Minimal Kiosk Browser Manual

Content

1) What this is all about – and how it came along

When I started to use my Raspberry Pi there was one thing I had in mind: to create a replacement for my Kaiboer K200 media player (a Popcornhour clone) with a few features I missed on this machine: a more modern browser (the Syabas browser supports only HTML 3.2) to access the internet, support for reading PDF documents and more …. I'll describe later, how I managed to achieve this.

So I was looking for a slim, fast web browser that should also be able to run without booting to the Desktop and also should provide a "kiosk mode" (full screen content without any kind of interface like window elements, toolbar etc.) to create embedded applications with a browser interface. When Ralph Glass published his "Minimal Web Browser" this seemed to be the right way to go. It was based on the webkit engine (like Midori, Chromium and others) and added only a minimalistic interface. Ralph also had some nice ideas to add support for streaming video and youtube. And it was by far the fastest browser on the Raspberry Pi with full JavaScript support and also HTML 5 support (there may be faster browsers which lack this kind of support – there's always a price to pay).

So I started playing with the code and added PDF support. In the beginning I worked together with Ralph but then he somehow lost interest in his project. At that time I forked it into my own development and called it "Minimal Kiosk Browser" (kweb). I'm not really a C programmer and hadn't done anything in C for about 20 years, but with Ralph's code as starting point I was able to realize step by step, what I had in mind.

I added a python script (kwebhelper.py), that "glued" everything together: Minimal Kiosk Browser, omxplayer (the only media player on the Raspberry Pi with hardware support), mupdf or xpdf for PDF support, youtube-dl to access videos from youtube and other video websites …. and in fact any other program you want to use (I'll explain that later on). kwebhelper.py is a kind of replacement for the plug-in support which all or most browsers on the Raspberry Pi are missing. It can be configured by the user in many ways to match his needs. In the first version you had to reinstall it afterwards; since version 1.1 there is a separate settings file, which can be configured at runtime.

I wrote this for my own use, but soon decided to make it available for other users.

2) What you get and how you can use it.

Minimal Kiosk Browser is available as a tar.gz file, that requires to install the program from a terminal with "sudo install.sh" (after unpacking it, of course). It contains the complete source code, so anybody can modify it, if he likes.

You may use it simply as any other browser from the desktop; it's slim and fast and nevertheless supports lots of things that other browsers don't: PDF documents with some special features, streaming video and audio including m3u and pls playlists, access to all embedded videos that are supported by youtube-dl (quite a lot of websites). Don't expect it to work like other browsers on your desktop computer: PDF documents are not displayed inside the browser window but with a separate program; videos are played full screen with omxplayer (and not inside your browser window) and so on. It can also be started from a terminal (or script) with some configuration options.

You can also use it standalone without starting the desktop environment, either in kiosk mode or in full window mode. It still needs X and some kind of window manager, but everything will be smoother and faster without the desktop overhead. You even can use it as replacement for your desktop, as you can call any other program from kweb, either from the URL entry line (putting a '#' in front) or from a special kind of links inside your homepage. After booting to the command line, kweb can be started with:

```
xinit ./kiosk
```

where "kiosk" is a small script file in your user directory. I've supplied two example kiosk files, using different window managers, which you can directly use or modify to your liking.

And there is one more requirement that makes kweb special: you need a file "homepage.html" in your user directory, which is opened, when the browser is started without any arguments, and also, when you click the "home" icon in the toolbar. This may be seen as a restriction, but it offers a lot of options: for example, there is no "favourites" function or database in kweb (remember: it is small and slim and therefore fast!). But you can simply add links to your favourite websites to your homepage and so replace the missing function. And you can do much more with it, like creating an interface for any kind of embedded application or a kind of desktop replacement, which enables you to open any kind of additional program that you like to use without going to the desktop.

I'm aware that not every user will be able to create his own homepage, as that requires a bit of knowledge about HTML (but the main reason for the existence of the Raspberry Pi is to learn something new, so you might as well start learning HTML). I've included a very simple example homepage.html file for a start. And since version 1.2 it contains a Python script (homepagecreator.py) that will create a homepage from some simple text files. Examples are provided which you can edit any way you like. If you don't like the way it looks, you can modify the HTML template (colours, fonts etc.)

If you don't want to use a homepage file at all, you can specify a website as your homepage from the command line. See chapter 6 for details.

3) Installation

I assume that you have a full blown Raspbian system installed. Otherwise you may have a problem with missing dependencies or helper programs. You will also need some other programs to use all features kweb offers. Since version 1.3 I've added a small Python script, that will check your system and tell you what need to install and how.

To install kweb, open a terminal and run the following commands:

```
wget http://steinerdatenbank.de/software/kweb_1.4.tar.gz
tar -xcf kweb_1.4.tar.gz
cd kweb-1.4
sudo ./install.sh
./check.py
```

The last program will check your system and tell you, if you need to install additional software and how to do it. It will also create a default homepage.html file in your user directory, but only if you don't have one already.

To remove kweb from your system, simply run

```
sudo ./remove.sh
```

If you want to use the homepage creator or try some of the examples, you have to unpack the tools archive:

```
tar -xcf tools.tar.gz
```

To use the homepage creator and create a homepage with it:

```
cp ./tools ~/
cd ~/tools
python homepagecreator.py
```

I'll explain later on, how you can modify this homepage.

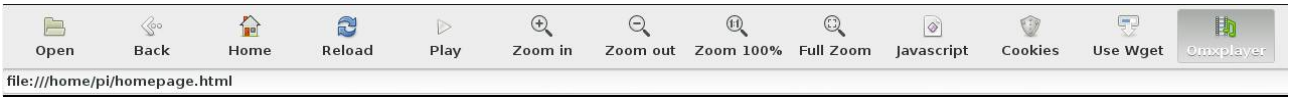
Now you are ready to use kweb (from your desktop). Start it like any other program (you will find Minimal Kiosk Browser in the Internet section) or call it from a terminal (or script) with some options like that:

`kweb url` (will open the url directly)
`kweb -options` (where options is a string built from certain characters as explained later on)
`kweb -options url`

"url" can be a any web or file address starting with "http://" or "file://".

4) Basic usage

In the upper part of the window you will see a toolbar with 13 icons and an URL entry text field below the toolbar. That's all – no menus, forms or things like that. Simple, isn't it?



a) The toolbar – all icons explained

"**Open**" - opens a file browser to select a file to show in kweb. HTML and text files (.txt) as well as images (JPEG, PNG and GIF) will be directly opened and shown inside the browser. Alls kinds of audio and video files, including m3u and pls playlist files, will be played with omxplayer. PDF files will also be opened with either mupdf or xpdf. And if the "X" option is set within the command line options string, all kind of executable files (scripts or binaries) are also opened and executed.
"**Back**" - go back to the last web page
"**Home**" - display your homepage.html file (or any other homepage set on the command line with the "H" options)
"**Reload**" - reload current page
"**Play**" - try to play embedded video (youtube and others) using omxplayer and youtube-dl. This function will also work with embedded HTML5 video tags.
"**Zoom in**" - magnify content by 10%

“**Zoom out**” - scale down by 10%
“**Zoom 100%**” - display content in original size

The last five icons are toggle buttons, that you can use to enable or disable some important options:

“**Full Zoom**” - zoom every element within the web page (full zoom) or zoom only text elements. The background of this new setting (introduced in version 1.3) is the fact, that webkit gets extremely slow (by a factor of about 10), if full zoom mode is enabled, zoom is not set to 100% and the web page contains color runs (and perhaps some other graphical elements). This may really freeze your Rpi for some time.
“**Javascript**” - enable / disable the use of Javascript and reload current web page. It's a good practice (speed!) on the Raspberry Pi to only enable Javascript when a web page really needs it to be displayed correctly.
“**Cookies**” - enable / disable support of cookies.
“**Use wget**” - if enabled, wget will be used to download any kind of file to your computer (using the right click menu), otherwise webkit's internal download method will be used (new in version 1.3). Both methods have their own advantages. If using wget, for example, a download folder on an external HD can be used (selectable in kwebhelper_settings.py). The internal method supports session cookies, which are not supported by the wget method. By default, all downloads are saved inside the “Downloads” folder in your user directory (which will be created, if it does not exist).
“Omxplayer” - if enabled (default), omxplayer will be used for all kinds of audio, video and media stream content. If you disable it, the gstreamer support built into the webkit library will be used for audio and video and try to reproduce the media content inside the browser window. As the current version of libwebkit does not support hardware accelerated video, this will only work with a few audio formats, but this may change in future webkit versions for the Raspberry Pi.

