanged.replace(/\n\?/g,"\n``");
  textChanged = textChanged.replace(/\s\"/g," ``");
  textChanged = textChanged.replace(/^\"/g," ``");
  textChanged = textChanged.replace(/^\‘/g," `");
  textChanged = textChanged.replace(/\n\"/g,"\n``");
  textChanged = textChanged.replace(/\n\“/g,"\n``");
  textChanged = textChanged.replace(/?/g,"\'");
  textChanged = textChanged.replace(/?/g,"\'\'");
  textChanged = textChanged.replace(/\‘$/g,"\'")
  textChanged = textChanged.replace(/\"$/g,"\'\'")
  textChanged = textChanged.replace(/\“$/g,"\'\'")
  textChanged = textChanged.replace(/?/g,"\'\'");

  openDouble = new RegExp("\u201C","g");
  closeDouble = new RegExp("\u201D","g");
  textChanged = textChanged.replace(closeDouble,"\'\'");
  textChanged = textChanged.replace(openDouble,"'`");

 return textChanged;

          }, // End.  twPan.convertSpeechMarks(text)
                                                                                    //%:    LineBreakIs
(text) : string ;

          LineBreakIs : function(text){

            //   http://www.codeguru.com/forum/showthread.php?t=253826

    if (text.indexOf("\r\n") > -1)
      {
     var  sep = "\r\n"; // Windows
      }
     else if (text.indexOf("\n") > -1)
      {
     var  sep = "\n"; // *nix, Mac OSX
       }
    else
      {
     var   sep = "\r"; // classical Mac
```

```
        }

        return sep;

            }, // End. twPan.LineBreakIs
                                                                                    //%:
lineNumber : integer ;


         lineNumber : function(){

        var lineSeperator = this.LineBreakIs(TW.target.text);


        var lines = TW.target.text.split(this.LineBreakIs(TW.target.text));

        var charCount = 0;

        var weAreHere = TW.target.selectionStart;

       for (line in lines)
        {

          charCount += (lines[line].length *1)
                    + lineSeperator.length; // for loss of separator on .split

          if (weAreHere < charCount)
            {
             return (line * 1) +1; // array zero offset in lines[]

            }

        }

            }, // End. twPan.lineNumber
                                                                                //%:
##cleanParaMarks##(text) : string ;



          cleanParaMarks : function(text){

          txt =  text.replace(/\\P/g, '\\textcolor{bibleParaMark} {\\P} \\ ');
                      txt =  txt.replace(/?/g, '\\textcolor{bibleParaMark} {\\P} \\ ');
                      txt =  txt.replace(/¶/g, '\\textcolor{bibleParaMark} {\\P} \\ ');

                                          return txt;

            }, // End.  twPan.cleanParaMarks(text)
```

```
//%: osCmd(commandText [, waitResponse]) : string | void ;

        osCmd  : function(){

        var waitResponse  = false;

    //  this.alert(this.osCmd.arguments.length);

              switch(this.osCmd.arguments.length)
                  {
                  case 0:
                    var retVal = {'status': twConst.SystemAccess_NotAttempted};
                   break;

                  case 1:
                    var retVal = TW.system(this.osCmd.arguments[0]);

                   break;

                  case 2:

                    waitResponse  = this.osCmd.arguments[1];
                    var retVal = TW.system(this.osCmd.arguments[0], waitResponse);
                   break;
                  } // End. switch(this.osCmd.arguments.length)


              switch(retVal.status)
              {
              case twConst.SystemAccess_NotAttempted:

                 this.alert('Please Specify a Command\n - that the System can Run');
               break;

              case twConst.SystemAccess_OK:

                 if (waitResponse == true) {return retVal.output; }
                                  // otherwise no return should be expected
                break;

              case twConst.SystemAccess_Failed:

                     this.alert(retVal.message
                           + '\nCode: '+ retVal.result // may be empty
                           + '\n\n' + retVal.output); // may be empty

              // even  if (waitResponse == true) {return nothing ; } and fail script


                break;
```

```
            case twConst.SystemAccess_PermissionDenied:

               this.alert(retVal.message);
                 return '';
              break;

               } // End. switch(retVal.status)


                                    }, // End.  twPan.osCmd(commandText [, waitResponse])

//%: findFromScript([string]) : void

 findFromScript : function ()
     {     with (TW.target)
     {
     var saveSelStart = selectionStart ;
     var saveSelectionLength = selectionLength ;

    if (this.findFromScript.arguments.length > 0 && this.findFromScript.arguments[0] != "")
      {
         useThis = this.findFromScript.arguments[0];

       }
       else
       {
        useThis = "Find Text"
       }


       selectRange(saveSelStart,0); // avoid overwriting existing text if selected
       insertText(useThis);
       selectRange(saveSelStart,useThis.length);

      copyToFind();
      insertText("");
      selectRange(saveSelStart,saveSelectionLength);
      doFindDialog();
     }
    },   // End.  twPan.findFromScript([string]) : void

//%:  ##escapeLatex##([string]) : string

   escapeLatex : function(LateXtext)
   {

   LateXtext = LateXtext + " ";

     if (    LateXtext == ""
```

```
        || LateXtext == " "
        || LateXtext == 'undefined'
        || LateXtext == undefined
        || LateXtext == null
        || LateXtext == "null"
        || LateXtext == "NULL"
       )
        { return '' ; }


    LateXtext = LateXtext.replace(/\n/g,"zxDcKvb") ;

    escapeList = [ "\\\\", "#", "\\$", "%", "&", "~", "_", "\\^", "{", "}"];

                for (escapee in escapeList)
                  {
                     buildReg = new RegExp( escapeList[escapee],"g");

