QS-TEX: Basic guide

September 24, 2017

1 What QS-TEX does?

QS-TEX is a program designed to allow the implication of code within a LaTeXscript and thus allowing scripts and documents to be kept together.

2 Getting Started

To get started you need to download the following documents from:

`https://github.com/QuantumSpaghettification/QS-TEX`

Into a directory named e.g. QS-TEX:

- FUNC directory
- LANG directory
- editor.py

Into the directory of your LaTeXdocument:

- QS.config

into this you must change the path to the path of your QS-TEX directory and have the option to change the sytle of listings (use '!bs!' for back slashes in this file).
The basic syntax for including script into your LaTeXprogram is:

```
<(!)script=lang:action={}>
your script here
</(!)script>
```

without the '(!)' and lang replaced by the language of your choice (see below). To compile your program you need to run (on Linux Mint[1]):

```
python2.7 <path-to-QS-TEX>/editor.py <file-Name>.tex
```

This will create a number of files inluding temp.tex which you should run your standard LaTeXcompiling program on.

3 Languages

3.1 Languages Supported

So far the following languages are supported.

Language	**lang**
Plain Text	plain
Python v2.7	python2.7
R	R

at the moment the number of langauges are limited but making new ones are easy...

[1] Other OS's may vary.

4 Adding a New Language

To create a new language you need to:

- add a ‘¡name-of-language¿.lang’ file to the LANG directory.
- add a ‘¡name-of-language¿’ subdirectory o the FUNC directory.

The ‘.lang’ file must take the form:

```
!!!<name-of-varible>={<Quantity>}!!!
```

and must include the following varibles:

- file_ext: The file extention of your language.
- lstlisting: What listing sytle you want to use (this can be left empty).
- back_slash_escape: Things that need to be escaped in listings.
- run_command: The command line command to run your script language.
- func_output: A function named ‘output’ written in the language under consideration which outputs a string to a file called ’outputfc_!!i!!.txt‘ (the !!i!! is replaced by the editor.py script).
- func.preamble: What needs to be placed at the begining of each script in this language.
- func.scrinc: A function which outputs a string ‘!!norm!!’ or ‘!!HF!!’ depending on arugment and replaced ‘!bs!’ in this string with backslashes.

5 Actions

Actions are included in the main header e.g.:

```
<(!)script=python2.7:action={!!<name-of-action1>!!!!<name-of-action2
    >!!}>
script
</(!)script>
```

again without the (!)’s included. Currently the actions avaible are:

- sid=...: scripts with the same sid (script id) are run together and sequentially, e.g. if you define a varible in one you can use it in the other etc.
- LinNoId=...: scripts with teh same LinNoId (line number id) have their linenumbers carried on sequentially in listings.
- output=...: Normally every script must have an output() in it (this can be left empty). The exception is when you give an output action in which case the string ‘...’ is given as the output. This can include the scrinc() function which outputs the script.

6 Examples

As well as this whole document been created with the QS-TEX program; here are some basic examples of working scripts (mainly in python2.7 - sorry):

```
<script=python2.7:action={}>
a=1
output(scrinc('HF')+' Output:'+str(a))
</script>
```

Output:1

```
<script=python2.7:action={!!sid=1!!!!LinNoId=1!!}>
def adder(a):
  return(a+1)
output(scrinc('HF'))
</script>
```

The above function will be usuable wih any script with the same sid e.g.

```
<script=python2.7:action={!!sid=1!!!!LinNoId=1!!}>
output(scrinc('HF')+' Output:'+str(adder(10)))
</script>
```

Output:11

```
<script=python2.7:action={!!output=scrinc('HF')!!}>
x=1
</script>
```