All these functions can also be called from the keyboard, if enabled by an options string (see below for details)

b) The URL entry text field – some niceties

The use seems to be obvious, but there are some additional functions.

Enter a web adress and press “Return”. If “http://” or “ftp://” or “file://” in front are missing, “http://” will be inserted automatically.

Enter a search text with “?” in front to open a web search. startpage.com is now the default search engine (introduced in version 1.3). It uses Google search but prevents Google from getting any information about you. If you don't mind, that Google (and with it the NSA) records all your searches and also which search results you visit, you can enable google.com as your search engine with the “G” option in a command line options string.

Enter a search string with “/” in front to search inside the current web page.

Enter a command line (as in a terminal) with “#” in front. This way you can start any kind of program including parameters. The command will be executed inside a terminal, but GUI programs can be called directly, if these are set in the configuration part of kwebhelper_settings.py (see below). Since version 1.3 most GUI programs are detected automatically and don't have to be added to the settings any more.

c) Functions inside the browser window

Links to PDF files will open the PDF documents with mupdf or xpdf. If the link contains additional information like “.....pdf#page=22”, the PDF document will be opened with that page (22 in this example). The PDF documents are downloaded to the “Downloads” folder first. If you click the same document link a second time, it will be opened from the Downloads folder directly.

Links with mime types starting with “audio/” or “video/” including m3u and pls playlist files and certain streaming links will be opened and played with omxplayer (if file type and codec are supported).

There is a special link type which only can be used in local HTML files, that will call programs or issue commands. This is explained in more detail below; homepagecreator.py will create a homepage containing such commands. There are also some examples in the default homepage.html file.

Right clicking on a link will give you a menu with a few options:
open link – will open the web site, same as left click
open link in new tab – will open in a new browser window
download link – will save the file from the link address into your “Downloads” directory.

copy link address – to paste it somewhere

Since version 1.2 “spacial navigation” has been enabled. You can use the arrow keys to navigate between form elements or links. To enable highlighting of selected links, you have to use a matching style for a.focus in your html page.

d) Calling other programs from kweb – kweb's special command links and command forms

If you want to use more programs than kweb supports and uses automatically (mupdf or xpdf, omxplayer), you can call them via the URL entry line with a ”#” in front. This will not be possible in kiosk mode, of course, but kweb supports a special kind of link that you can embed into your homepage.html file (or any other HTML file you want to use). It will not work from pages served by a web server (for security reasons, but there is a special option now to enable this for local servers). The following example will include a “Shut down” command into your HTML file:
Shut down

the basic syntax for href is:
file:///homepage.html?cmd=commandline
Spaces in the command line must be replaced with “%20”.

If you use the homepagecreator.py script to generate your homepage file, it will create these links automatically for you from text files containing lines like that:
Shut down=sudo shutdown -h now
This way you can call any kind of program or your own script commands. Kweb will try to check, if it is a Desktop GUI application; if that is not successful, the command will be executed from a terminal. If you want to avoid that (for GUI programs not automatically found or for scripts you want to run in background), you may add these programs to your kwebhelper settings. It contains a list of programs, that are executed directly without starting a terminal first:

```
direct_commands = ['kwebhelper.py','omxplayer']
```

You can edit this line (even at run time) to add your preferred programs. More details about kwebhelper configuration are explained in chapter 8.

Not all things are possible with such command links (URLs or file paths containing spaces as arguments , for example, or user selectable options). Therefore kwebhelper.py now supports commands coming from form elements inside a HTML file, if they follow certain rules.
The form must start with:
<form accept-charset="utf-8" enctype="application/x-www-form-urlencoded" method="get"
action="file:///homepage.html" name="anyname">
the first form element must be named "cmd" and begin with "formdata":
<input name="cmd" value="formdata" type="hidden">
Form elements whose names start with "quoted" result in quoted arguments
Form elements whose names start with "dquoted" result in quoted arguments using double quotes
Submit buttons should not have a name.

The default homepage.html file contains a simple example.

5) Document and media support

a) PDF documents

Although not part of any HTML specification, PDF has become the most important document format on the internet. Some browsers support it out of the box, but most need a special plug-in to view PDF documents inside the browser. There are no such plug-ins for the Raspberry Pi, but kweb supports a plug-in like behaviour. If you click on a link to a PDF document, it will be downloaded automatically and opened in a separate program.

Even links of the form “…pdf#page=17” are supported and will not only open the document but also navigate to a special page (17 in this example). This is a less known feature supported by the original Adobe Acrobat plug-in and a few other PDF plug-ins. I've created search engines for the web that give page specific search results and use this option to open PDF documents directly on the page matching the search result.

All PDF documents are saved in your “Downloads” folder (usually inside your home directory, but that can be changed in kwebhelper_settings). If you click the same PDF link a second time, kweb will discover that it has already downloaded the file and will open it directly. There's one disadvantage to this solution: you should delete files that you

don't need any more from your “Downloads” folder from time to time.

A standard Raspbian distribution has mupdf installed, but it is highly recommended to install the much more comfortable xpdf. Kwebhelper will use xpdf, if it is installed, and mupdf, if not.

b) Audio and video support

From your experience on desktop computers you may be used to watching video or listening to audio directly inside your browser. On the Raspberry Pi this is (currently) only possible to a very limited degree. The small ARM processor is simply not capable of decoding and displaying video content and needs the help of the GPU. The current webkit library which is used by kweb does not support hardware accelerated video, but that may change in the future. Until then, omxplayer is the only application that really makes use of the full potential of the Raspberrry Pi's GPU for video and audio content. Omxplayer is not an X-Windows application; it puts an overlay on the screen and the only way to control it is via keyboard commends. Kweb uses omyplayer for all kinds of media contents (audio, video, streaming links, web video, playlists), if not disabled from the toolbar. To get full keyboard control, kwebhelper has to start a terminal first, which also helps blanking the screen (but this can be disabled now for embedded applications, see chapter 8).

Video (or audio) from the internet can be available in three different ways:

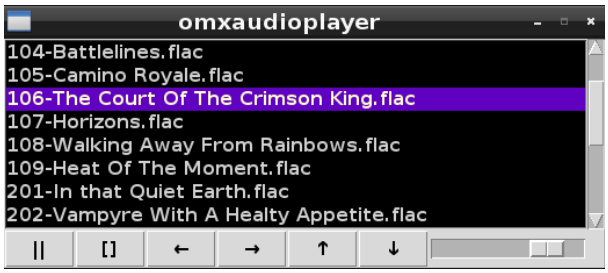
1. as video or audio or playlist file links or links to streams. If you click on such a link, kweb will play the file or stream or playlist using omxplayer, if it is not disabled in the toolbar.