                      LateXtext = LateXtext.replace(buildReg, "\\" +  escapeList[escapee]);
                                              }

                return LateXtext.replace(/zxDcKvb/g,"\n");

    },     // End. escapeLatex : function(LateXtext)

//%:  ##pasteToLatex##([string]) : string

    pasteToLatex : function(text)
    {

    text = this.escapeLatex(text);
    text = this.convertSpeechMarks(text);


    return text;

     },     // End. pasteToLatex(text)

//%:  ##latexFileName##([string]) : string

    latexFileName : function(startName)
    {
     escapeList = ['" ", "`", "\'", "'", "#", "\\$", "%", "&", "~", "_", "\\^", "{", "}", " "];

                for (escapee in escapeList)
                  {
                     buildReg = new RegExp( escapeList[escapee],"g");

                      startName = startName.replace(buildReg, "-" );
                                              }
```

```
                return startName;



  },      // End. latexFileName : function(startName)

//%:  getSnippet(query : string) : { dataRecords_ArrayOfObjectsof_fieldName_And_fieldValue : array,
commandUsed : string } : object

 getSnippet : function (database, table, chosen_id, chosenColumn)
      {
       /*
         For batch file and php argv[1] ...

             %1 DataBase - information_snippets
             %2 Table - entries
             %3 Id - chosen_titleId
             %4 Column - chosenColumn
            */

               var command =   'cmd /c getMySqldData.php.bat '
                           + database + ', '
                           + table +', '
                           + chosen_id + ' \'"
                           + chosenColumn +'\'";

     objectString  = this.osCmd(command, true);

       if (objectString.indexOf("Qhole query") > -1)
         {  this.alert(objectString); }
   else{
    return {
          'dataRecords_ArrayOfObjectsof_fieldName_And_fieldValue': JSON.parse(objectString),
          'commandUsed':command
          };
      }

   },// End. getSnippet : function (database, table, chosen_id, chosenColumn)

 //%:  toTitleCase (text : string) : string

 toTitleCase :function(text)
 {
  return text.replace(/(\\?[\w&`'??"?.@:\/\{\([<>_]+-? *)/g, function(match, p1, index, title) {
   if (index > 0 && title.charAt(index - 2) !== ":" &&
       match.search(/^(a(nd?|s|t)?|b(ut|y)|en|for|is|i[fn]|o[fnr]|t(he|o)|vs?\.?|via)[ \-]/i) > -1) // added is Paul
Norman
     return match.toLowerCase();
   if (title.substring(index - 1, index + 1).search(/['"_{(\[]/) > -1)
```

```
      return match.charAt(0) + match.charAt(1).toUpperCase() + match.substr(2);
    if (match.substr(1).search(/[A-Z]+|&|[\w]+[._][\w]+/) > -1 ||
       title.substring(index - 1, index + 1).search(/[\])}]/) > -1)
      return match;
    return match.charAt(0).toUpperCase() + match.substr(1);
  });
},     // End. twPan.toTitleCase (text : string) : string

//%:  getArrayIndexKeyFromValue(myArray ,arrayValue) : integer;

getArrayIndexKeyFromValue : function (myArray, arrayValue)
 {

   for (look in myArray)
      {
       if (myArray[look] == arrayValue)
         {
          return look;
         }
       }

       return -1; // not found

  }, // End. getArrayIndexKeyFromValue(myArray ,arrayValue) : integer;

//%:  cleanLineEnd(lineText) : string; // removes any start end spaces and any trailing new line
  cleanLineEnd : function(lineText)
   {
                    var text = twPan.trim(lineText);

                     do{
                       if (text.lastIndexOf("\n") == text.length -1)
                        {
                         text = text.substr(0, text.length -2);
                        }
                       }
                       while(text.lastIndexOf("\n") == text.length -1);

                     return text;

     }, // End. cleanLineEnd(lineText) : string;


//%:  emptyString( something : empty string?) : boolean

 // after http://stackoverflow.com/questions/154059/what-is-the-best-way-to-check-for-an-empty-
string-in-javascript/3215653#3215653  by Jet
                    emptyString : function (e)
                    {
                          switch(e) {
```

```
                                        case "":
                                        case 0:
                                        case "0":
                                        case null:
                                        case false:
                                        case undefined:
                                            return true;

            break;
                                                default : return false;
                                    }
                        }  // End.  emptyString( something : empty string?) : boolean


        }; // end  twPan

    twPan.initialise();


    $tw = twPan; //shorhand

   // TW.information(null, 'test', ' function global_load');

    function global_load(globalName)
     {
       if (TW.app.hasGlobal(globalName) === true)
         { return  TW.app.getGlobal(globalName);
         }
        else
         { twPan.alert('Global '+globalName+' not Available');
           return undefined;
                                                        }

      }

//    TW.information(null, 'test', ' function  create_helper');

     function create_helper(globalName) //
       {
        eval(global_load(globalName));
         }

    // Do following when needed in individual script, it is possible to overflow the stack
   // This form can be called even if twPan is not created

//       eval(TW.app.getGlobal("helper_PhpJs"));
//


/*
```

```
// Was

file_get_contents : function(fileNameFullPath){
          var scriptRoot = this.callingScript.scriptNamePath.substr(0,
               this.callingScript.scriptNamePath.lastIndexOf("/")) ;
              scriptRoot = scriptRoot.substr(0,scriptRoot.lastIndexOf("/") +1) ;

       var loadFile =  "\"" + scriptRoot + 'mod_loadFile.php\" \"
             + fileNameFullPath + '\"';  // \" to quote spaces in files' paths

        // Non-windows OSes won't need the cmd /c
         var text = TW.app.system('cmd /c php ' +  loadFile);
           return text;
                        }


 */


// Following alters the QtScript generic String object to add .toTitleCase()
//%:  String.toTitleCase()

/* To Title Case 1.1.1
 * David Gouch <http://individed.com>
 * 23 May 2008
 *
 * Copyright (c) 2008 David Gouch
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
```

```
DEALINGS IN
 * THE SOFTWARE.
 */

/* MODIFIED by Jonathan Kew for use with TeXworks: added optional leading
   backslash to the word-finding regex, to protect TeX control words */



//%:  Array.contains(element)
//from http://www.go4expert.com/forums/showthread.php?t=606>Extending JavaScript Arrays
// http://www.go4expert.com/forums/member.php?u=146 Pradeep

/* some sort of iteration problem for (x in array) over runs
Array.prototype.contains = function (element)
  {
        for (var i = 0; i < this.length; i++)
      {
           if (this[i] == element)
        {
                return true;
           }
        }
        return false;
  };
*/
//%:  Array.lastIndexOf(element)
/* Altered from http://www.go4expert.com/forums/showthread.php?t=974
             http://www.go4expert.com/forums/member.php?u=146 Pradeep

  ' Returns the index of the element matched from the end'
*/
/* some sort of iteration problem for (x in array) over runs
 Array.prototype.lastIndexOf=function(element)
 {
    var i=this.length;
    while(i--)
    {
       if(this[i]===element)
       {
          return i;
       }
    }
    return -1;
 }
*/
 var   buildText  = "";
 var here = this;

   function doDocumentation()
```

```
   {
/*
       if ( TW.question(null, "Get text of Object/Function List?", "Get text of Object /Function List?\nor
function list (.js)?" , msgBox.Yes | msgBox.No, msgBox.Yes) == msgBox.Yes)
    {  var scriptContents = TW.readFile(TW.app.getOpenFileName()).result;
     }
    else
    {
    return;
    }
*/

  var appName = TW.getText(null, "Application Name", "Application Name");

    if (appName == undefined)
      {
      return
       }

       var appNameDescription = TW.getText(null, "Application Description", "Application
Description");


    if (appNameDescription == undefined)
      {
      return;
       }

  var objFuncName = TW.getText(null, "Object/Function Name", "Object/Function Name", "this");


     if (objFuncName !== undefined)
      {

        if (objFuncName == "Process Function Names")
          {
            var  functionList = scriptContents.match(/\s(function)(?=\s*.*\()\s+[a-zA-Z_\-]*\(/g);
            if ( functionList !== false)
              {

             for (var x in functionList)
                {
                 functionName = ">" + functionList[x].replace(/function|\s+|\(/g,"")+"<";
               if (functionName != "><")
                 {
               //  TW.information(null,"",  functionName);
                 }

                }
             //  TW.information(null, "results", functionList.toString().replace(/,/g,"\n"));
```

```
                eval(scriptContents); // scriptFile

              docDone = selfDoc(appName, this, appNameDescription);
              makeInformation(docDone);
               }
            return;

           }
         else
          {

          docDone = selfDoc(appName, here,  appNameDescription);
          makeInformation(docDone);
          }
      }
     else
      {
     return;
      }



     TW.target.insertText(buildText);

  }

  function makeInformation(docDone)
    {
  //    TW.information(null, "result", docDone.properties.length);

  /*
         docDone.properties[p].name
         docDone.properties[p].comment
         docDone.properties[p].implementation
  */



     for (var p in docDone.properties)
       {


        if (docDone.properties[p].type != "function" || docDone.properties[p].implementation.indexOf
("[native code]")>-1) // avoid Tw and Qt globals
          {
          continue;
           }
       // twPan.alert(          docDone.properties[p].name
      //              + "\n  " + docDone.properties[p].type
```

```
      //              + "\n  " + docDone.properties[p].comment
      //              + "\n  " + docDone.properties[p].implementation
      //              );

        buildText += "\\subsection{" + twPan.escapeLatex(docDone.properties[p].name) + "}\n\n";
        buildText +=  "\\noindent " + twPan.escapeLatex(docDone.properties[p].comment) + "\n\n";
        buildText +=  "\\begin{lstlisting}["
                  + "caption=" + twPan.escapeLatex(docDone.properties[p].name) + ","
                  + "label=script." + twPan.escapeLatex(docDone.properties[p].name) + ","
                  + "fontadjust]\n\n";
        buildText += docDone.properties[p].implementation + "\n\n";
        buildText += "\\end{lstlisting}\\\n\n"
       // TW.app.clipboard = buildText;
          }

    }

  // from - https://github.com/rudenoise/selfDoc.js

 var selfDoc = (function (maxDepth) {
  maxDepth = maxDepth || 10;
  var doc, extract, loopProps, matchComment, replaceComment, splitLine, clean;
  doc = function (appName, root, overview) {
    // Accepts appName/string, root/function/object, overview
    // Returns an objects containing all nested properties, with documentation
    if (typeof appName  === "string" &&
        (typeof root  === "function" || typeof root === "object")) {
      return {
        appName: appName,
        overview: overview,
        comment: doc.parse(root),
        properties: loopProps(root)
      };
    }
    return false;
  };
  doc.parse = function (fun) {
    // doc.parse accepts fun/function/string as its argument
    // returns an array of strings
    // strings are extracted from uninterupted comment blocks at the top of the function (as illustrated
here)
    fun = typeof fun  === "string" ? fun : (fun + "");
    // pass in all but first line
    fun = fun.split(splitLine).slice(1);
    return extract(fun, [], true);
  };
  // PRIVATE
  matchComment = new RegExp("^ *\/\/.*$|^ *\/\\*.*$|^ *\\* .*$");
  replaceComment = new RegExp("^ *\/\/ |^ *\\* |^ *\/\\* |^ *\/\\*\\*");
  splitLine = new RegExp("\r\n|\n");
```