2. embedded video using some kind of flash (or other) player (the old way and soon to be deprecated). There is no flash available for the Raspberry Pi and so these videos cannot be played directly. But if you click on the “Play” button in the toolbar, kwebhelper will use youtube-dl to try to extract the video URL and – if successful – will play the video full screen with omxplayer.

3. embedded video using the HTML5 video tags. Kweb may even try to play such video inside the browser but without much success. But if you click the “Play” button, kwebhelper will try to extract the video links and play the video with omxplayer. (You should stop the video display inside the browser first!)

This only relates to video; Audio files and playlists containing only audio files are handled a bit differently from video files. Playing audio on a black screen is usually not a good option. Since version 1.4 a small GUI has been built into kwebhelper, to manage audio playback with omxplayer, which is now the default method. But there are also other options possible: you can use VLC player, or omxplayer controlled from a terminal or run omxplayer completely in background. The details are described in chapter 8.

c) Omxaudioplayer



Buttons and their keyboard controls:

Play/Pause – Space, Return or Keypad Enter: will play the currently selected song, or pause or resume playing, when it is already playing.
Stop – ESC or q: will stop playing a song.
Rewind - ←: Jump backwards by about 10 seconds.
Forward -→: Jump forward by about 10 seconds.
Previous – ↑: If a song is playing and selected, stop it and play the previous song in the list. If a song is playing and you have selected another song, jump to that song and play it. If no song is playing, select the previous song and play it.
Next – ↓: If a song is playing and selected, stop it and play the next song in the list. If a song is playing and you have selected another song, jump to that song and play it. If no song is playing, select the next song and play it.

Volume control: move the slider to change the volume between -60 and +12 db. You can also click into the grey areas besides the slider to change the volume by 3 db, or use the '-' and '+' keys (also on the numeric keypad).

The keyboard commands are very similar to those used by omxplayer, except for the Up and Down arrow keys, which are not used for jumping ahead or backwards by 10 minutes, but for playing the previous or next song.

Functions inside the playlist window:

If you select another song, while a song is playing, it will be played, when the current song is finished or when you click on the Previous or Next button. If you double click another song than the currently selected one, it will be played (and a currently playing song will be stopped). If you double click the currently playing song, it will be stopped.

6) Command line options – fine tuning, part 1

kweb can be started from a terminal command line (or from a script) with an options string (and / or an URL to visit). The options string must begin with a "-", followed by a list of characters, and must be the first command line argument (an URL may follow as second). The following list explains each possible character:

a) Global program flags:
K = run browser in kiosk mode
A = use left ALT-key for keyboard commands in kiosk mode also
Z = enable full zoom on start instead of text zoom only, which is the default now
J = enable javascript (default is disabled)
E = enable cookies (default is disabled)
W = enable external downloads with wget (default is internal method)
Y = disable video and audio being played with omxplayer (by default it is enabled)
X = allow executables to be started from the "open" command
I = use only icons for the toolbar (default is to use both icons and text labels)
T = use only text labels for the toolbar
S = use small icons for the toolbar
P = disable private browsing (enabled by default)
G = use Google as search engine instead of using it via startpage.com (now default)
M = Don't maximize kweb window on start (except in kiosk mode).
H = use second command line argument, if supplied, as homepage instead of the usual homepage.html file. This can be either a file://... or http://... URL.
L = enable command link interface on http://localhost... which must be given as second argument like "http:localhost/" or "http://localhost:8080/". The path must end with a slash. Command links will have to add "homepage.html?cmd=..." to that argument. This is a security risk and should be used with care. Normally command links only work inside "file://..." URLs.
N = set this, if you want to use kweb in kiosk mode without a window manager. The default window size (see below) must match the screen resolution. Not everything may work as expected. Only suitable for some kinds of embedded applications (like digital signage, slideshows etc.).
F = disable special webkit features: plugins, OpenGL ES support, hardware accelerated compositing, which are now (since version 1.4) enabled by default. The current webkit library may not really support these options completely, but this may change in future versions.

b) Default window sizes
Set default window size (default value is 1920 x 1080). Especially useful (or needed) in combination with either "M" or "N". Use only one of these options:
0 = 640x400
1 = 768x576
2 = 800x600
3 = 1024x768
4 = 1280x1024
5 = 1280x720
6 = 1366x768
7= 1600x900
8 = 1600x1050
9 = 1920x1200

c) Keyboard commands
Each keyboard command you want to use, must be enabled by adding it to the options string (to be used together with ALT key except in kiosk mode, when the "A" flag is not set):
o = open file
b = back
h = home

r = reload page
p = play embedded video
+ = zoom in by 10%
- = zoom out by 10%
z = reset zoom to 100%
g = enable full zoom mode
t = text zoom only
j = enable Javascript
n = disable Javascript
e = enable cookies
d = disable cookies
w = enable external download method using wget
i = select internal download method
x = enable omxplayer for media
y = disable omyplayer for media
q = stop any running omxplayer instance (the hard way)
f = toggle full screen
c = close browser

You can use any combination you like, depending on the kind of embedded application you want to create.
BEWARE: if you run in kiosk mode without "c" or "f" enabled, you won't be able to close the browser any more!

7) Using Minimal Kiosk Browser without starting the desktop

The desktop of the Raspberry Pi can be a pain to use, because it is quite slow, especially with full HD resolution – at least compared to modern desktop computers. The Raspberry Pi is at its best, when you do one thing at a time and avoid to use programs that need lots of resources. Minimal Kiosk Browser can be used without entering the desktop mode. It still needs X and some kind of windows manager, but this is still much faster than a full desktop environment.

There are lots of applications for this. If you are mostly using the Raspberry Pi from the command line, but need a browser from time to time, this is the way to go. But you can do much more: create embedded applications with a browser window as their main interface – a media player, an educational environment with books and videos, a presentation system ….. - you can even use the browser window as desktop replacement, calling other programs through kweb's command interface, either from the URL entry text field or with its special command links or forms. In all these cases the homepage.html file will be your starting point and main interface. In kiosk mode its content will fill the whole screen and no window elements are shown. I'll describe two scenarios here: a simple one, especially suited for embedded applications and kiosk mode, and a more complex one supporting a multiple windows environment.

a) A minimal system, especially suited for embedded applications and kiosk mode

For this solution you will have to install matchbox-window-manager.
```
sudo apt-get install  matchbox-window-manager
```
Copy the file “kiosk” from the examples folder to the root of your user directory or create your own one (and make it executable). It may look like this:
```
#!/bin/sh
matchbox-window-manager &
kweb
```

This will start a full browser including interface inside a full screen window. For kiosk mode, the kweb line may look like this (with full keyboard support):
```
kweb -KAobhrp+-zgtjnediwxyqcf
```

If you want to open a special web site or local application server upon start, you can also add an URL:
```
kweb http://localhost:8080/
```
or (running in kiosk mode with full keyboard control and making the local web server your homepage):
```
kweb -KAHobhrp+-zgtjnediwxyqcf http://localhost:8080/
```

As a last example a simple Youtube watcher to be started from the command line in kiosk mode with a subset of keyboard controls, connecting youtube.com to your “home” command:
```
kweb -KAHZbhrp+-zjncf http://www.youtube.com
```

Then start kweb from the command line with:
```
xinit ./kiosk
```

b) A multi window system with the browser as a desktop replacement

For this solution you have to install “tint2”.
```
sudo apt-get install tint2
```
As window manager we use openbox, which is also used for the desktop and is already installed. This is less suited for kiosk mode and will also be somewhat slower than the matchbox-window-manager. Copy the file “kioskm” from the examples folder to the root of your user directory or create your own one (and make it executable). It may look like this:
```
#!/bin/sh
openbox --startup tint2 &
kweb
```

If you want to have the same look (theme, icons, font size – all set with the “Openbox Configuration” and the “Appearance” tools) as on your desktop, you can also use:
```
openbox --config-file /openbox/LXDE-rc.xml --startup tint2 &
```

Start this mode from the command line with:
```
xinit ./kioskm
```

At the bottom you will see a very simple task bar (tint2), which allows switching between windows, if you start other applications or a second browser window.