```
  clean = new RegExp("\n *//.*|\r\n *//.*", "g");
  loopProps = function (prop, depth) {
    depth = depth || 1;
    var k, arr = [];
    for (k in prop) {
      if (prop.hasOwnProperty(k)) {
        arr.push({
          name: k,
          type: typeof prop[k],
          implementation: (prop[k] + "").replace(clean, ""),
          comment: doc.parse(prop[k]),
          properties: depth > maxDepth ?
            ["..."] : loopProps(prop[k], (depth + 1))
        });
      }
    }
    return arr;
  };
  extract = function (lines, comments, uninterupted) {
    return lines.length === 0 || uninterupted === false ?
      comments :
      lines.slice(0, 1)[0].match(matchComment) === null ?
      // no match, recurse
      extract(lines.slice(1), comments, false) :
      // match, collect and recurse
      extract(lines.slice(1), comments.
        concat(lines.slice(0, 1)[0].
          replace(replaceComment, "")), true);
  };
  return doc;
}());



// TW.information(null, 'test', 'All Set up');
```

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

twConst

```
 const twConst = {

                SystemAccess_OK : 0,
              SystemAccess_Failed : 1,
      SystemAccess_PermissionDenied : 2

            }; // end twConst
```

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

msgBox

These are used in setting and reading message box buttons in critical information question warning and possibly createUI dialogues

```
const msgBox = {

 Ok              : 0x00000400, // An "OK" button defined with the AcceptRole.
 Open            : 0x00002000, // A "Open" button defined with the AcceptRole.
 Save            : 0x00000800, // A "Save" button defined with the AcceptRole.
 Cancel          : 0x00400000, // A "Cancel" button defined with the RejectRole.
 Close           : 0x00200000, // A "Close" button defined with the RejectRole.
 Discard         : 0x00800000, // A "Discard" or "Don't Save" button, depending on the platform,
defined with the DestructiveRole.
 Apply           : 0x02000000, // An "Apply" button defined with the ApplyRole.
 Reset           : 0x04000000, // A "Reset" button defined with the ResetRole.
 RestoreDefaults  : 0x08000000, // A "Restore Defaults" button defined with the ResetRole.
 Help            : 0x01000000, // A "Help" button defined with the HelpRole.
 SaveAll         : 0x00001000, // A "Save All" button defined with the AcceptRole.
 Yes             : 0x00004000, // A "Yes" button defined with the YesRole.
 YesToAll        : 0x00008000, // A "Yes to All" button defined with the YesRole.
 No              : 0x00010000, // A "No" button defined with the NoRole.
 NoToAll         : 0x00020000, // A "No to All" button defined with the NoRole.
 Abort           : 0x00040000, // An "Abort" button defined with the RejectRole.
 Retry           : 0x00080000, // A "Retry" button defined with the AcceptRole.
 Ignore          : 0x00100000, // An "Ignore" button defined with the AcceptRole.
 NoButton        : 0x00000000  // An invalid button.

  };
```

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

twPan

Wrapper and extender.

Many of these functions provide same or similar features as JavaScript would provide in the browser environment, and are simplified versions of the TW procedures they are built on.

This is very much the beginnings of a work in progress.

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

select

twPan.select(message, optionsArray) : string

Displays message to User and presents a drop down box of options to choose from.

Returns chosen option/item or an empty string if Cancel is chosen.

First Item is the default.

For more options use TW.getItem get List Choice getItem(QWidget*,QString, QString,QStringList) etc.

Example:

```
 var answer =  twPan.select("Please Make a Choice",
["Chocolate","Soya","Cream"]);
```

Created with the Personal Edition of HelpNDoc: Easily create Help documents

prompt

twPan.prompt(message) : string

Displays message to User and presents a blank text input for the User to fill out.

Returns typed text or an empty string if Cancel is chosen.

For more options use TW.getText(QWidget*,QString,QString,QString)

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

confirm

TW.confirm(message) : boolean

Displays message to User and presents Yes and No buttons.

Returns True if Yes button is chosen, and False for No or Cancel.

For more options and button choices use TW.question(QWidget*,QString, QString,int) etc...

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

alert

twPan.alert(message) : void

Shows message to User as in browser: window.alert('Hello');

The message box says "TeXworks Message" in title bar if visible in the User's OS

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

file_get_contents

twPan.file_get_contents(fileNameFullPath) : string

Attempts to load form disk the contents of fileNameFullPath

Upon a sucessful read of the file requested - returns a string reflecting the files contents.

On faliure displays a message to the User giving information (e.g. Tw file permissions not set) on the failure.

Created with the Personal Edition of HelpNDoc: Full featured multi-format Help generator

file_put_contents

twPan.file_put_contents(fileNameFullPath , fileContents) : void

Attempts to save to disk fileContents (a valid string) to the location file name fileNameFullPath

Upon a sucessful save of the file requested - returns silently.

On faliure displays a message to the User 'Please Turn on File Writing in \n\ Edit Menu / Preferences / Scripts'.

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

osCmd

osCmd("string to run on system" [, whether to wait for a result])

var dirList twPan.osCmd("cmd /c dir/b \"c:\temp\some folder\"", true);

twPan.osCmd("cmd /c notepad \"c:\sopme where\my File.txt\"");

```
//%: osCmd(commandText [, waitResponse]) : string | void ;

         osCmd  : function(){

         var waitResponse  = false;

    //  this.alert(this.osCmd.arguments.length);

               switch(this.osCmd.arguments.length)
                    {
                    case 0:
                       var retVal = {'status': twConst.
SystemAccess_NotAttempted};
                      break;

                    case 1:
                       var retVal = TW.system(this.osCmd.arguments[0]);

                      break;

                    case 2:

                       waitResponse  = this.osCmd.arguments[1];
                       var retVal = TW.system(this.osCmd.arguments[0],
waitResponse);
                      break;
                    } // End. switch(this.osCmd.arguments.length)
```

```
                switch(retVal.status)
                {
                case twConst.SystemAccess_NotAttempted:

                    this.alert('Please Specify a Command\n - that the System
can Run');
                 break;

                case twConst.SystemAccess_OK:

                    if (waitResponse == true) {return retVal.output; }
                                              // otherwise no return should be
expected
                  break;

                case twConst.SystemAccess_Failed:

                         this.alert(retVal.message
                                  + '\nCode: '+ retVal.result // may be empty
                                  + '\n\n' + retVal.output); // may be empty

                // even  if (waitResponse == true) {return nothing ; } and
fail script

                  break;

                case twConst.SystemAccess_PermissionDenied:

                    this.alert(retVal.message);
                       return '';
                  break;

                     } // End. switch(retVal.status)


                         }, // End.  twPan.osCmd(commandText [, waitResponse])
```

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

getRelativePath (Experimental)

(Experimental) twPan.getRelativePath(mainPath, wantedPath) : string

Very much need this, and could not find a JavaScript function on the web for it, and it does not appear to be in PhpJs either, so it is experimental.

It was needed for working out relative file paths for graphics from main Tex document especially for use in Portable Tw situations.

MiKTeX and LaTeX endeavour to find a graphic on a relative path.

mainPath could be the main document's path, and **wantedPath** is the full path to the graphic, above, at or below the main document's position.

Relative path returned, or full path if in different tree.

Could be used as in ...

```
// TeXworksScript
// Title:  Insert Graphic(s) to Col Width
// Description: Allows multiple selection of file names and inserts with title
description and copyright note, using column width
// Author:  Paul Norman
// Version: 0.3
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+G, Alt+I


  eval(TW.app.getGlobal("helper_twPan")); // Un-Comment if Needed

//  eval(TW.app.getGlobal("helper_PhpJs")); // Un-Comment if Needed

//%: insertGraphics


var chosenImages =   TW.app.getOpenFileNames("Image Files (*.jpg *.jpeg *.
png *.pdf)");


if (chosenImages.length > 0)
                    {
                     for (image in chosenImages)
                         {
        var graphicPath =   twPan.getRelativePath(TW.target.fileName,
chosenImages[image]);

        var graphicCaption = twPan.prompt("Graphic Caption");
           graphicCaption = graphicCaption.toTitleCase();

        var copyRightInfo = twPan.prompt("Copyright Info");

        var includeGraphicsOptions = twPan.select("Graphic Options", ["[width=\
\ScaleIfNeeded]","","[]","[width= ]"])

           TW.target.insertText(
                        "\\noindent\\includegraphics"
                        + includeGraphicsOptions
                        +"{"
                        + graphicPath
                        +"}\\captionCentredNote{"
                        + graphicCaption
                        +"}{"
                        + copyRightInfo
                        +"}\n"
                        );

                          } // End.  for (image in chosenImages)
                    } // End. if (chosenImages.length > 0)
```

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

.toTitleCase()

.toTitleCase()

Is added to the QtScript String object in the current Scripting session.

e.g.

```
  MySting.toTitleCase();

   var changed =  "some text".toTitleCase();

   twPan.alert(changed);
```

Would show "Some Text".

```
* David Gouch <http://individed.com>
* 23 May 2008
* Copyright (c) 2008 David Gouch
```

Created with the Personal Edition of HelpNDoc: Free PDF documentation generator

helper_PhpJs.mod

```
// STEP THINGS THROUGH ONE STEP AT A TIME
// original phpjs.namespaced.mod from http://phpjs.org/ * Dual licensed under the MIT (MIT-LICENSE.txt) * and GPL (GPL-LICENSE.txt) licenses.

var window = {};  // fix for not being in a browser environment

eval(TW.app.getGlobal("phpjs_namespaced"));
var PhpJs = new PHP_JS();