You can add some options to the kweb command, of course, and also an URL to got to a certain website directly.

For a start, you may use the homepage creator utility to create something like a desktop replacement homepage. But of course you can create a nice looking “Desktop” page yourself, e. g. with the WYSIWYG HTML editor Iceape Composer.

If you want your system to boot into your browser desktop, you have to edit two files (as root, e. g. with “sudo nano”):
1) /etc/inittab
Look for a line like that:
```
1:2345:respawn:/sbin/getty --noclear 38400 tty1
```
Put a “#” in front and add the following line after it:
```
1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
```

2) /etc/rc.local
Add the following line just in front of the line with “exit 0”
```
su -l pi -c "xinit ./kioskm"
```

If you changed the default user from “pi” to something else, you have to replace “pi” with your user name.

If you close the browser window, you will return to the command line.

8) kwebhelper settings – fine tuning part 2

As I have already explained above, kweb uses a python script named kwebhelper.py, to access other programs – a kind of plug-in interface. This is used to play audio or video files (also from m3u or pls playlists) with omxplayer, extract embedded videos from websites like youtube.com and play them full screen (using youtube-dl and omxplayer), download and display PDF documents (using mupdf or xpdf), execute commands (either from the URL entry line or from kweb's special command links and forms) and also to download files (using wget). You can fine tune its behaviour in many ways. But the default settings should work for most people out of the box.

There are a lot of configuration options for fine tuning kwebhelper.py. You will find them in the first part of the script. The same settings (except for one) are also found inside the script kwebhelper_settings.py, which is loaded when kwebhelper.py starts and overwrites the default settings. You may disable this behaviour (not use the settings script at all) by setting the first variable inside kwebhelper.py like that:
```
settings = ''
```
You can also move the kwebhelper_settings.py script to another place (e. g. in your user directory) and make it available with something like that:
```
settings= '/home/pi/kwebhelper_settings.py'
```
The default value is:
```
settings = '/usr/local/bin/kwebhelper_settings.py'
```
and that's where kwebhelper_settings.py will be installed.

There is one great advantage when using a separate settings script: if you do something wrong, when you edit it, kwebhelper.py will still work (only your new settings may not be applied). This is still a python script, so you need to be a bit careful not to violate the python syntax, when you modify it. But you don't have to learn python programming to edit it. The best way to do this, is using the python editor “idle”, because it can run a syntax check for you and warn you, if you made a mistake. But you can also use a text editor like leafpad or nano for editing. Call it this way from a terminal:

```
sudo idle /usr/local/bin/kwebhelper_settings.py
```

or replace “idle” with an editor of your choice. You can also add a command link to your homepage.html file, to call it directly from within your browser:

Edit kweb Config

Some things you should know, before you start editing:
Lines starting with a “#” are comments. The settings script contains some alternatives you can use, by simply removing or adding the '#' at the beginning of the line.
Quite a few options are Python lists of strings, using the following syntax:
['option1','option2','option3']
Sometimes a list is empty and looks like that: “[]”

If the Python syntax is new to you, you should always run “check” from Idle's run menu, which will show you all syntax errors.

Full list of settings, divided into sections (since version 1.3):

```
# GLOBAL OPTIONS
```

```
homedir = ''
```

This is the default setting. For downloads and also PDF documents and playlists your user directory in your main file system will be used (usually on your SD card). If you have a HD connected to your Raspberry Pi you may want to change this, to avoid writing to your SD card. Let's assume, your HD is mounted to /media/Volume1. Then change this setting to:

```
homedir = '/media/Volume1'
```

```
# OMXPLAYER AUDIO VIDEO OPTIONS
```

```
omxoptions = []
#omxoptions = ['-o','hdmi']
#omxoptions = ['-o','local']
```

This is a list of options sent to omxplayer, when it plays video files or streams. Some examples are given how to set it. You can use the default setting, or comment it out and activate another one by deleting the “#” in front or create your own list of options (see omxplayer documentation for details).

```
omx_livetv_options = ['--live']
```

The newer versions of omxplayer can treat live TV streams differently. You can set special options for such live streams here. To make use of these settings, add the start of your live TV stream links to the following list to enable live TV options like this:

```
live_tv = ['http://192.168.0.5:9082']
```

The default value is:

```
live_tv = []
```

which means, that live TV options are never used.

```
kill_omxplayer = True
```

Usually a running omxplayer program is stopped, before another one is run. This can be changed by setting

```
kill_omxplayer = False
```

e. g. to let music play on in the background when opening a video. This should be handled with care.

```
mimetypes = []
```

This is the default setting. If this is used, kweb will try to play all kinds of audio and video files (streams) with omxplayer. If you add mime types to the list, it will restrict, what will be played, to these mime types. If you set it, for example, to

```
mimetypes = ['audio/mp3']
```

only MP3 files will be played and nothing else. My recommendation: use the default settings. If omxplayer cannot play something, it will simply return.

```
omxplayer_in_terminal_for_video = True
```
Using this default setting, video played with omxplayer will always be started from a terminal (xterm) giving full keyboard control over the player. For embedded applications you may set this to:
```
omxplayer_in_terminal_for_video = False
```
and omxplayer will be called directy. You may have to set the '-b' option to blank the screen, or you can use the '--win' option followed by 'x1 y1 x2 y2' values to display video in a definite area of the screen. There is no keyboard control, but you can stop omxplayer from kweb with the (ALT+) 'q' command, if enabled on the command line, or with a special command link.

```
omxplayer_in_terminal_for_audio = True
```
This setting is relevant only, if both “useAudioplayer” and “useVLC” are set to “False”. Then omxplayer will be started from a terminal for audio files and playlistst containing only audio files and you can control it with the usual keyboard commands. Especially for embedded applications (like digital signage) you may want to set it to:
```
omxplayer_in_terminal_for_audio = False
```
Then audio will be played completely in background using omxplayer. There is no keyboard control, but you can stop omxplayer from kweb with the (ALT+) 'q' command, if enabled on the command line, or with a special command link.

```
audioextensions = ['mp3','aac','flac','wav','wma','cda','ogg','ogm','ac3',
'ape']
```
This list is used to detect audio files by their extension.
```
try_stream_as_audio = False
```
Usually links to audio or video streams are treated as video, if they have not file extension in their name. If you set this to “True”, they are interpreted as audio streams and will be played with one of the audio player solutions. This may be useful for some web radio streams. In this case the following list will be used to detect video files:
```
videoextensions = ['asf', 'avi', 'mpg', 'mp4', 'mpeg', 'm2v', 'm1v','vob',
'divx', 'xvid', 'mov', 'm4v', 'm2p', 'mkv', 'm2ts', 'ts', 'mts', 'wmv', 'webm']
```

```
useAudioplayer = True
```
Play audio files or playlists that contain only audio files in omxaudioplayer GUI. Set this to “False”, if you want to play audio from a terminal or in background.
```
omxaudiooptions = []
```
Options for omxplayer to be used when playing audio.