/*
Do following when needed in individual script, it is possible to overflow the stack
This form can be called even if twPan is not created
eval(TW.app.getGlobal("helper_PhpJs"));
This will create PhpJs and all its members for use
*/
```

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

PhpJs Function List

All this is available thanks to the fantastic work done at http://phpjs.org/

There will be redundant functions in this list. Most string, array and numeric functions will be useful in TeXworks Scripting. I have not tried them all yet.

- abs

- acos
- acosh
- addslashes
- array_change_key_case
- array_chunk
- array_combine
- array_count_values
- array_diff
- array_diff_assoc
- array_diff_key
- array_diff_uassoc
- array_diff_ukey
- array_fill
- array_fill_keys
- array_filter
- array_flip
- array_intersect
- array_intersect_assoc
- array_intersect_key
- array_intersect_uassoc
- array_intersect_ukey
- array_key_exists
- array_keys
- array_map
- array_merge
- array_merge_recursive
- array_pad
- array_pop
- array_product
- array_push
- array_rand
- array_reduce
- array_reverse
- array_search
- array_shift
- array_slice
- array_splice
- array_sum
- array_udiff
- array_udiff_assoc

- array_udiff_uassoc
- array_uintersect
- array_uintersect_assoc
- array_uintersect_uassoc
- array_unique
- array_unshift
- array_values
- array_walk
- array_walk_recursive
- arsort
- asin
- asinh
- asort
- atan
- atan2
- atanh
- base64_decode
- base64_encode
- base_convert
- bin2hex
- bindec
- ceil
- checkdate
- chop
- chr
- chunk_split
- class_exists
- compact
- cos
- cosh
- count
- count_chars
- crc32
- date
- decbin
- dechex
- decoct
- deg2rad
- doubleval
- echo

- end
- exp
- explode
- expm1
- floatval
- floor
- fmod
- get_class
- get_defined_vars
- get_headers
- get_html_translation_table
- getdate
- getrandmax
- hexdec
- html_entity_decode
- htmlentities
- htmlspecialchars
- htmlspecialchars_decode
- http_build_query
- hypot
- implode
- in_array
- intval
- ip2long
- is_bool
- is_double
- is_finite
- is_float
- is_infinite
- is_int
- is_integer
- is_long
- is_nan
- is_null
- is_numeric
- is_real
- is_scalar
- is_string
- join
- json_decode

- json_encode
- krsort
- ksort
- lcfirst
- lcg_value
- levenshtein
- log
- log10
- log1p
- long2ip
- ltrim
- max
- md5
- method_exists
- microtime
- min
- mktime
- mt_getrandmax
- mt_rand
- natcasesort
- natsort
- nl2br
- number_format
- octdec
- ord
- parse_str
- parse_url
- pi
- pow
- preg_grep
- preg_quote
- print_r
- printf
- property_exists
- quotemeta
- rad2deg
- rand
- range
- rawurldecode
- rawurlencode

- reset
- round
- rsort
- rtrim
- serialize
- setcookie
- setrawcookie
- settype
- sha1
- shuffle
- sin
- sinh
- sizeof
- sort
- soundex
- split
- sprintf
- sql_regcase
- sqrt
- str_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcspn
- strip_tags
- stripos
- stripslashes
- stristr
- strlen
- strnatcasecmp
- strnatcmp
- strncasecmp

- strncmp
- strpbrk
- strpos
- strrchr
- strrev
- strripos
- strrpos
- strspn
- strstr
- strtok
- strtolower
- strtotime
- strtoupper
- strtr
- substr
- substr_compare
- substr_count
- substr_replace
- tan
- tanh
- time
- trim
- uasort
- ucfirst
- ucwords
- uksort
- unserialize
- urldecode
- urlencode
- usort
- utf8_decode
- utf8_encode
- var_dump
- var_export
- vprintf
- vsprintf
- wordwrap

Created with the Personal Edition of HelpNDoc: Easy to use tool to create HTML Help files and Help web sites

mod_general.js

```
// TeXworksScript
// Title: Pre-Load Globals
// Description: Pre-loads gloabal varibales used in script modules
// Author: Paul A Norman
// Version: 0.3
// Date: 2010-12-19
// Script-Type: hook
// Hook: TeXworksLaunched

// TW.information(null,"Hello","Hello from NewFile Hook"); // will change to ApplicationReady hook later on

function setUp(globalName, fileName)
{
with(TW.app)
   {
     if (hasGlobal(globalName) == false) // this test should be redundant
        {
         var fileResult = TW.readFile(fileName);
         setGlobal(globalName, fileResult.result) // makes it available for later eval() in other scripts
        }
   }
}

setUp("helper_twPan", "helper_twPan.mod");// later eval() of this will create:
// twConst, msgBox, twPan, global_load(globalName), create_helper()

setUp("phpjs_namespaced", "phpjs.namespaced.mod"); // later eval() of this will create:
// PHP_JS(); object factory function

setUp("helper_PhpJs",'helper_PhpJs.mod'); // later eval() of this will create:
// an object called window and PhpJs from new PHP_JS(); see helper_PhpJs.mod for more information
```

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

mod_reload.js

```
// TeXworksScript
// Title: Re-load Globals
// Description: Re-loads gloabal varibales used in script modules
// Author: Paul A Norman
// Version: 0.3
// Date: 2010-12-19
// Script-Type: standalone
// Context: TeXDocument
// Shortcut: Alt+R, Alt+L
```

```
// TW.information(null,"Hello","Hello from Module Re-Load");


   function setUp(globalName, fileName)
    {

        var fileResult = TW.readFile(fileName);

        TW.app.setGlobal(globalName, fileResult.result); // makes it available for later eval() in other
scripts
    }


    setUp("helper_twPan", "helper_twPan.mod");//  later eval() of this will create:
                                    //  twConst, msgBox, twPan, global_load(globalName), create_helper()

    setUp("phpjs_namespaced", "phpjs.namespaced.mod"); // later eval() of this will create:
                                        // PHP_JS(); object factory function