The following options apply to omxaudioplayer only:
```
defaultaudiovolume = 0
```
Volume setting when starting omxaudioplayer ranging from -20 to 4 (-60 to +12 db)
```
autoplay = True
autofinish = True
```
Start playing and close after playing last song automatically (if "True", set to "False" to disable)
```
fontname = 'SansSerif'
```
Font to be used for playlist and buttons
```
fontheight = 14
```
Value between 10 and 22; defines the font size of the text display and will also determine the size of the GUI window:
```
maxlines = 8
```
Number of entries displayed in playlist window at the same time (without scrolling), between 5 and 25:
```
lwidth = 40
```
A value between 40 and 80 defines the width of the playlist window and the player GUI. This defines the minimal number of characters of song names being shown (usually much more are displayed).
The last three values together define the size of the GUI window, ranging from a very small player window to almost full screen on a full HD monitor.

```
useVLC = False
```
If you set this to “True” (if you have installed VLC, of course) this takes precedence over all other audio settings and the playlist or audio file or link will be sent directly to VLC player. VLC will take three times as much CPU power as omxplayer, but it has a nice user interface and extended options.

#COMMAND EXECUTION OPTIONS

```
check_desktop = True
```
If this is set to “True”, which is the default value, most Desktop GUI applications will be detected automatically and not be run from a terminal. You don't have to add them to the list below then.

```
direct_commands = ['kwebhelper.py','omxplayer']
```
This is a list of programs that are called form a command link without opening a terminal first (as explained in the last chapter). You may want to add your preferred programs here, if they are not detected automatically as GUI applications or if you want to run a script in background, for example.

```
preferred_terminal = 'lxterminal'
```
If you call other programs from kweb, which are not in your direct_commands list (see above) or detected as GUI applications, kwebhelper will open a terminal first and run the command line inside the terminal. You can replace "lxterminal" by another terminal program, if you like, but the terminal program must support the "-e" (execute) option. Xterm may be the preferable alternative, as it has to be installed anyway.
```
formdata_in_terminal = False
```
The HTML form command options (new in version 1.3) may use quoted arguments, which are not passed on, when the called application is run from a terminal. So, by default, running such programs from a terminal is disabled. You can enable it, by setting it to "True", but that will not work with quoted arguments (you can still add your command to the direct_commands list to prevent it from running inside a terminal).

```
run_as_script = False
```
Commands are usually called from kwebhelper.py as a daughter process (which in turn is a daughter process of kweb). That means, that for each program started from kweb you have a Python interpreter as memory overhead. This can be avoided by setting this to "True", but there will be a script file created on disk each time a command is called. To avoid too many disk writes, this option is disabled by default.

```
# PDF OPTIONS
```

```
pdfprogpath = ''
pdfprog = ''
```
These settings belong together and must both be set. You could set them to use a PDF viewer of your choice, but only mupdf and xpdf have all the options that kwebhelper needs to set in the same way. If the strings are empty (default), the logic is as follows: if xpdf is installed, it will be used, if not, mupdf will be used, which is installed in Raspbian by default. So better don't change this. But if you want to use mupdf although xpdf is installed, you may set it to:
```
pdfprogpath = '/usr/bin/mupdf'
pdfprog = 'mupdf'
```

You can add options to the call of the PDF program (which the PDF program of your choice has to support, of course):
```
pdfoptions = ['-fullscreen']
```
The default setting is
```
pdfoptions = []
```
which means "no options"

```
pdfpathreplacements = {}
```
This is something very special, disabled by default. If you have a local web server on your Raspberry Pi, which includes a collection of PDF documents, accessing them from kweb would usually require to download them to your downloads folder first and thus create duplicates in your file system. To avoid this, you can use this option (a python dictionary). If your PDF files are served from "http://localhost:8073/Ebooks1" and they are placed in '/var/www/Ebooks1', for example, set it to:
```
pdfpathreplacements = {'http://localhost:8073/Ebooks1' : 'file:///var/www/Ebooks1'}
```
kwebhelper will then send the file paths directly to mupdf or xpdf instead of downloading the files to the "Downloads" folder in your user directory first. You can add more than one replacement path, separated by a ",".

```
# DOWNLOAD OPTIONS
```

If you use the external download mode with wget (selectable from the tool bar), you can set the following option to either "True" or "False":
```
show_download_in_terminal = True
```
For large, long running downloads it may be useful, to check their state and success (enabled by default).

```
# ONLINE VIDEO OPTIONS
```

```
preferred_html5_video_format = '.mp4'
```
kweb can extract videos embedded into the HTML5 video tag. Sometimes such tags contain more than one format for the same video. With this option you can select the preferred format.

Choose, if HTML5 video URL extraction is tried first and youtube-dl extraction afterwards or vice versa by setting the

following option to either “True” (default, because it takes less time to check for HTML5 video) or “False”:
```
html5_first = True
```
For a dedicated youtube application, for example, you might want to change this.

youtube-dl is used to extract the video URL from a web page (many more than just youtube.com), which then is used by omxplayer to play the video. Youtube-dl can be configured with a lot of different options, which you can set here:
```
youtube_dl_options = []
```
By default, no options are sent. To select certain preferred file formats, for example, you can set it to:
```
youtube_dl_options = ['-f','37/22/18']
```
Run “youtube-dl --help” for a list of possible options.
```
youtube_omxoptions = []
```
omxplayer options to be used for web video. To use the same options as for other video, set
```
youtube_omxoptions = omxoptions
```

9) Using the homepagecreator.py script to generate your homepage file

This is a simple python script that will generate a homepage.html file for you from a set of simple text files and a HTML template file. All text files (there are 5 categories) are really simple to understand. Samples are supplied for each which you can edit any way you like. If you don't want to use one category at all, remove the file from the tools folder or uncomment all text lines inside the file by setting a “#” in front of each line.

To use everything without any changes, you should move the tools folder (after unpacking it) to the root of your user directory.

All text files contain a list of definitions like that (and some comment lines starting with a “#”):
name=value
But the kind of value to use is different for some files. For the categories “Services” and “Internet”, “value” is a link like in the following examples:
My media server=http://localhost:8083
Google=https://www.google.com
My ebooks list=file:///home/pi/Ebooks/index.html

The idea of using two different categories is, that “Services” should be used for local stuff (files or servers), and “Internet” might be a replacement for the favourites function of other browsers, containing all web addresses that you frequently use to visit.

For the other three categories (“Applications”, “Tools”, “Configurations”), “value” is a command line string like in the following examples:
Terminal=lxterminal
Editor=leafpad
Synaptic=sudo synaptic
Atari800 emulator=sudo /home/pi/atari800/atari800
Reboot=sudo reboot
Recreate Homepage=/home/pi/tools/homepagecreator.py
Edit boot config=sudo leafpad /boot/config.txt
Edit kweb config (Idle)=sudo idle /usr/local/bin/kwebhelper_settings.py
Edit Internet=leafpad /home/pi/tools/Internet.txt
Edit Applications=leafpad /home/pi/tools/Applications.txt
Edit Homepage Template=leafpad /home/pi/tools/template.html

As you can see from the examples, it is even possible to edit the text files for the creation of your homepage.html from within your homepage itself and recreate it “on the fly”. If you want to add a favourite, call the “Edit Internet” command, add the link (with a name), save the file and call the “Recreate Homepage” command (by clicking the link).

These are just examples. You can do with it, what you prefer. If you don't like the way it looks - colours, font-sizes etc. -, open the file “template.html” and change them in the styles-section. You can also modify the python script itself, renaming the categories or adding more or whatever you like. It's all quite simple.

To finally create the homepage file, open a terminal:
```
cd tools
python homepagecreator.py
```
(or: ./homepagecreator.py)

And,.of course, you can create a nice looking homepage file with images and a neat layout, if you know a bit about HTML or by using a wysiwyg HTML editor like Iceage Composer.

10) A few tips and tricks – mostly about speed

You may have read in publications about the Raspberry Pi, that it is not very well suited for browsing the web. But this is not completely true, as it depends on the content of the website you visit. If it is built dynamically from lots of Javascript and / or contains any kind of animated graphic stuff, then displaying the website on the Raspberry Pi may really become slow. And don't expect your browser to play videos inside your browser – but does it really make a difference to play them full screen with omxplayer and return to the website afterwards?
You can control the behaviour of many websites to a certain degree. It's always a good idea, to disable Javascript and only enable it, if you really need it. A search page from Google, for example, loads almost three times as fast with Javascript disabled – and you won't notice any difference; in fact, Google serves a different page, when Javascript is not available.

The same applies to Youtube. It loads much faster with Javascript disabled, but does not look as nice – no video previews, but do you really need them? And there is another reason, not to use Javascript on Youtube and other pages, serving embedded videos: sometimes (depending on the video format), kweb will start playing the video inside the browser window (using javascript and HTML5), but this is really slow; if you then click on the “play” icon to play the same video full screen with omxplayer, it may block.

For some websites Javascript must be enabled; but a little bit of Javascript does not really slow it down. All that is the reason for the Javascript icon in the toolbar, so you can easily switch it off and on with one mouse click.

If HTML5 video tags are included into a website, kweb will try to play the videos (if autoplay is enabled), but this will be too slow to be of any use. Stop playback and click the “Play” icon in the toolbar instead; this will call the video extractor and play the video(s) with omxplayer.

Another problem, which is not only Rasperry Pi specific but a general webkit problem (at least of the version installed in Raspbian), exists with colour runs; when displayed with another zoom setting than 100% and full content zoom enabled your Raspberry Pi seems to freeze for ten seconds or more. This is the reason, why I introduced the new toggle button in version 1.3, to switch beween “Full zoom” and “Text only zoom”.

11) Application examples

a) A simple and fast media player system

I mainly use my Rasbperry Pi as a media player and that includes – for me – not only tv, videos and music but also old fashioned media like books. Do you know of any media player, that supports PDF documents? I know there are solutions like XBMC but I prefer something that gives me the freedom to use it more like a conventional computer system.

As a backend I use a media server written in python. It is based on a very old sourceforge project, “Edna”, and was only an MP3 music server in the original version. I've been using it for many years to access my music collection in my home network or over the internet when I was staying with friends. Then I started to modify it and over the years it became a full media server with lots of new features, which I call now “Edna++” and also a “Social Media Server”, because it can connect to other Edna++ implementations running on your friends' computers across the internet. It also detects Popcornhour media players automatically, supports their playlist format and creates special links to make use of their remote control. And it will run on any system where a Python 2.x version can be installed: Linux, Windows, Mac, many NAS devices etc. It does not need a database or any other software packages, is very light weight and low on resources and so well suited for the Rasperry Pi. Watching a HD movie on the Raspberry PI and serving two other HD movies across the network at the same time is not a problem at all.

Edna++ makes heavy use of m3u playlists, so you need a browser that can handle these play lists and send them to a media player. On the Raspberry Pi Minimal Kiosk Browser is it's ideal counterpart.

You can download a beta version from the following link:
http://steinerdatenbank.de/software/ednapp.zip

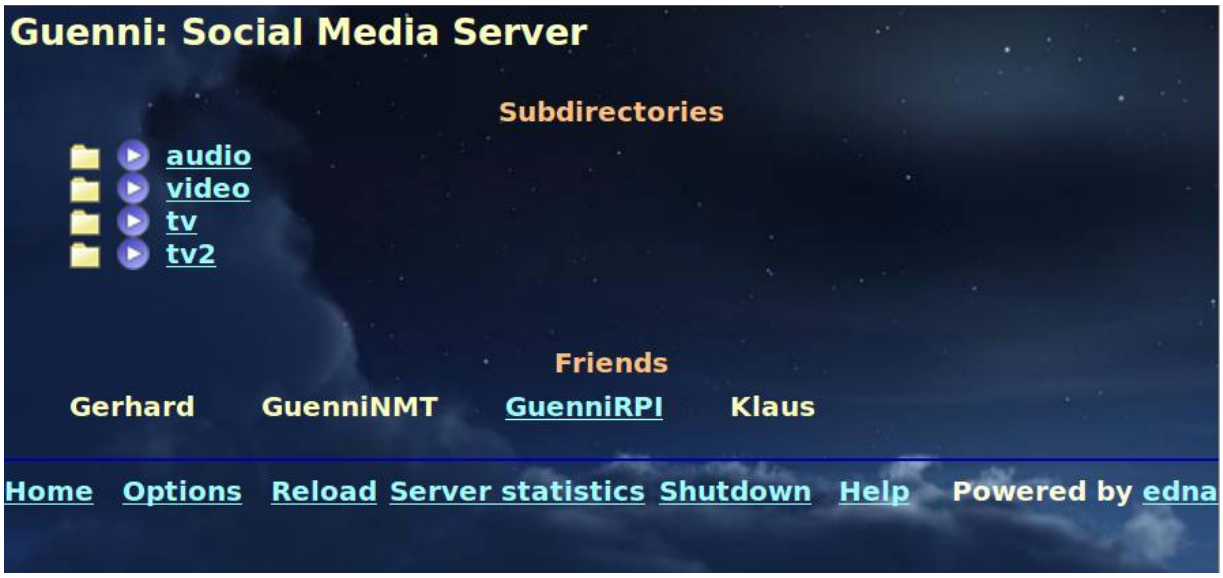
It's not really “beta”, but rather stable and has not been changed for some years, but I somehow never managed to really write a good documentation and also the “Help” function inside the program has not been implemented yet. That's why I call it a “beta”. After downloading it, unzip it in your main users directory, creating a folder “ednapp”. Open “edna.conf”with a text editor and work through it to adapt it to your system. There are lots of comments and examples.

You can start it from a terminal or script like that:

```
cd ednapp
python ednapp.py
```

Now add a link to your kweb homepage:
Social Media Server
(8083 is the default port, but can be configured in edna.conf), or call it with kweb directly or from a kiosk file:

```
kweb -options http://localhost:8083
```



You can modify the way it looks by selecting different templates and background images (or no background images at all, which is definitely faster on the Raspberry Pi). You can use it as a universal media server on any computer, of course, but for this project it is used as an interface to the media library on the Raspberry Pi itself, using Minimal Kiosk Browser as a frontend.

b) A digital signage or presentation system

I have included a simple example for a digital signage or presentation system. You will find it in the "slideshow" folder inside the "tools" folder. It consists of a number of HTML files and a "kiosks" file to start the (endless) show.

Timing and switching between pages is accomplished with the meta refresh tag in the header of the html files. The following example will display the content of the page for 5 seconds and then switch to "p002.html":
<meta http-equiv="refresh" content="5; URL=p002.html">

The examples will show you some tricks: how to display content from the internet (images, videos); how to start videos automatically; how to include commands (for example to include a youtube video into the show).

The "kiosks" file uses the following options:

```
#!/bin/sh
kweb -KAHNhcf file:///home/pi/tools/slideshow/p001.html
```

It does not use a window manager ("N" option) and connects p0001.html to the "home" function ("H" option). It requires a full HD monitor/TV. To run it, copy the "kiosks" file into your home directory, edit the "file:///..." path, and start it with:

```
xinit ./kiosks
```

c) Create your own applications – recommending "Karrigell"

If you've learned to program in python or want to learn it (and learning something like that is the main reason for the existence of the Raspberry Pi), you can create your own web applications and so extend what you can do with Minimal Kiosk Browser. There are lots of python web frameworks around. My favourite is the less known "Karrigell" package, especially in its 3.1.1 version for python 2.x. It's very easy to understand and use and comes with a good documentation and lots of examples. It has a built-in web server, so you don't have to install a separate web server (though it can run behind Apache2 or lighttpd, of course). By "simple" I don't mean it's only suited for simple stuff; in fact, I have used it

to create professional database driven solutions for companies as well.