    setUp("helper_PhpJs",'helper_PhpJs.mod'); //  later eval() of this will create:
                            // an object called window and PhpJs from new PHP_JS();
```

Created with the Personal Edition of HelpNDoc: Full featured Help generator

Existing Libraries Modules

MUCH IMMEDIATELY USEFUL CODE.

php.js

php.js is an open source project that brings high-level PHP functions to low-level JavaScript platforms such as web browsers, browser extensions (Mozilla/ Firefox, Chrome), AIR, and SSJS engines like V8 (node.js, v8cgi), Rhino, and SpiderMonkey (CouchDB)

If you want to perform high-level operations on these platforms, you probably need to write JS that combines its lower-level functions and build it up until you have something useful like:strip_tags(), strtotime(), number_format (), wordwrap().

http://phpjs.org/

For an example of a user TeXworks implementation of this see helper_PhpJs.mod

CODE WORTH EXPLORING.

JS.Class

`JS.Class` is a set of tools designed to make it easy to build robust object-oriented programs in JavaScript. It's based on Ruby, and gives you access to Ruby's object, **module** and **class** systems, some of its **reflection** and **metaprogramming** facilities, and a few of the packages from its standard library. It also provides a powerful **package manager** to help load your applications as efficiently as possible.

http://jsclass.jcoglan.com/

Created with the Personal Edition of HelpNDoc: Full featured Help generator

Areas for Possible Future Adaptation

Customisable Conversion Chunks of HTML/RDF/XML etc to be inserted in TeXworks as LaTeX

XML.ObjTree - XML source code from/to JavaScript object like E4X

XML.ObjTree class is a parser/generater for XML source code and JavaScript object. This is a JavaScript version of XML::TreePP for Perl.
http://www.kawa.net/works/js/xml/objtree-e.html

ajaxslt

"Pure javascript implementation of XSLT and XPath ... This DOM comes with a minimal XML parser that can be used to generate a suitable DOM representation of the input documents if they are present as text."
http://code.google.com/p/ajaxslt/

Pure JavaScript HTML Parser

" ... decided to go about writing a pure JavaScript HTML parser ... it's able to get the job done with little fuss - while still being highly portable.
http://ejohn.org/blog/pure-javascript-html-parser/
http://ejohn.org/files/htmlparser.js

Envjs

"The goal of Envjs is to provide a highly portable javascript implementation of the Browser as a scripting environment (*often referred to as a 'headless' browser*). ...
"Given the follow task you could have written it several different languages, straight bash, python, ruby, perl, etc. *But I also know I could write in half the time with half as many lines using* **jQuery** *because that's what I'm really good at.* Now I finally can..."
http://www.envjs.com/docs

(see http://code.google.com/p/html5lib http://code.google.com/p/html5lib/wiki/UserDocumentation for HTML5 Python and Php headless browsers - Paul)

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

Byond Expected Js/EMCA

jspdf

jsPDF generates PDF documents using pure JavaScript
(Useful for quick snippets - Paul)
http://code.google.com/p/jspdf/

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Permissions

Permissions

This help was compiled using commercial program which is free to use for non-commercial purposes, an explanation follows on the next page contd ...

Tim Caswell has kindly given permission for his articles from to be incorporated here. contd...

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

HelpNdoc Permission - to use

---------- Original Message ----------
Subject: Re: [General Information] Sharing help files on Open Source Project
From: "Paul A Norman" <paul.a.norman@gmail.com>
Date: Fri, November 5, 2010 11:13 pm
To: "IBE Software - Support" <support@ibe-software.com>

Thank you John, much appreciated,

Paul

On 5 November 2010 21:02, IBE Software - Support
<support@ibe-software.com> wrote:
> Dear Paul,
>
> Thank you for your interest in HelpNDoc and for considering using it.
>
> According to your description, it looks like your use of HelpNDoc completely
> fits in the Personal Edition license terms so we are glad to tell you that
> you can use it freely for your project and for as long as you'd like.
>
> Please let me know if I can be of further assistance.
>
> Best regards,
>
> John, IBE Software.
> support@ibe-software.com
> http://www.ibe-software.com
> http://www.helpndoc.com
> http://www.7capture.com
>
> -- Did you know ? HelpNDoc can generate online HTML help web sites --
>
>
> On 05/11/2010 05:10, paul.a.norman@gmail.com wrote:
>>
>> Paul Norman sent a message using the contact form at
>> http://www.helpndoc.com/contact.

>>
>> Bon jour Helpndoc,
>>
>> I am a user of an open source (non-commercial) text editor called TeXWorks
>> (http://code.google.com/p/texworks/)
>>
>> I am compiling some help information for their scripting engine and
>> usage, and *TeX (http://tug.org/) and LaTeX (
>> http://www.latex-project.org/lppl/) and would like please to be able to
>> freely share that with other TeXWorks, LaTeX, *TeX Users if I make it with
>> Helpndoc personal edition, if that was ok please?
>>
>> Many thanks for your consideration,
>>
>> Paul
>

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

Tim Caswell's Graphjic Presentation opf Js Objects

Tim Caswell has kindly give permission for his articles from to appear in these notes:

from **Tim Caswell** <tim@creationix.com>
to Paul A Norman <paul.a.norman@gmail.com>
date 11 November 2010 07:45
subject Re: Preparing Help Information

Yes, the content of the articles on howtonode is copyrighted to the author (as stated in the footer) I give you permission to inline the articles as long as you attribute back to thehowtonode.org site.

Good luck, JavaScript is a fun adventure. I'm still learning every day, so don't be surprised to see update my articles and remove errors as I continue to learn.

to Tim Caswell <tim@creationix.com>
date 11 November 2010 16:10
subject Re: Preparing Help Information
mailed-by gmail.com

Dear Tim,

Thanks for that, very much appreciated - just be sure first - I don't know of you followed the link on (http://twscript.paulanorman.com/) but they can download the compiled help as .chm, .pdf or .html (http://twscript.paulanorman.com/docs/index.html) is that ok with you?

I just want to be very sure before motoring ahead.

Will keep an eye on your postings - and try to update material as necessary!

Paul

from	**Tim Caswell** <tim@creationix.com>
to	Paul A Norman <paul.a.norman@gmail.com>
date	11 November 2010 16:43
subject	Re: Preparing Help Information

Yeah, that's fine

Sent from my Nexus One

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

Resources

Created with the Personal Edition of HelpNDoc: Single source CHM, PDF, DOC and HTML Help creation

Help Utilities

Doctor JS

http://doctorjs.org/

http://doctorjs.org/try.html

"Doctor JS is a genius. ... He'll analyze your JavaScript code ..Just POST to **http://doctorjs.org/analyze** with the src field set to your source code, and the doctor will give you an analysis in JSON. "

The Regex Coach - interactive regular expressions

The Regex Coach is a graphical application for Windows which can be used to experiment with (Perl-compatible) regular expressions interactively. It has the following features:

http://weitz.de/regex-coach/

(For another ground up explanation also see http://www.websina.com/bugzero/kb/regexp.html)

BESEN - *Js/EMCA IDE*

BESEN is an acronym for "Bero's EcmaScript Engine", and it is a complete ECMAScript Fifth Edition Implemention in Object Pascal, which is compilable with Delphi >=7 and FreePascal >= 2.5.1 (maybe also 2.4.1).

Its will compile under Lazarus on numerous Linux distributions, Mac, and on Windows under Lazarus or Delphi.