On the Raspberry Pi I use it as an interface for watching Sat TV, for example. The web site simply shows a channel list and the web server application starts mumudvb to serve DVB streams and displays a link to watch the stream.

12) Finally – things that didn't fit anywhere else

a) Modify the software itself, if you like

Minimal Kiosk Browser comes with full source code (C and Python). I could create a deb package ready for easy installation, of course, but I prefer to do it this way. I want to invite people to apply their own changes. The code is small and simple: just a bit more than 800 lines of C code (and that includes a lot of empty lines in between) and the Python script has about the same size. Both are easy to understand and modify.

There's also an educational aspect to it. It demonstrates, that rather complex programs can be created with little effort, when we make use of existing libraries (webkitgtk in this case). We don't have to reinvent the wheel but use what other people already have created. One of the best things about Python is, for example, that for almost anything you want to do you will find some kind of package or module someone else has already written that you can use for your own project and you will get it for nothing. That's perhaps the most fascinating aspect of computer technology and software development: the whole world is working together here.

b) Using kwebhelper.py standalone

Although its main use is to help kweb connecting to other programs, kwebhelper.py can also be used from the command line, e. g. for playing m3u playlists with omxplayer. The syntax is:

```
kwebhelper.py command url [mimetype]
```

where command must be one of the following:

av	play audio or video or m3u playlist files or streams
pdf	download (if needed) and open PDF document with mupdf or xpdf
dl	download a file into the Downloads folder using wget
ytdl	check URL for embedded video and play it (using youtube-dl and omxplayer)

and url must be either a “http://” or “file://” link or a file path.

c) Known problems

If you use keyboard commands in kiosk mode without the “A” (Alternate) option, all keyboard command characters are not available for text input. If you need text input, you must set the “A” option.

When started from the LXDE Desktop you always get the small icon size for the toolbar icons. Which icons are used depends on themes and system settings and so the toolbar may look differently on different systems.

13) Changelogs

a) Changelog for version 1.4

1) New toggle button in the task bar:
Use Omxplayer
By default, this is enabled. If you disable it, kweb will try to play audio and video files and streams directly within the browser, using the gstreamer-library that libwebkit supports. Currently this will only work with some audio formats. Future webkit libraries, using gstreamer-1.0 with OpenMax support, may also play video with HW acceleration.

2) New audio player GUI for playing audio files and playlists containing only audio files with omxplayer:
"omxaudioplayer" has 6 buttons (play/pause, stop, rewind (about 10sec), forward (about 10 seconds), previous and next song), a volume slider, and a scrollable playlist window (for larger playlists).
Keyboard controls:
Space,Return or Enter to play/pause, ESC or q to stop, ← = rewind, → = forward, ↑ = previous, ↓ = next, + and - for volume control.

3) New (optional) keyboard commands (usually with ALT+):
q = stop any running omxplayer instance
x = enable omxplayer for audio and video
y = disable omxplayer for audio and video

4) New or modified command line options
F = Disable plugins and other exotic stuff (now enabled by default, CHANGED!)
Y = Disable the use of omxplayer upon start (by default enabled)

x,y,q for new keyboard controls (see above)

5) New or modified options in kwebhelper(_settings).py

The option "playaudioinbackground" is gone. It has been replaced by a few new options.
"omxoptions" will now be used only for playing video files or streams.
omxplayer_in_terminal_for_audio = True
if set to "False", audio can be played completely in background, without starting a terminal or a player GUI, depending on other settings (see below). To play audio on a web page automatically, include an (invisible) iframe, whose "src" points to an audio file or playlist.

omxplayer_in_terminal_for_video = True
if set to "False", video can be played completely in background, without starting a terminal first. You can then also use omxplayer's "-win" to simulate video playing inside the browser window. For full screen video you should use the "-b" option. Keyboard control of omxplayer is not possible then, but omxplayer can be stopped with ALT+q (if enabled on the command line). To automatically play video on a web page, include an (invisible) iframe, whose "src" points to a video file or playlist.

try_stream_as_audio = False
Normally stream links that don't have a file extension are handled as video by kweb. If you set this to "True", they will be handled as audio streams and use the audio settings. If the stream is a video stream will still be played as such, but using audio settings and maybe even the omxaudioplayer GUI. I added this option, because quite a number of m3u(8) files lists from web radio servers contain such streams. You should only set this to "True", if you want to play such audio streams inside a terminal or omxaudioplayer.

videoextensions
This is a list of video file type extensions. It's only used for counter checking, if you set "try_stream_as_audio" to "True"

useAudioplayer = True
If this is "True" (default), audio files and playlists containing only audio files will be played inside the new "omxaudioplayer" GUI. The setting "omxplayer_in_terminal_for_audio" will have no effect in this case. If this is set to "False", audio will be played inside a terminal or completely in background, if "omxplayer_in_terminal_for_audio" is set to "False".
One more thing: if "useVLC" is set to "True", all other audio options don't matter; VLC will be used to play audio (playlists) in this case.

omxaudiooptions = []
This list of omxplayer options (empty by default) will be used for all kinds of audio playback (except when using VLC). To use "pass through", for example, set it to ['-p'].

defaultaudiovolume = 0
volume setting when starting omxaudioplayer ranging from -20 to 4 (-60 to +12 db)

autoplay = True
autofinish = True
start playing and close after playing last song automatically (if "True", set to "False" to disable)
fontname= 'SansSerif'
Font to be used for playlist and interface
fontheight = 12
font size, value between 10 and 22, will also determine the size of the GUI window
maxlines = 8
number of entries displayed in playlist window of omxaudioplayer at the same time; value between 5 and 25
lwidth = 40
Minimal width of playlist entries in characters (usually much more are displayed); value between 40 and 80.

youtube_omxoptions = []
These is also a list of omxplayer options, to be used for all kinds of "web video", to be played when you click the "Play" buttons in the toolbar; this includes HTML5 video tags inside a web page and all websites using flash player, which are

supported by youtube-dl. If you want to use the same options as with "normal" video (files or stream links) you can set it to:
youtube_omxoptions = omxoptions

b) Changelog for version 1.3

1) Ported to GTK+3
kweb has been ported from GTK+2 to GTK+3 now. You won't notice a big difference (from the outside); some icons had to be exchanged and the file select box for the new "Open file" command looks and works much better now. But in the future that will help with things like Wayland support (the newest versions of GTK+3 and libwebkit have at least some support for Wayland built in).

2) New function and toolbar icon: "Open file"
Will open a file select box and if you have selected a file, it will be opened, if possible:
HTML, Text, JPEG, PNG, GIF files will be opened and displayed inside the browser
Audo and video files and m3u or pls playlists will be opened and run with omxplayer
PDF files will be opened with either xpdf or mupdf
If the "X" flag is set in a command line options string, executable files will be executed
Keyboard command: (ALT+) o

3) New toggle button in the toolbar to switch between full zoom mode or text zoom only, which is now the default. The background of this new setting is the fact, that webkit gets extremely slow (by a factor of about 10), if full zoom mode is enabled, zoom is not set to 100% and the web page contains color runs.
There is another side effect (on purpose): if you open a page in a new window, it is opened with 100% zoom, if full zoom is enabled, but in the current zoom setting of the old window, when it's disabled. This should prevent your system from freezing for a long time (often 10 to 20 seconds) when opening new browser windows.
Keyboard commands to switch: (ALT+) "g" (full zoom) or "t" (text zoom only).

4) Two different download methods to choose from with a new toolbar icon
a) internal download method of webkit (default). If you start a download, a message like "downloading: file name" will appear in the URL entry field. This method supports all kinds of cookies including session cookies. Files are always downloaded to the "Downloads" folder inside the user's home directory.
b) external download using wget (icon activated). Cookies are supported, but no session cookies. The download can be run in a terminal window to control its progress (depending on a setting in kwebhelper_settings.py). Files are downloaded to the "Downloads" folder inside the "homedir" defined in kewebhelper_settings (default is also the Downloads folder inside the user's home directory).
Keyboard commands to switch: (ALT+) "w" (wget) or "i" (internal).
Downloads can only be started from the right click menu!

5) The toolbar icons have been arranged in a new order: the "Play" button is placed left of the zoom buttons, so that all zoom functions (4 now) are grouped together.

6) startpage.com is now the default search engine, if starting a search from the URL entry line (with a "?" in front). startpage.com uses google search but prevents google from getting any information about you. I wanted to do this all along, but the damned colour run on the startpage.com website used to slow it down terribly. For this reason, full zoom is deactivated automatically, when you start a search and zoom is not set to 100%. Searching on Google directly instead can be enabled by adding "G" to a command line options string.

7) Support of embedded videos (HTML5 or flash players on all websites supported by youtube-dl) works without any fifo buffer now. Youtube-dl is still needed to extract the URLs, but omxplayer now plays the videos directly from those URLs. This approach works with some websites which didn't work before and it avoids buffering on the SD card. Youtube videos (only) seem to start later, but this is an illusion because omxplayer is started at a later time (less time to wait on a dark screen).

8) Support for PLS playlists (radio stations mostly) has been added.

9) Not only playlists containing only audio files but also single audio files can be played "in background" now (this is the default setting in kwebhelper), which means that omxplayer does not take over the screen. A terminal window is opened in which you can control omxplayer with keyboard commands, but you can put that in the background, of course, and listen to some music while you are doing something else. And optionally VLC can be used for that.
10) Javascript, cookies, full zoom and external downloads using wget are now disabled by default. They can be enabled at run time with the toggle buttons in the toolbar or with keyboard commands. They can also be enabled by command line flags in the options string.

11) The options string as first argument on the command line (starting with a "-") now uses capital letters for global options and lower case characters to enable keyboard commands and numbers for default window size (new!). As there have been lots of changes, check chapter 6 of this manual for details. All kiosk files you may have been using with earlier versions, have to be modified!

12) Extended command interface
Until now, commands could be executed from the browser either from the URL entry line putting a "#" in front or with special command links using the form:
href="file///homepage.html?cmd=cmdline", where is "cmdline" is the command line to be executed. Spaces have to be escaped with "%20". Some things were not possible with such links (URLs or file paths containing spaces as arguments, for example). And user selectable options were also not possible.

Now kwebhelper.py supports commands coming from form elements inside a HTML file, if they follow certain rules.
The form must start with:
<form accept-charset="utf-8" enctype="application/x-www-form-urlencoded" method="get"
action="file:///homepage.html" name="anyname">
the first form element must be named "cmd" and begin with "formdata":
<input name="cmd" value="formdata" type="hidden">
Form elements whose names start with "quoted" result in quoted arguments
Form elements whose names start with "dquoted" result in quoted arguments using double quotes
Submit buttons should not have a name.

The default_homepage.html file contains a simple example, which plays video in a small window area.

Two new options for command execution have been added in kwebhelper.py (see above). Now most GUI applications are detected automatically and are not run from a terminal without adding them to the direct_commands list. And optionally a script may be generated to execute a command to reduce the memory overhead of the Python interpreter for each application started from kweb (disabled by default, to avoid too many disk accesses).

13) The settings part of kwebhelper.py and kwebhelper_settings.py have been reorganized into subsections and a few new options have been added (and a few others removed). Use
```
sudo idle /usr/local/bin/kwebhelper_settings.py
```
to edit the settings (at run time)
All options are commented, quite a number can be set by commenting or commenting out the provided examples. Here is a list of the new options (for more details see chapter 8 of this manual):

AUDIO VIDEO OPTIONS
# special options for watching live TV streams (requires omxplayer 0.32 or later)
omx_livetv_options = ['--live']
# add the start of your live TV stream links to this list to enable live TV options
live_TV = []
# like this:
#live_TV = ['http://192.168.0.5:9082']
# set this to False, if you want to allow more than one omxplayer instance
kill_omxplayer = True

COMMAND EXECUTION OPTIONS
Some terminal programs seem to have problems with double quoted arguments following the -e (execute) option. Here you have the choice to run all commands coming from formdata without opening a terminal first.
formdata_in_terminal = False
#formdata_in_terminal = True

DOWNLOAD OPTIONS
If you use the external download method with wget, you may want to see what is happening, especially with large downloads:
show_download_in_terminal = True
#show_download_in_terminal = False

ONLINE VIDEO OPTIONS
# Choose, if HTML5 URL extraction is tried first (it's faster) and youtube-dl extraction afterwards or vice versa
html5_first = True
#html5_first = False
#additional youtube-dl options, e. g. selecting a resolution or file format

youtube_dl_options = []
#youtube_dl_options = ['-f','37/22/18']

14) A few small bugs and side effects have been fixed. The state of the toggle buttons is now set if you call the respective keyboard commands and they are also synchronized between different browser windows. And the annoying text messages in the terminal window when starting (web) videos has been made invisible.

c) Changelog for version 1.2

I've included an extractor for videos embedded into HTML5 tags. Usually libwebkit will try to play such videos, but far too slow. Clicking on the "Play" icon will now extract the video source links and play the video(s) full screen with omxplayer.
I've also enabled the"spacial navigation" feature of webkit, which provides fast navigation through form elements or links with the arrow keys. In embedded applications this can be used to control the browser content with an ir remote control (which has to issue key strokes). To make that visible on links, the focus style has to be set.

There are four new command line options:

'x' - use argument 2 as your homepage (bound to the "home" button). This can be a file:// or http:// link. If you create a script (or kiosk) file, that contains
kweb -x http://myhomepagelink
the browser will open that page when it starts and always return to it, when you click the "home" button. The command links won't work, if you connect to a server, because they can only be used from file links (but see below).

'z' - disable private browsing (which is by default enabled in Minimal Kiosk Browser)

The last two options are very special ones, only to be used in kiosk mode:

'n' - kweb can be used without a window manager; screen size is hard coded to 1920x1080 then. Not all options work as expected (keyboard control of omxplayer, for example), so this is only useful for special applications.

'l' - use the command links from a server running on localhost (only useful in combination with the x option). For example
kweb -k...xl http://localhost:8080/
will use this link as homepage and support command links of the form '<a href=http://localhost:8080/homepage.html?cmd=top" (to run the top command inside a terminal). This is risky from a security aspect and is only recommended for very special embedded applications.

d) Changelog for version 1.1

Two more toolbar icons: “Reload page” and “Enable/Disable Cookies”

Tooltips on all toolbar icons
Icon labels are now visible when started from Raspbian desktop

Different handling of videos embedded in web pages; it now supports all websites that can be accessed by youtube-dl (and not only youtube); it does not reload the page any more.
Command line now supports up to two arguments (options string and URL)

More options and keyboard commands:
interface options for the toolbar (icons and icon size, labels)
two different characters for switching Javascript or Cookie support on and off, instead of toggling them.

Support for calling programs (defined in kwebhelper_settings) as root (sudo) without opening a terminal

Separate settings file for kwebhelper.py, that can be edited at runtime (changing the settings does not require a new install any more)

Added homepagecreator.py script to create a homepage.html file with example configuration text files (in separate compressed tools folder)

Readme file has been replaced by this manual