```
(BESEN is copyrighted free software by Benjamin Rosseaux
<benjamin@rosseaux.com>.
 You can redistribute it and/or modify it under either the terms of the
AGPLv3
 or the conditions below ...)
```

Also a command line shell for Js/EMCA - Linux 386, Linux AMD64, and Windows.
http://besen.sourceforge.net/

English http://blog.rosseaux.net/en/posts/l0kce9#l0kce9
German http://blog.rosseaux.net/posts/l0kce9#l0kce9

Created with the Personal Edition of HelpNDoc: Free help authoring tool

References

Web References

Created with the Personal Edition of HelpNDoc: Create HTML Help, DOC, PDF and print manuals from 1 single source

Web ReferencesScripting TeXworks

http://code.google.com/p/texworks/wiki/ScriptingTeXworks
Updated Oct 21, 2009
by jfkthame

TeXworks Manual AllTeXing
http://www.leliseron.org/texworks/
Updated mid 2010(?)
Alain Delmotte ("with the help of: Jonathan Kew, Stefan Löffler, David J. Perry, Joel C. Salomon and Joseph Wright; but the errors are still mine!")

ECMAScript Reference
http://doc.trolltech.com/4.7/ecmascript.html

Licensees holding valid Qt Commercial licenses may use this document in accordance with the Qt Commercial License Agreement provided with the Software or, alternatively, in accordance with the terms contained in a written agreement between you and Nokia.
Alternatively, this document may be used under the terms of the GNU Free Documentation License version 1.3
as published by the Free Software Foundation.) © 2008-2010 Nokia Corporation and/or its subsidiaries. Nokia, Qt and their respective logos are trademarks of Nokia Corporation in Finland and/or other countries worldwide. All other trademarks are property of their respective owners.

Using JavaScript as a Real Programming Language
http://labs.oracle.com/techrep/2007/abstract-168.html
http://labs.oracle.com/techrep/2007/smli_tr-2007-168.pdf

"In this paper we summarize our experiences using JavaScript, focusing especially on its use as a real, general-purpose programming language"

Created with the Personal Edition of HelpNDoc: Free CHM Help documentation generator

Copyright/Licnse Notices

Copyright 2010 2011 see Introduction - Disclaimer - Copyright
Paul A Norman P.O. Box 1005, Nelson 7040, New Zealand. http://PaulANorman.com

Project:Copyrights

MDC wikis have been prepared with the contributions of many authors, both within and outside the Foundation. Unless otherwise indicated, the content is available primarily under the terms of the Creative Commons: Attribution-Sharealike license v2.5 or any later version.

Code samples added to this wiki before August 20, 2010 are available under the MIT license; use the following attribution information: "© <date of last wiki page revision> <name of person who put it in the wiki>".

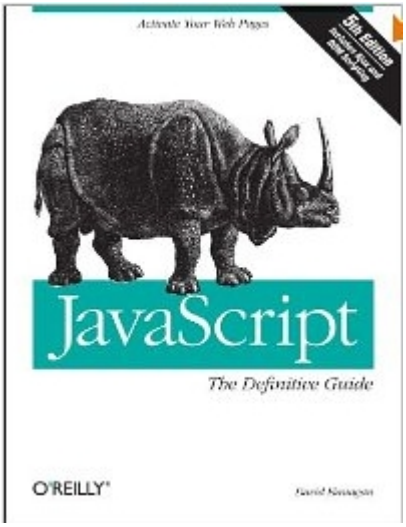
Code samples added on or after August 20, 2010 are in the public domain. You can use the following notice if necessary: "Any copyright is dedicated to the Public Domain."

While it's our intention to make the entire wiki available under these licenses, some wiki entries may be covered by other licenses. If a page is to be under a license other than the licenses noted above, it must be indicated at the bottom of each page by way of an Alternate License Block.

---

davidflanagan.com/javascript5

Examples from *JavaScript: The Definitive Guide, Fifth Edition*



http://www.davidflanagan.com/javascript5/

```
// This code is from the book JavaScript: The Definitive Guide, 5th
Edition,
// by David Flanagan. Copyright 2006 O'Reilly Media, Inc. (ISBN
#0596101996)
```

Created with the Personal Edition of HelpNDoc: Full featured Documentation generator

New topic

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator