Günter Kreidl

Minimal Kiosk Browser

Kweb 1.7.0

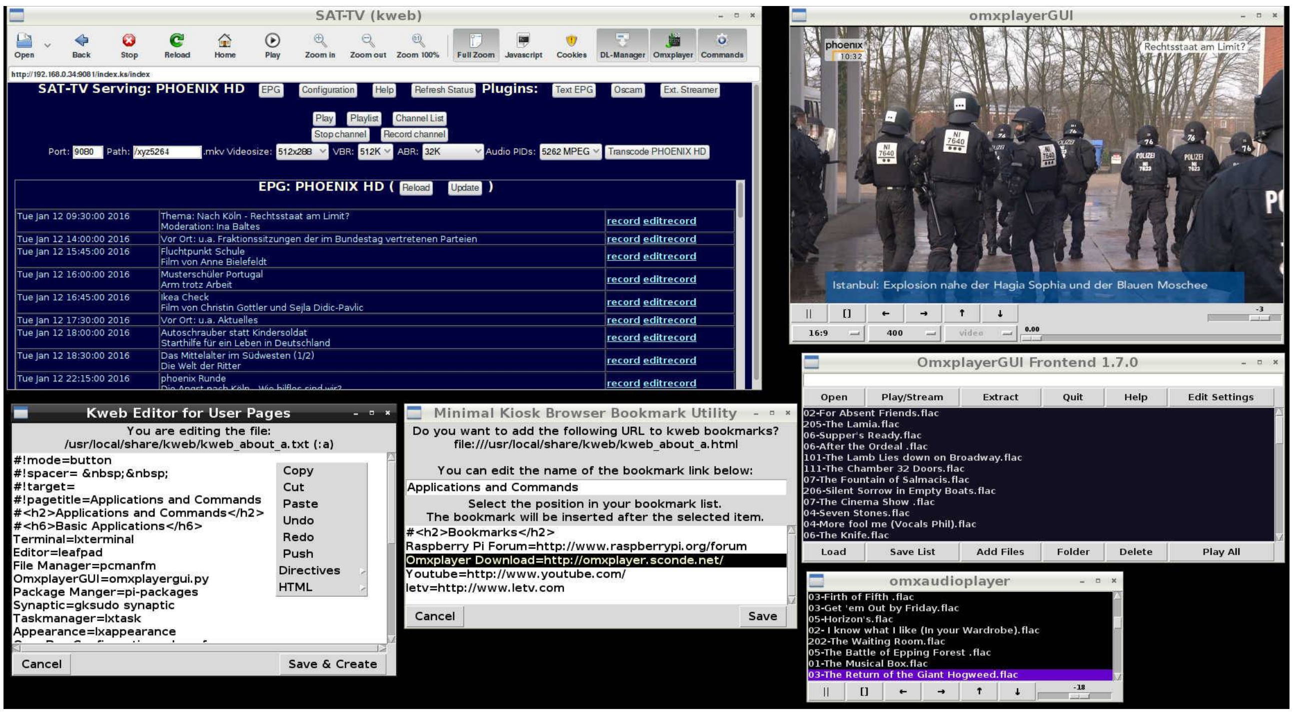
A Lightweight Browser and Application Interface for the Raspberry PI

7th Edition 2016

Content

Part One

What You Really Should Know About Kweb



http://192.168.0.34:9081/index.ks/index

SAT-TV Serving: PHOENIX HD

Port: 9080 Path: /xyz5264 .mkv Videosize: 512x288 VBR: 512K ABR: 32K Audio PIDs: 5262 MPEG

EPG: PHOENIX HD

Tue Jan 12 09:30:00 2016	Thema: Nach Köln - Rechtsstaat am Limit? Moderation: Ina Baltes	record editrecord
Tue Jan 12 14:00:00 2016	Vor Ort: u.a. Fraktionssitzungen der im Bundestag vertretenen Parteien	record editrecord
Tue Jan 12 15:45:00 2016	Fluchtpunkt Schule Film von Anne Bielefeldt	record editrecord
Tue Jan 12 16:00:00 2016	Musterschüler Portugal Arm trotz Arbeit	record editrecord
Tue Jan 12 16:45:00 2016	Ikea Check Film von Christin Gottler und Sejla Didic-Pavlic	record editrecord
Tue Jan 12 17:30:00 2016	Vor Ort: u.a. Aktuelles	record editrecord
Tue Jan 12 18:00:00 2016	Autoschrauber statt Kindersoldat Starthilfe für ein Leben in Deutschland	record editrecord
Tue Jan 12 18:30:00 2016	Das Mittelalter im Südwesten (1/2) Die Welt der Ritter	record editrecord
Tue Jan 12 22:15:00 2016	phoenix Runde Die Angst nach Köln - Wie hilflos sind wir?	record editrecord

You are editing the file:
/usr/local/share/kweb/kweb_about_a.txt (:a)

```
#!mode=button
#!spacer= &nbsp;&nbsp;
#!target=
#!pagetitle=Applications and Commands
#<h2>Applications and Commands</h2>
#<h6>Basic Applications</h6>
Terminal=lxterminal
Editor=leafpad
File Manager=pcmanfm
OmxplayerGUI=omxplayergui.py
Package Manger=pi-packages
Synaptic=gksudo synaptic
Taskmanager=lxtask
Appearance=lxappearance
```

Do you want to add the following URL to kweb bookmarks?
file:///usr/local/share/kweb/kweb_about_a.html

You can edit the name of the bookmark link below:
Applications and Commands

Select the position in your bookmark list.
The bookmark will be inserted after the selected item.

```
#<h2>Bookmarks</h2>
Raspberry Pi Forum=http://www.raspberrypi.org/forum
Omxplayer Download=http://omxplayer.sconde.net/
Youtube=http://www.youtube.com/
letv=http://www.letv.com
```

```
02-For Absent Friends.flac
205-The Lamia.flac
06-Supper's Ready.flac
06-After the Ordeal .flac
101-The Lamb Lies down on Broadway.flac
111-The Chamber 32 Doors.flac
07-The Fountain of Salmacis.flac
206-Silent Sorrow in Empty Boats.flac
07-The Cinema Show .flac
04-Seven Stones.flac
04-More fool me (Vocals Phil).flac
06-The Knife.flac
```

```
03-Firth of Fifth .flac
03-Get 'em Out by Friday.flac
05-Horizon's.flac
02- I know what I like (In your Wardrobe).flac
202-The Waiting Room.flac
05-The Battle of Epping Forest .flac
01-The Musical Box.flac
03-The Return of the Giant Hogweed.flac
```

This image shows all GUI components of the kweb package. Here kweb is used as frontend of my SAT-TV-Server application.

1. Introduction

Minimal Kiosk Browser (kweb & kweb3) is a slim and efficient web browser and application interface based on the webkit engine, written especially for the Raspberry Pi. It offers:

- HTML5 support
- PDF support via Evince, Xpdf or Mupdf
- asynchronous download using wget or the download manager uGet
- full media support (audio, video, playlists) using omxplayer
- omxplayerGUI, a window based front end for omxplayer
- web media support for HTML5 video or for websites supported by youtube-dl
- opening HTML, text and PDF documents, images (GIF, JPG, PNG) and all kinds of media directly from your file system
- a special command interface, which supports any kind of program from inside an html file
- lots of different fine tuning options
- a menu and a control panel page to access all configuration options in one place
- GPU acceleration for web video and image decoding in kweb3

Kweb can be used

1. as a browser for your desktop
2. to replace your desktop completely
3. as front end for embedded applications (kiosk mode)

The design principle of kweb was to keep it as simple as possible and to make the most of the limited resources of the Raspberry Pi (processing power and memory). Instead of inflating the program with lots of functions, it uses other, already existing programs for things it cannot do by itself. Nevertheless it offers some unique possibilities which you will not find in any other web browser. These features make it possible to use kweb as a replacement for your desktop or as a front end for embedded applications.

For embedded applications kweb can be started in kiosk mode and some of its normal features can be disabled to restrict its use for a special purpose. It has been successfully used for a number of digital signage and interactive kiosk systems (probably many more than I know of) and also has found its way into a bachelor thesis paper which describes how to use the Raspberry Pi for an information and TV solution in a university.

Although the main programs (kweb, kweb3 and omxplayerGUI) have been kept as simple as possible (and will remain so), they are highly configurable. Configuration and extension is realized by the kweb environment, which consists of a number of local web pages and a few additional utilities. These web pages make extensive use of kweb's unique application interface.

Kweb and kweb3 (written in C) and its helper programs (written in Python) are open source programs and you can download a separate source code distribution besides the Debian package distribution.

The complete documentation now consists of 3 documents:

The Kweb Manual (PDF, this one)

The OmxplayerGUI Manual (PDF)

The Kweb Changelog (HTML)

All three are installed together with the program and can be opened from inside the browser.

This manual is divided into three parts:

Part one describes the user interface and all basic functions. Don't expect to understand and really use kweb without reading it.

Part two has been written for power users who want to make the most of kweb and its environment.

Part three shows how you can create simple applications with kweb. For many things like creating your own desktop environment or a simple application interface you do not have to learn any programming language or know about HTML coding and so it's very well suited for beginners. More experienced software developers will find all the information they need to create web frontends for their applications in this part of the manual.

Note: This manual has been written for version 1.7.0 of kweb. Small changes in newer versions will be explained only in the changelog.

2. Installation and Usage

Install kweb from the command line or a terminal with the following commands:

```
wget http://steinerdatenbank.de/software/kweb-1.7.0.tar.gz
tar -xzf kweb-1.7.0.tar.gz
cd kweb-1.7.0
./debinstall
```

Note: The version number may change. You'll always find the latest version and installation instructions at: http://www.raspberrypi.org/forums/viewtopic.php?t=40860. Please use this forum thread for any kind of support questions and problem reports.

If you have already installed an earlier version of kweb (at least version 1.6.9 for Raspbian Jessie), you can always update to the newest version from inside kweb (built-in update function).

Kweb is now installed as a Debian package and needs the package installer gdebi. The installation script checks, if it is installed on your system and will install it, if it is not found. To remove kweb run one of the following commands from inside the installation folder:

```
./remove
./removeall
```

You can also remove it any time with:

```
sudo dpkg -r kweb
```

At the end of the installation process your system will be checked if programs needed by kweb are installed. Please check the output of this script carefully and install any missing programs.

Minimum requirements are:
omxplayer, youtube-dl, wget, mupdf or xpdf, leafpad, lxterminal

Except for youtube-dl, all these programs should already be installed on your Raspbian distribution.

Note: For best performance you'll need a special installation method for youtube-dl. To make things as easy as possible for you, you can run this installation directly from the application page in kweb's environment. It doesn't require more than a simple mouse click.

Installation of the following programs is recommended:
evince and/or xpdf for better PDF support
uget-gtk as GUI based download manager
tint2 as task bar, if you want to use kweb from the command line or as desktop replacement
xterm, if you want to use kweb's old method of playing videos full screen without a GUI or if you prefer a simpler terminal needing less resources
The following command line will install all programs at once:

```
sudo apt-get install evince xpdf xterm uget-gtk tint2
```

The installation of kweb will only take a few seconds. Afterwards you will find two new entries in the internet section of your applications menu:
Minimal Kiosk Browser
This will start kweb.
Minimal Kiosk Browser (GTK 3)
This will start kweb3

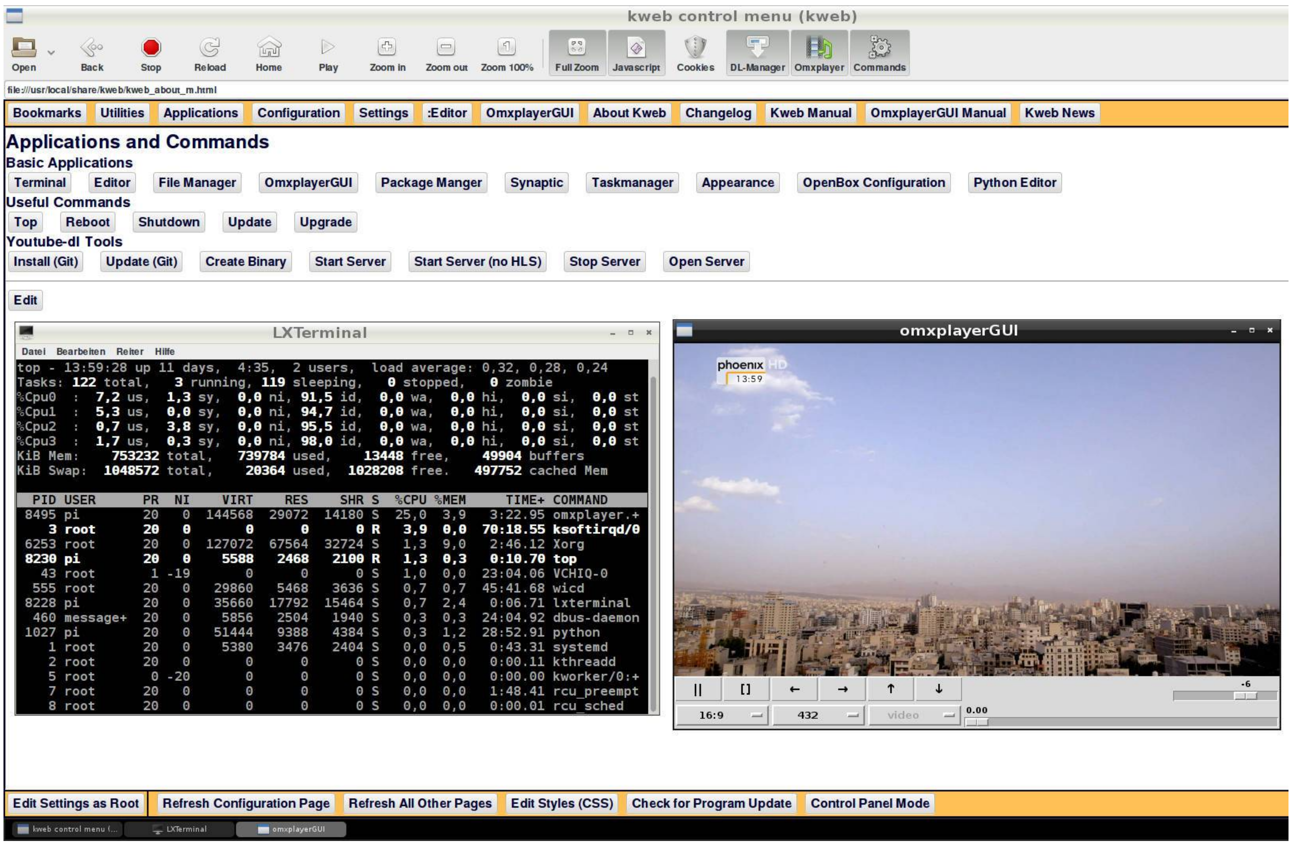
Both programs (kweb and kweb3) have identical functions, but they use different interface libraries and webkit engines:

kweb uses GTK+2 and the default webkitkgtk-1.0 engine of the stable Raspbian Jessie release. It's very stable and provides best compatibility.

kweb3 uses GTK+3 and the new hardware accelerated webkitgtk-3.0 engine introduced by the Raspberry Pi Foundation with the publication of the new web browser epiphany. Kweb3 can make use of all the new features like playing HTML5 web video inside the browser window, but will also suffer from all weaknesses of the new engine. Currently there are a number of bugs and they may even lead to crashes. But kweb3 will automatically profit from all epiphany updates in the future and so may become the preferred version when all the current bugs of the new webkit library are fixed. At the moment kweb3 disables hardware accelerated scrolling because this is the cause of a number of problems, and so should run more reliably than epiphany.

Everything said in this manual about kweb also applies to kweb3 (except where noted differently). Together with kweb, omxplayerGUI is installed as a separate application and you will find it in the media section of the application menu. Its usage is described in a separate manual.

Kweb can also be started from a terminal or script like this:
`kweb [options] [url]`
“options” is a single string starting with a “-” and followed by a number of characters, each one for a certain configuration setting. But this is only required, if you use kweb as front end for an embedded application and want to define its behaviour. Using the browser configuration page inside kweb is much easier and recommended for normal use. “url” can be any domain name or full file path or any valid “http://” or “file://” URL.
Replace “kweb” by “kweb3” to use kweb3.



```
top - 13:59:28 up 11 days,  4:35,  2 users,  load average: 0,32, 0,28, 0,24
Tasks: 122 total,   3 running, 119 sleeping,   0 stopped,   0 zombie
%Cpu0  :  7,2 us,  1,3 sy,  0,0 ni, 91,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu1  :  5,3 us,  0,0 sy,  0,0 ni, 94,7 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu2  :  0,7 us,  3,8 sy,  0,0 ni, 95,5 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
%Cpu3  :  1,7 us,  0,3 sy,  0,0 ni, 98,0 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem:    753232 total,   739784 used,    13448 free,    49904 buffers
KiB Swap:  1048572 total,    20364 used,  1028208 free.   497752 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 8495 pi        20   0  144568  29072  14180 S  25,0  3,9   3:22.95 omxplayer.+
    3 root      20   0       0      0      0 R   3,9  0,0  70:18.55 ksoftirqd/0
 6253 root      20   0  127072  67564  32724 S   1,3  9,0   2:46.12 Xorg
 8230 pi        20   0    5588   2468   2100 R   1,3  0,3   0:10.70 top
   43 root       1 -19       0      0      0 S   1,0  0,0  23:04.06 VCHIQ-0
  555 root      20   0   29860   5468   3636 S   0,7  0,7  45:41.68 wicd
 8228 pi        20   0   35660  17792  15464 S   0,7  2,4   0:06.71 lxterminal
  460 message+  20   0    5856   2504   1940 S   0,3  0,3  24:04.92 dbus-daemon
 1027 pi        20   0   51444   9388   4384 S   0,3  1,2  28:52.91 python
    1 root      20   0    5380   3476   2404 S   0,0  0,5   0:43.31 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:00.11 kthreadd
    5 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0:+
    7 root      20   0       0      0      0 S   0,0  0,0   1:48.41 rcu_preempt
    8 root      20   0       0      0      0 S   0,0  0,0   0:00.01 rcu_sched
```

If you want to start kweb from the command line (for casual use without starting the desktop or as a desktop replacement), run the following command:

```
xinit ./ktop
```

(Don't try to start it this way from a terminal running inside your desktop!)
This will start X-Windows, the openbox window manager, the small task bar tint2 and kweb. The image above shows an example of a desktop session with two programs running on top of (and opened from within) kweb.

Installing from source

```
wget http://steinerdatenbank.de/software/kweb-1.7.0-src.tar.gz
tar -xzf kweb-1.7.0-src.tar.gz
cd kweb-1.7.0-src
sudo make install
preparekweb
```

Note: The version number may change. You'll always find the latest version and installation instructions at: http://www.raspberrypi.org/forums/viewtopic.php?t=40860. Please use this forum thread for any kind of support questions and problem reports.

To remove it again, run

```
sudo make remove
```

or

```
sudo make removeall
```

The source distribution already contains the compiled binaries and you don't have to compile them first using 'make'. If you want to modify the source code and compile a new version you'll need to install some .-dev packages. Start with

```
sudo apt-get install libwebkit-dev libwebkitgtk-3.0-dev libwebkitgtk-dev
```

3. Kweb's User Interface

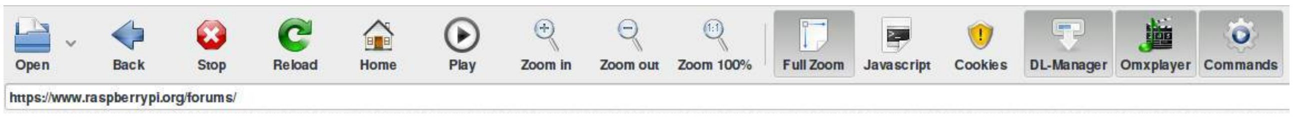
a) Introduction

Kweb's user interface is very simple: a toolbar and an URL entry field, nothing else. In version 1.7.0 a small menu has been attached to the first icon in the toolbar to give access to some functions which have only been accessible using keyboard commands so far. Everything you need is just a mouse click away.

Kweb also supports a rich set of keyboard short-cuts, usually called in combination with the left ALT key. When you move the mouse to any toolbar element a tool tip will be displayed that also shows the keyboard command(s) which can be used instead.

Nevertheless kweb and its helper programs are highly configurable using special web pages (similar to the “about:config” function of firefox and other browsers), the “Configuration” page for the browser itself and the “Settings” page for fine tuning the helper programs. Both pages (and many more) are available from kweb's menu page, which is also the default home page (if not set differently).

b) The Toolbar – All Icons Explained



Note: How the icons in the toolbar really look, depends on the icon theme you have installed and selected in lxappearance. The image above uses the “tango” theme which I like most.

The first 9 icons are command buttons; clicking them will result in immediate action.

“**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. All kinds of audio and video files, including m3u and pls playlist files, will be played with omxplayerGUI. PDF files will also be opened with either evince, xpdf or mupdf. And if the “X” option is activated in the browser configuration, all kind of executable files (scripts or binaries) are also opened and executed.

“**Back**” - go back to the previous web page.

“**Stop**” - stop loading the current page. You may know the annoying habit of many websites to continuously load more stuff (mostly crap like ads) which nobody needs and which will only slow down your computer's performance.

“**Reload**” - reload current page.

“**Home**” - display your home page. If you have a homepage.html file in the kweb directory “/usr/local/share/kweb”, it will be used as your personal home page. You can define any local HTML file or any web site as your home page in your browser configuration. If nothing is defined, kweb's menu page will be shown as your home page.

“**Play**” - if the web page currently displayed contains embedded video, clicking on this button will call omxplayerGUI, which will try to extract and play the video in a separate window or full screen. This will work with embedded HTML5 video tags and with all websites supported by youtube-dl. Youtube-dl needs a little while to extract the video address, but by using the new youtube-dl-server this time can be reduced to about 2 to 4 seconds. HTML5 video may also start playing inside your browser window, but you should immediately stop this when using the play command. Although kweb3 can also play video inside the web page itself, resolution and quality will be much better using omxplayerGUI. And it also works with many websites still using a flash player.

"**Zoom in**" - magnify content by 10%.

"**Zoom out**" - scale down by 10%.

"**Zoom 100%**" - display content in original size.

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

"**Full Zoom**" - zoom every element within the web page (full zoom), when enabled, or zoom only text elements.

"**Javascript**" - enable or 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. Often JavaScript is only used to open a lot of crap (ads) or to spy on you (did you know that you can be identified with a certainty of about 90% even if cookies are disabled?). A Google search page for example, will load twice as fast with JavaScript disabled without any loss of functionality.

"**Cookies**" - enable / disable support of cookies. Cookies are in many cases only used to spy on you. It's good practice to only enable them when you really need them, e. g. when you enter a web site with user name and password.

"**DL-Manager**" - if enabled, use uGet as download manager with a GUI (if installed), otherwise use wget for downloads. Downloads are disabled by default. Clicking on a link that cannot be displayed by kweb or its helper programs (media files or PDF), will never start a download. You have to right click a link and select "Download" from the pop-up menu to start downloading anything. All downloads go into the "Downloads" folder in your user directory, but this can be changed on the settings page.

"Omxplayer" - if enabled (default), omxplayer(GUI) will be used for all kinds of audio and video stream content accessible via links. If you disable it, the gstreamer-1.0 support built into the webkit libraries will be used for audio and video links and try to reproduce the media content inside the browser window.

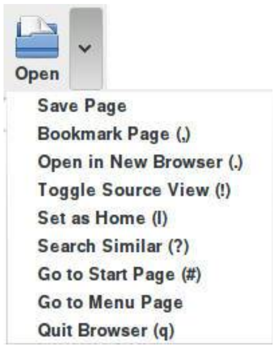
In kweb (webkit1) video playback is done by software only; a Raspberry Pi 3 is fast enough to play SD video in decent quality.

In kweb3 (webkit3) acceleration by the GPU is used and video up to 720p will play inside the browser.

Note: The gstreamer-1.0 players used by the webkit libraries are very buggy - the hardware accelerated version even more so than the pure software solution. Using them will sooner or later lead to memory corruption and crash the browsers. That's outside of my control and I cannot do anything about it but hope for bug fixing updates. Playing everything with omxplayerGUI is rock solid, delivers much better quality, supports more formats including playlists and is the method I recommend. And it works on all Raspberry Pi models.

"Commands" - if enabled, kweb's command execution interface is active and can be used to issue commands from inside web pages (files only). It's usually enabled, when the browser starts, whenever it is considered secure. For use in embedded applications it can also be explicitly enabled when starting the browser from the command line or a script.

c) Kweb's Single Menu



There's a small menu attached to the “Open” icon which gives access to some further functions:

“Save Page” - Saves the current web page including all elements (images etc.) to your downloads directory. This function uses wget with a special set of options and runs inside a terminal.

“Bookmark Page” - saves the current web page's URL as bookmark using kweb's small bookmark application which will let you edit the name and position in the bookmark list.

“Open in New Browser” - opens the current web page in a new browser instance. It will also close the current browser window, if it is not the only one. This is only useful on the RPi2, because then the new browser instance will run on a different processor core.

“Toggle Source View” - switch to source code view or back to normal view.

“Set as Home” - set current web page as home page for the current session (which is called when you click the “Home” button).

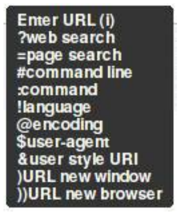
“Search Similar” - execute a web search based on the title of the current web page.

“Go to Start page” - open the page with which the browser was started (e. g. from a command line or another application); defaults to the home page.

“Go to Menu Page” - opens kweb's menu page (useful, if it is not your home page), which by default shows your bookmarks.

“Quit Browser” - closes all windows and quits kweb.

d) The Multifunctional Text Entry Field – URLs and Much More



The use of the text entry field seems obvious. Type a web site address into it and hit the “Return” key to navigate to it. “http://” will be inserted automatically, if it's missing and no “://” is found inside your text entry. If the URL starts with a “/”, it will be expanded to a “file://” URL.

Besides “http://”, kweb supports “https://”, “file://”, “rtp://”, “rtsp://”, “rtmp” and “mmsh://” type of URLs (not “ftp://”, as it is not supported by the engine, but many public ftp sites also support http access). And there's one more option: URLs starting with “file://~/” will be expanded to your user directory, e.g. “file:///home/pi/...”. Using a “file://” URL pointing to a directory will display the directory content inside the browser window.

While you are browsing (by clicking links), the entry field always shows the full URL of the currently displayed web page.
BTW, there is a nice keyboard short-cut that you may use when you want to enter a new URL. Pressing ALT+i will empty the entry field and activate it for text input (placing the cursor at the start of the field).

Auto-completion of a web address you are typing is beyond the scope of kweb. You will have to do all typing yourself.

So far the text entry field does exactly what you expect it to do and as you know it from any other browser. But there is much more to it:

Putting “)” in front of the URL will open the URL in a new window, putting “))” in front will open the URL in a new browser instance.

If you start a text entry with a question mark, your remaining text is used as argument for a web search. For example, entering
`?raspberry pi`
will search the web for “raspberry pi”. By default, kweb uses https://startpage.com as the web search site, which is nothing else than a completely anonymous search using Google. If you prefer Google to be spying on you, you can select this option in the browser configuration.

If you start a text entry with an equal sign, the currently displayed web page will be searched for the text following the equal sign, e. g.
`=raspberry pi`
will search the currently displayed web page for all occurrences of “raspberry pi”. Hitting the Return key repeatedly will jump to the next occurrences of your search terms.

There's another option, that comes very handy if you use kweb as a replacement for your your desktop. Putting a “#” in front will turn the text entry line into a command line.
`#leafpad`
for example, will open the text editor leafpad,
`#top`
will open a terminal and run the “top” command inside that terminal. Multiple arguments are also supported. Known desktop (GUI) programs are started directly, everything else will run inside a terminal.

But that's not all. Perhaps you know URLs like “about:config”, which are used by many browsers to access the browser's configuration page. In kweb this has been shortened to:
:c
which will open its configuration page. There are many more similar short commands and you can even extend them with your own “:”-commands. Besides “:c”, the following commands are predefined:
:s – open settings page for kwebhelper and omxplayerGUI
:m – open kweb's menu page, giving access to all other pages
:p – open kweb's control panel, showing all other pages at the same time
:o – open omxplayerGUI's web interface
:k – open “about kweb” info page
:b – open bookmarks page (editable)
:a – open application page (editable)
:u - open utilities page (editable)
:e – open the “:Command” editor, which assists you in creating your own “:command” pages and keyboard short-cuts.

There are four more kinds of entries to modify browser settings:

Since version 1.5.1 support for spell checking is enabled for all kinds of web text input. By default, your current system language will be used. You can change that at run time by entering a command like this in the URL entry field:
`!en_US`
(to switch to US English, for example). Multiple language support is also possible like this:
`!en_US,de_DE`
You must have the matching aspell/myspell/hunspell dictionaries installed; otherwise it won't work.

Language codes are either two or three letter codes, followed by an underscore and a country code (en, en_GB, en_US, de_LU etc.). Install enchant-lsmod and run:

```
enchant-lsmod -list-dicts
```

to get a list of available spell checking dictionaries on your system and their language codes.

By default kweb is set to use "utf-8" as default encoding, which is used when a web page does not contain a proper encoding tag. Using

```
@encoding
```

with a proper encoding value (see appendix), will set a new default encoding. This may be helpful when loading text pages, which have no encoding information, e. g.

```
@windows-1252
```

will select the well known "latin-1" encoding used by many Windows text files. A simple "@" without any value will restore the default "utf-8" encoding.

The "$" sign followed by some text will set this text as user agent which kweb will send to web pages. Sometimes a website won't let you enter when you are not using firefox or chrome. It's also possible to simulate a mobile browser this way. For example,

```
$Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:43.0) Gecko/20100101 Firefox/43.0
```

will simulate a rather recent Firefox version coming from an i686 Ubuntu PC.
A simple "$" followed by nothing else will restore kweb's default user-agent string.

One last option entry begins with an "&" followed by a file URI to a local CSS file, which can be applied as user style sheet to all web pages to modify their view, e. g.

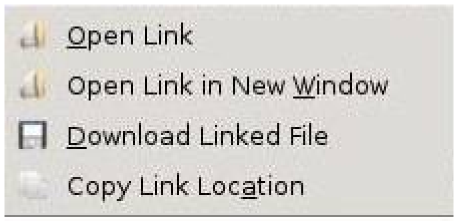
```
&file:///usr/local/share/kweb/color.css
```

will apply a user style sheet that is installed as an example with kweb. You may find similar style sheets on the web, e. g. at http://userstyles.org.
To remove the user style sheet again, enter a single "&" followed by nothing else and hit the return key.

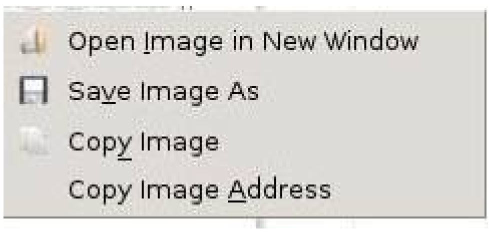
Note: All these text entries can also be executed by kweb's application interface using special links or buttons. Kweb's "Utilites" page contains a number of examples. In parts two and three you will learn how you can easily extend these examples with your own commands.

e) Popup Menu Functions Inside the Browser Window

Right clicking on a link will offer you the following options inside a small pop-up menu:

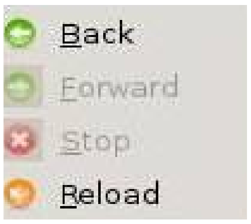


"Open Link" - same as left click
"Open Link in New Window" - open link in new window
"Download Linked File" - download the content pointed to by the link
"Copy Link Location" - copy it to the clipboard, ready for pasting it somewhere else



Right clicking on an image opens another pop-up menu with the following options:
"Open Image in New Window"
"Save Image As" - download the image
"Copy Image" - to the clipboard
"Copy Image Address" - to the clipboard

Right clicking into an "empty area" of a web page opens another pop-up menu with the following options:
"Back" - go back to previous page (or frame)
"Forward" - go forward in history (if possible), applies to a frame also
"Stop" - stop loading a page (or frame)
"Reload" - reload page or frame



f) Additional Keyboard and Mouse Modifier Functions

There are a few functions that are only available via keyboard short-cuts:

ALT+f: toggle between full screen and window mode

ALT+k: toggle between hiding or showing the toolbar and URL entry field.

By going to full screen mode and hiding the toolbar you are in fact switching to kiosk mode.

ALT+m: toggle between maximized and not maximized window.

ALT+c: Close current window; this will also close the browser if there is only one window.

And there is one more keyboard short-cut that is really useful when you want to enter any type of text into the entry line:
ALT+i will clear the entry line and put the cursor in place, ready for typing.

Using the mouse with modifier keys:

Left clicking on a link while holding the SHIFT key down, will open the matching URL in a new browser instance. On a Rpi 2/3 this will run on another processor core.

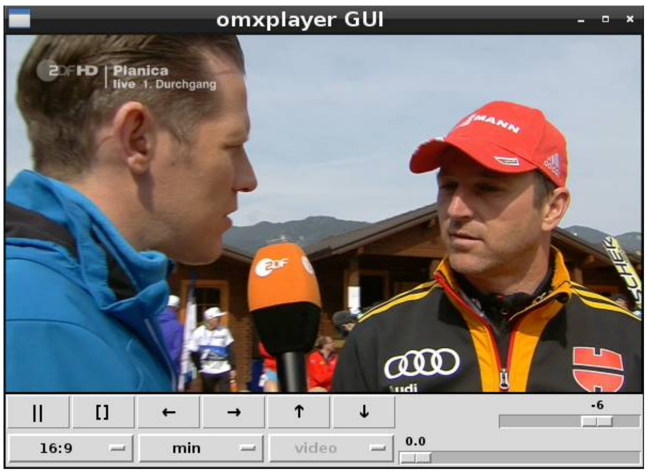
Moving the mouse scroll wheel while holding the CTRL key down will zoom in (forward move) or out (backward move). Each "click" of the mouse wheel will increment or decrement the zoom by 10%, so use it with care. This is limited to 50% minimal and 200% maximal zoom size.

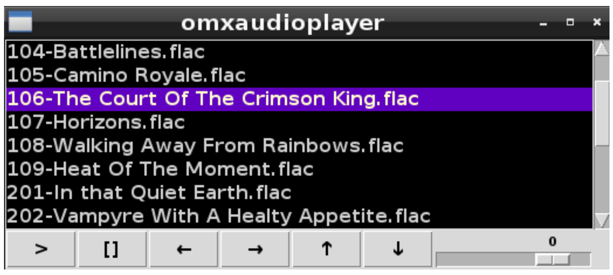
4. Media Support

Playing video inside a browser, as you may be used to on powerful desktop computers, tablets or even phones, is impossible with the small ARM processor of the Raspberry Pi without help from the powerful GPU. This has changed a bit on the RPI2 or 3, but it takes a highly optimized code to display video without the support of the GPU. In the beginning there was no browser engine that could use the GPU to play video inside the browser window, but that has changed now thanks to the efforts of the Raspberry Pi Foundation and the people from collabora.com. Kweb3 makes use of that new engine and so can play most HTML5 web videos directly inside the browser.

This is not possible with kweb (using the old engine), and it is also not possible for websites using those horrible flash based video players, which hopefully soon will vanish completely and be replaced by HTML5 video.

But both kweb and kweb3 can make use of omxplayer, the only media player for the Raspberry Pi with full support for the GPU. It's still unsurpassed by any other method regarding quality and use of resources. The gstreamer-1.0 OpenMax plugin used by the new webkit3 engine (and kweb3) is far less efficient and so you may still prefer to use omxplayer even in kweb3. For comfortable use of omxplayer, kweb uses omxplayerGUI, which can play all kinds of media content (audio, video, playlists) either within a window or full screen.




omxaudioplayer

104-Battlelines.flac
105-Camino Royale.flac
106-The Court Of The Crimson King.flac
107-Horizons.flac
108-Walking Away From Rainbows.flac
109-Heat Of The Moment.flac
201-In that Quiet Earth.flac
202-Vampyre With A Healty Appetite.flac

Each time you click a link pointing to some kind of media content (a file or a stream), omxplayerGUI is called automatically to play that content (if supported by omxplayer), and of course omxplayerGUI can play all kinds of media directly from your file system when you select them with the “Open” command from the toolbar. OmxplayerGUI can also extract video URLs from many websites, using its own HTML5 extractor or youtube-dl for lots of websites using flash based players, and play these videos. This always happens when you click the “Play” button in the toolbar on appropriate web pages.

OmxplayerGUI is installed as a separate application together with Minimal Kiosk Browser and can also be used standalone. Therefore it comes with its own manual. Look there for further details.

It should be noted that kweb's old method of playing video, always full screen without any GUI elements, is still supported. There's even a preset “nogui” for this mode installed on your system which can be activated with a mouse click. Embedded applications (presentations, for example) will often prefer this method, which also allows tricks that make it look like a video is running inside the browser window.

5. PDF Support

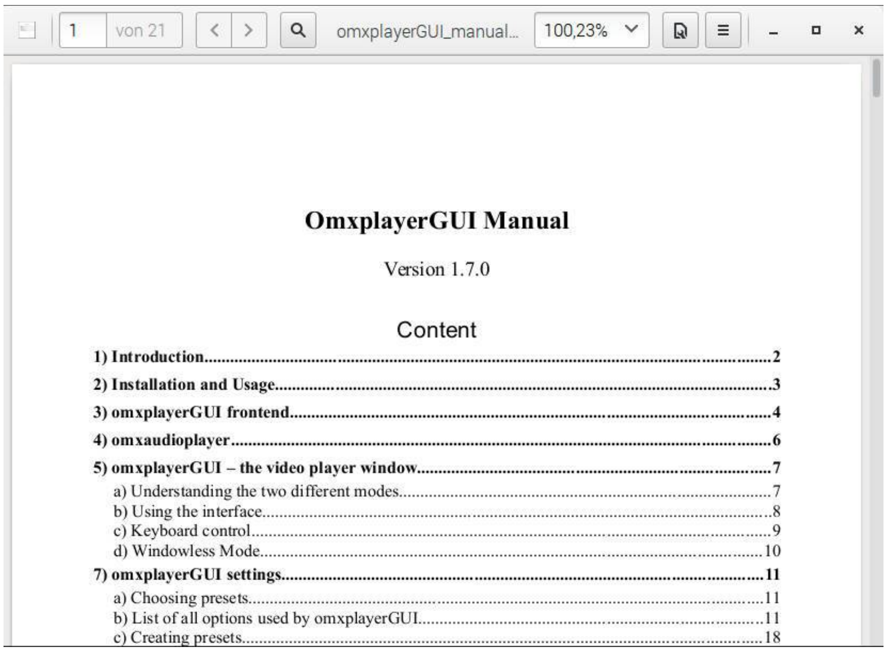
Although not part of any HTML specification, PDF has become the most important document format on the internet. Some browsers support it out of the box, but most need a special plug-in to view PDF documents inside the browser. There are no such plug-ins for the Raspberry Pi, but kweb supports a plug-in 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 on the settings page). 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.

Kweb supports three different PDF programs: mupdf, xpdf and evince. Kwebhelper will use evince, if it is installed, then it will look for xpdf and use mupdf as last resort. You can select the PDF program you want to use on the settings page.

Evince takes a bit longer to load, but gives the "smoothest" PDF experience.



OmxplayerGUI Manual

Version 1.7.0

Content

6. Download Manager

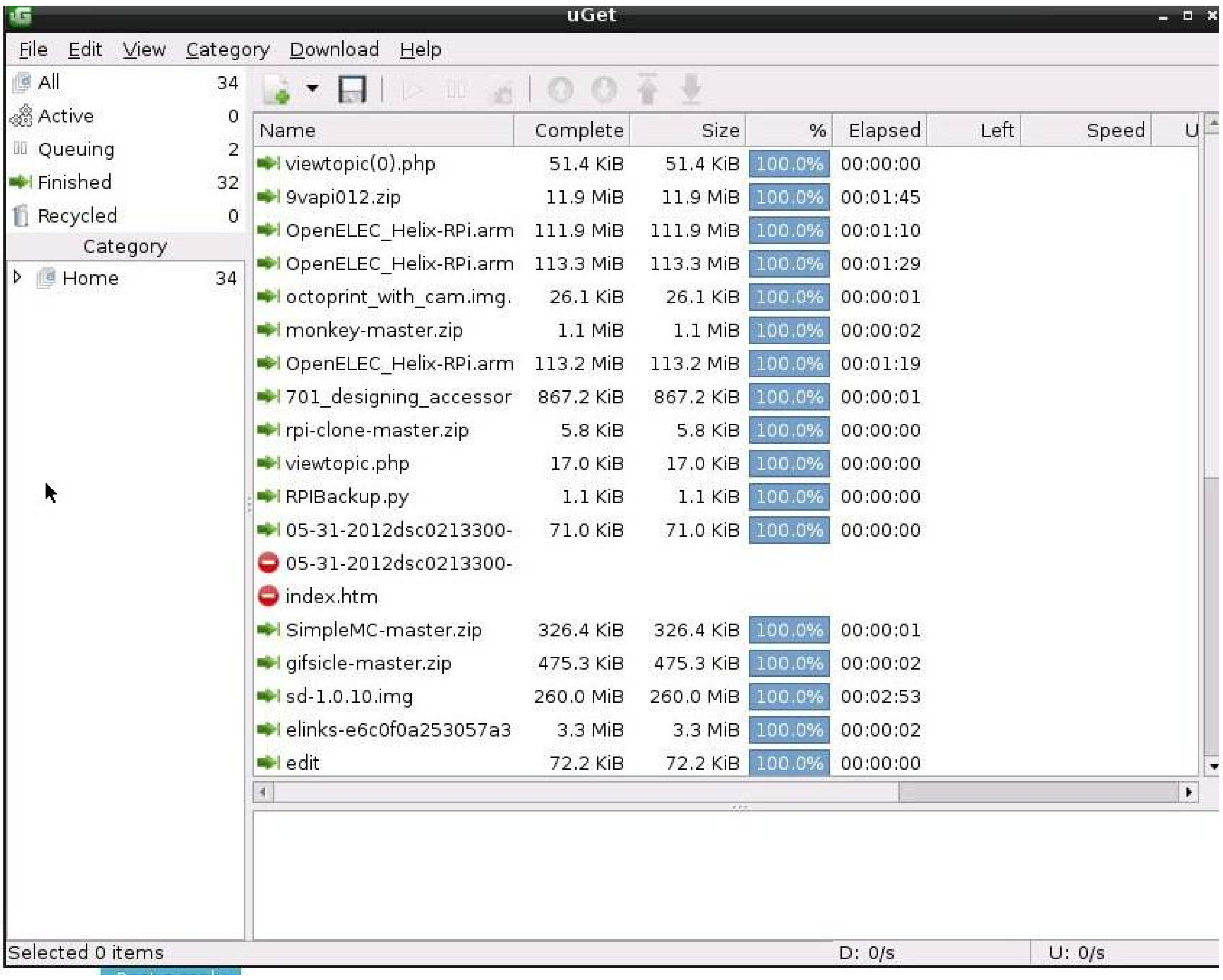
Downloading files of any kind from the internet (using the right click pop-up menu) is also managed by external programs, either the command line tool “wget” (running in a terminal or in the background, depending on your settings), or the comfortable download manager uGet, which offers a lot of options and has a nice GUI.

If uGet is not installed, wget will always be used. Otherwise you can switch between using wget and uGet with the toggle button “DL-Manager” in the toolbar.

For both programs you can set special options on the settings page. The predefined options should be fine for most usages.

All Downloads go by default into the “Downloads” folder in your home directory, but you can set a different destination on the settings page.

Both wget and uGet can use the cookie library of kweb. If a download requires cookies to be allowed, you can enable them in kweb and they will also be used by both download applications.



All	34
Active	0
Queuing	2
Finished	32
Recycled	0
Category	
Home	34

Name	Complete	Size	%	Elapsed	Left	Speed	U
viewtopic(0).php	51.4 KiB	51.4 KiB	100.0%	00:00:00			
9vapi012.zip	11.9 MiB	11.9 MiB	100.0%	00:01:45			
OpenELEC_Helix-RPi.arm	111.9 MiB	111.9 MiB	100.0%	00:01:10			
OpenELEC_Helix-RPi.arm	113.3 MiB	113.3 MiB	100.0%	00:01:29			
octoprint_with_cam.img.	26.1 KiB	26.1 KiB	100.0%	00:00:01			
monkey-master.zip	1.1 MiB	1.1 MiB	100.0%	00:00:02			
OpenELEC_Helix-RPi.arm	113.2 MiB	113.2 MiB	100.0%	00:01:19			
701_designing_accessor	867.2 KiB	867.2 KiB	100.0%	00:00:01			
rpi-clone-master.zip	5.8 KiB	5.8 KiB	100.0%	00:00:00			
viewtopic.php	17.0 KiB	17.0 KiB	100.0%	00:00:00			
RPIBackup.py	1.1 KiB	1.1 KiB	100.0%	00:00:00			
05-31-2012dsc0213300-	71.0 KiB	71.0 KiB	100.0%	00:00:00			
05-31-2012dsc0213300-							
index.htm							
SimpleMC-master.zip	326.4 KiB	326.4 KiB	100.0%	00:00:01			
gifsicle-master.zip	475.3 KiB	475.3 KiB	100.0%	00:00:02			
sd-1.0.10.img	260.0 MiB	260.0 MiB	100.0%	00:02:53			
elinks-e6c0f0a253057a3	3.3 MiB	3.3 MiB	100.0%	00:00:02			
edit	72.2 KiB	72.2 KiB	100.0%	00:00:00			

Selected 0 items D: 0/s U: 0/s

7. The Kweb Environment

a) Introduction

The kweb environment consists of a number of local web pages which make extensive use of kweb's application interface and so provide a high level of interactivity and communication with kweb's helper programs or any other software installed on your Raspbian system.

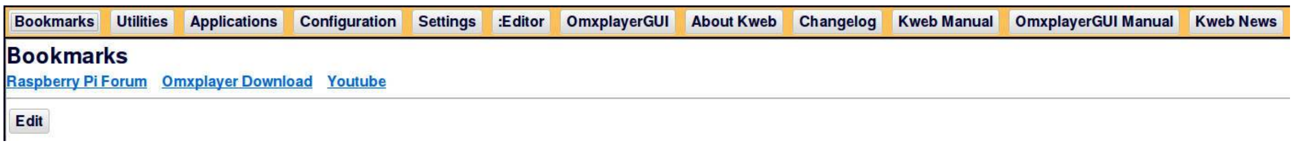
Another part of the kweb environment is a set of utility programs; some of them are only used in the background, a few others have their own GUI and will be called when needed (omxplayerGUI, kweb's bookmark and text utilities).

Access to the complete environment is possible from kweb's menu and and control panel pages which are described in the next chapter.

b) Kweb's Menu and Control Panel Pages

If you have not created your own homepage.html file and have not configured another home page and/or start page in the browser configuration, kweb will start with its menu page and also use it as its home page (when you click the home button). It's also available from kweb's menu or by typing ":m" into the entry field.

This page will give you access to all predefined pages (one at a time), the documentation, and also offer a few special functions. The upper part looks like this (when opened, with the bookmarks page as default content):

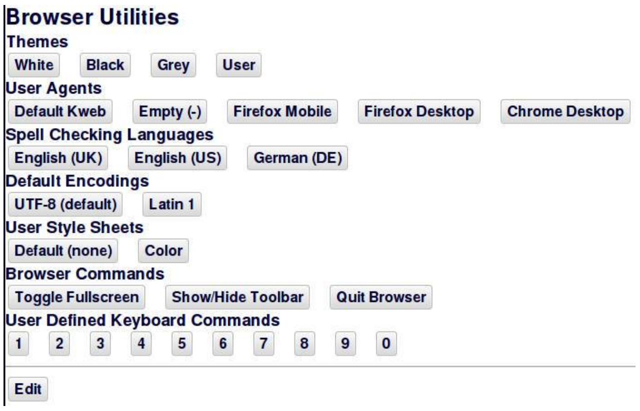


Clicking on one of the first 9 menu buttons will load the selected page into the frame below the menu bar. Clicking on one of the following two buttons will open either one of the manuals with your preferred PDF application. And the last button will display a kweb news page coming from the internet.

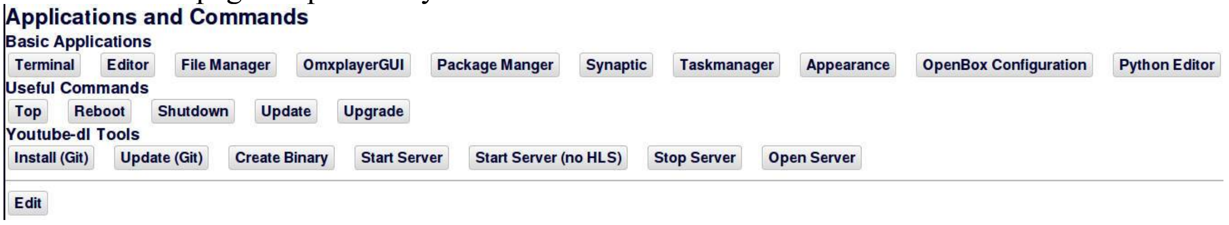
Details about most pages will be explained later on and also how you can use them to change your configuration, or to edit some pages, like the bookmarks page for example. Here's just a short overview:

Bookmarks: You can add bookmarks directly from your browser, which will use kweb's bookmark utility. This is a user editable page. In part two you will learn how to edit it.

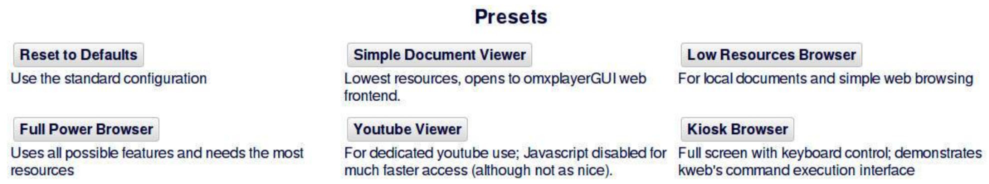
Utilities: This page lets you change a lot of browser and environment settings in real time and also provides access to user definable keyboard commands as buttons. This is a user editable page. In part two you will learn how to edit it.



Applications: This page demonstrates kweb's ability to start other programs from inside a browser window. At the bottom you'll find some buttons to help you with installing and updating the special youtube-dl version kweb needs to play web video with omxplayerGUI with best efficiency, and also to start and stop the youtube-dl-api-server, which speeds up access to web video even more. This is a user editable page. In part two you will learn how to edit it.



Configuration: This is the browser configuration page which allows fine tuning of the browsing engine and the browser's start up behaviour. A detailed description will be given in part two of this manual. For now let's just have a look at the bottom part of the page:



Use the standard configuration

Lowest resources, opens to omxplayerGUI web frontend.

For local documents and simple web browsing

Uses all possible features and needs the most resources

For dedicated youtube use; Javascript disabled for much faster access (although not as nice).

Full screen with keyboard control; demonstrates kweb's command execution interface

Clicking on one of the buttons lets you select one of six predefined configuration presets. You have to restart kweb(3) to use the new configuration.

Settings: This page lets you set lots of common fine tuning options, which are used by kweb's helper programs, especially kwebhelper.py (responsible for PDF, downloads and command execution, described in part two of this manual) and omxplayerGUI (a detailed description is given in the omxplayerGUI manual).

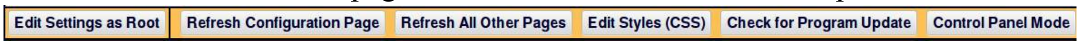
:Editor: This page helps you to manage user editable ":Command" pages, to create your own extensions and to define the behaviour of user definable keyboard commands. A detailed description is given in part two.

OmxplayerGUI: a simple web interface for omxplayerGUI; also contains a full list of keyboard commands used by the media player.

About Kweb: A typical "About" information page for the current kweb version.

Changelog: Displays the complete changelog of kweb starting from version 1.0.

At the bottom of the menu page is another menu bar with a few special functions:



Here is a short explanation of these commands, although many details will be explained (and better understood) later on:

"Edit Settings as Root": will open another kweb instance with a special configuration as root and open the settings page for the helper programs (kwebhelper and omxplayerGUI). You will better understand later on, why editing them as root simplifies things.

"Refresh Configuration Page" and **"Refresh All Other Pages"**: all configuration pages and all editable pages (like the bookmarks page) have to be recreated after changes have been applied. In most cases this happens automatically, but sometimes you need to call the refresh manually.

“Edit Styles (CSS)”: all predefined (and user created) pages are styled with a common CSS file. If you want to edit font sizes, colours etc. of the “user” style sheet, you can do that with this command if you know a bit about CSS styles. Leafpad will be opened (as root) to edit the CSS file.

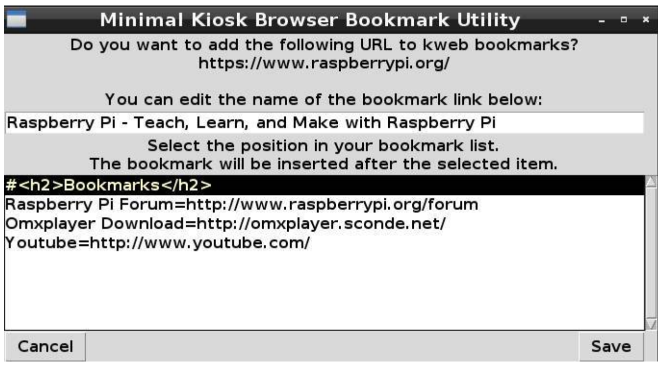
“Check for Program Update”: This checks if a program update is available. A terminal will open and either display the message “your version is up to date” or announce the availability of a new version, which can be downloaded and installed directly from within that terminal,if you agree to do so. All future updates will be available in this way.

“Control Panel Mode” will open the control panel. This offers almost the same functionality as the menu page, but with everything on one web page. It takes a bit longer to load (in fact, it loads 8 pages at once). The control panel can also be opened by entering “:p” into the entry line.

c) Kweb's Helper Programs

The kweb package contains a number of helper applications, mostly written in Python. Most of them are only used in the background, but besides omxplayerGUI (described in its own manual) there are two more GUI applications:

Kweb's Bookmark Utility



Do you want to add the following URL to kweb bookmarks?
https://www.raspberrypi.org/

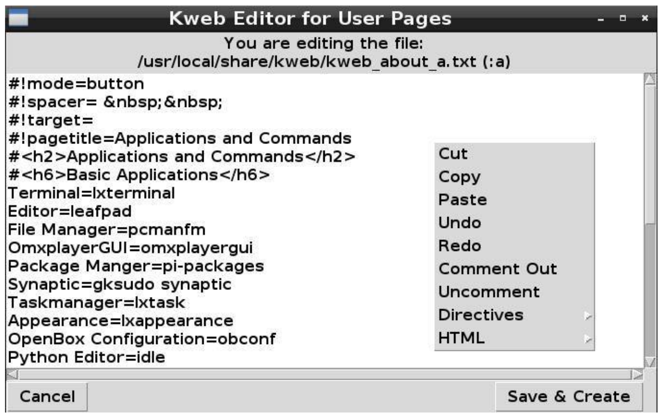
You can edit the name of the bookmark link below:
Raspberry Pi - Teach, Learn, and Make with Raspberry Pi

Select the position in your bookmark list.
The bookmark will be inserted after the selected item.

```
#<h2>Bookmarks</h2>
Raspberry Pi Forum=http://www.raspberrypi.org/forum
Omxplayer Download=http://omxplayer.sconde.net/
Youtube=http://www.youtube.com/
```

This program is opened, whenever you use the “Bookmark Page” command from kweb's menu or by using the ALT+, keyboard short-cut.
You can modify the name of the link which will appear on your bookmarks page (default is the page title) and also select the position where the new bookmark should appear in your list of bookmarks.

Kweb's Text Editor



You are editing the file:
/usr/local/share/kweb/kweb_about_a.txt (:a)

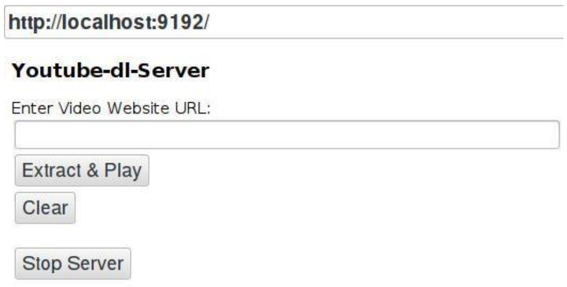
```
#!mode=button
#!spacer= &nbsp;&nbsp;
#!target=
#!pagetitle=Applications and Commands
#<h2>Applications and Commands</h2>
#<h6>Basic Applications</h6>
Terminal=lxterminal
Editor=leafpad
File Manager=pcmanfm
OmxplayerGUI=omxplayergui
Package Manger=pi-packages
Synaptic=gksudo synaptic
Taskmanager=lxtask
Appearance=lxappearance
OpenBox Configuration=obconf
Python Editor=idle
```

Kweb's user editable pages (Bookmarks, Utilities, Applications) are created from simple text files, which you can edit to remove or add your own links, commands or settings without any knowledge of HTML coding. This editor will be opened when you click one of the “Edit” buttons at the bottoms of these pages or in the “:Editor” page.
If you save your changes, the HTML page will be immediately compiled from the modified text file.
This editor can also be used to create your own :Command pages, or to modify user definable keyboard commands. Developers of embedded applications may use it to generate kweb's special command links (and more) and to insert them into their HTML pages. Part three of this manual contains a detailed description of the syntax and of the editor's possibilities.

Youtube-dl-server

There is one more application, which is not a GUI program but a web server and therefore will run in a terminal when you start it from kweb's application page (at the bottom, one of the "Start Server" buttons). It is also run in the background if you open omxplayerGUI's standalone interface. This application reduces the access time for extracting video URLs from web pages using the "Play" command. You should run it whenever you plan an extended web video session.

You can access the youtube-dl-server from within kweb directly (click the "Open Server" button) and use it interactively. Just copy a video web page URL into the entry field and click "Extract & Play". You can also use drag and drop to insert links from another browser window, e. g. from a youtube search result. This should even work from other browsers (but the youtube-dl-server page must be running within kweb!).



d) Kweb's Home Page

Kweb must go somewhere when you click it's "Home" button. By default kweb's menu page is used as your home page, but you can also set a home page address on kweb's configuration page which can be any local HTML file or a website on the Internet.
You can also create your own home page file, which will be used automatically if you name it "homepage.html" and move it to "/usr/local/share/kweb". In part three you will learn how you can create your own home pages from simple text files, especially for use as a desktop environment.

Note: If you set a home page file path outside of the protected (root) area, e. g. in "/home/pi", command execution will be disabled by default, because this is considered as not secure. If you have used a file "homepage.html" in your main user directory in kweb versions prior to 1.7.0, you have to move it to "/usr/local/share/kweb".

It's also possible to modify the home page address at run time, using the "Set as Home" command from kweb's menu (or with ALT+l). This is not used permanently, but only during the current session.

8. A Few Recommendations for Best User Experience

a) Browser settings

Kweb's default settings are very “defensive”: JavaScript and cookies are disabled and the browser runs in “Private Mode”, which means that nothing will be stored permanently on your file system. And some “experimental extensions” of the webkit engine are also shut off.

Kweb was developed quite some time before the Raspberry Pi 2 became available and it's main goal was to use as few resources as possible. Many people have said that you cannot use the Raspberry Pi for decent web browsing, but in fact it's not the Raspberry Pi that's too slow but the way many websites are designed: using hundreds of elements on a page, with tons of JavaScript, most of which is not really needed at all except for throwing ads at you or using it for fingerprinting (identifying you without the help of cookies).

Many websites which are using JavaScript will also work without, and usually much (sometimes ten times) faster. So it's good practice on the Pi to only switch on JavaScript if it is really needed. If you open multiple websites in different windows, enabling JavaScript in one window won't affect the web pages in the other windows.

But if you think: “I don't care, I just want to get all the features I get when using any other 'large' browser”, there is a simple solution. Open kweb's configuration page, scroll down to the bottom, click the “Full Power Browser” preset and restart kweb(3). From now on all possible features are enabled when you start kweb(3). And use kweb3 for best overall performance!

b) Web Video

Both kweb and kweb3 can play embedded web video (usually JavaScript has to be enabled). In kweb web video is not accelerated by the GPU and so it will really lag. Kweb3 uses the webkit3 engine with hardware acceleration for web video and JPEG decoding. Web video will play quite well, as long as the size of the video area is not too large (don't go to full screen video!).

But my recommended way of watching web video is using omxplayer(GUI). You'll get a higher resolution and much better performance. And it will also work with many websites which still use the Adobe flash player instead of HTML5 video.

You need a working and up to date version of youtube-dl, which is used to extract the real video URLs from video websites. This tool has grown a lot over the years and now supports more than 600 websites. Unfortunately it has become rather slow on the Rpi, because it's now using more than 600 different modules which require quite some time to load when you use it in the normal way. I've always searched for solutions to speed up its use and version 1.7.0 now offers the best possible solution I can think of. This requires two steps on your side:

1) Install the github version of youtube-dl and update it regularly. All this can be done with one mouse click on kweb's application page. Access time for web video should be between 5 and 6 seconds (videos starting to play after clicking the “Play” button) on an RPi2.

2) Use the youtube-dl-server whenever you want to watch lots of videos. You can start it from the application page, and it is also started automatically if you open the omxplayerGUI frontend. This will reduce access time by another 2 or 3 seconds.

c) Using Multiple Cores (Raspberry Pi 2/3 only)

The webkit engines used by kweb and kweb3 use only one of the four cores on a Raspberry Pi 2. When opening multiple web pages in different windows, one long loading web page (especially with JavaScript enabled) may block everything else inside the browser. Kweb supports multiple cores in two ways.

1) Kweb uses external programs for a number of things: playing media, downloading files, opening PDF documents etc. All those programs are started as a separate process and will run on a separate core.

2) It's possible to open a new web page in a new browser instance (running on another core) instead of in a new window (of the same browser instance). There are two ways to do this:

One way is to hold the SHIFT key pressed while clicking on a link. You can use this method for example, if you open a new website from your bookmarks page and know that this page will take some time to load and get running.

The second method requires that a web page has already been opened. Using the menu command "Open in New Browser" (or ALT+. from the keyboard) will close the current window (if it is not the only one) and open the same web page in a new browser instance.

Of course this method has its limits. Multiple browser instances require more memory and we have only four cores that we can use. You should not run more than 3 instances at once (leaving one core for system processes).

d) Conclusion

Getting a good Internet performance is not as easy on a Raspberry Pi as on a much more powerful desktop PC (which also needs much more power). It takes some know-how and you have to learn to use the limited resources sparingly. But after all the Raspberry Pi has been developed for education and you may learn a lot about how the Internet works that way. Kweb gives you the most direct way of controlling your resources.

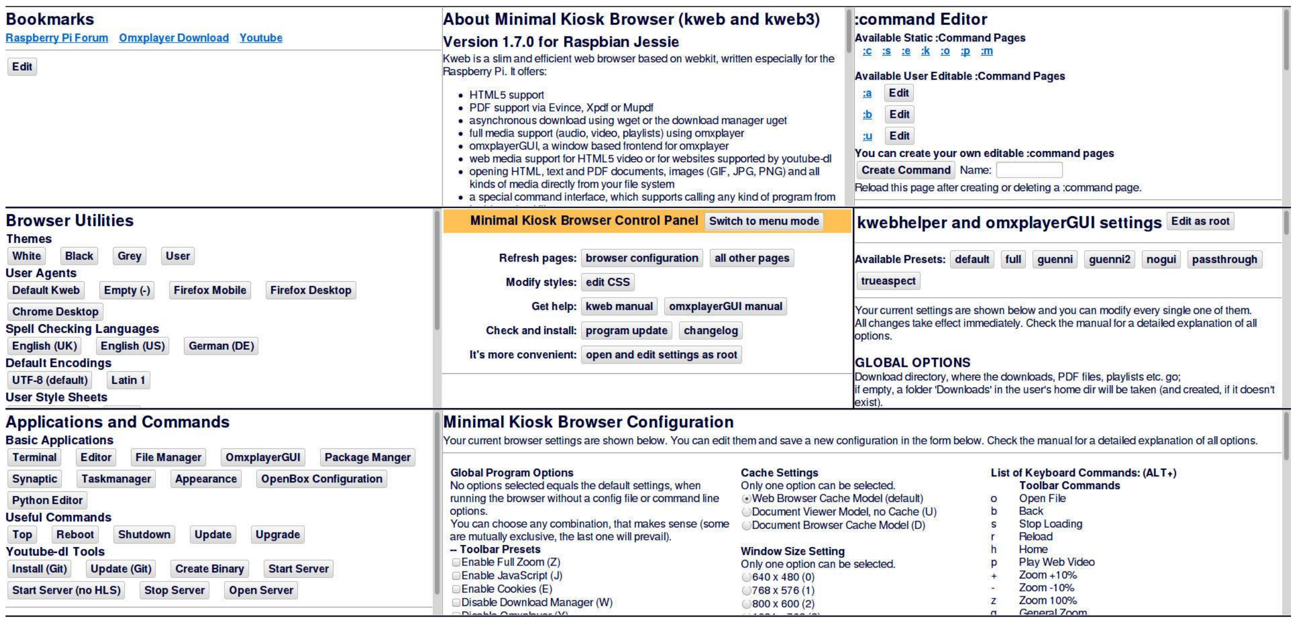
You may miss some features in kweb you are used to having in other browsers: tabbed browsing, auto completion of URLs while you are typing, some kind of progress bar etc. That is the price you have to pay for its minimalistic (and effective) approach. It has been designed this way and that won't change. If you think you need these features, use one of the "large" browsers like epiphany or iceweasel.

Other browsers have their own merits. The Mozilla engine (Iceweasel, Firefox) has a better performance in many ways than the webkit engines (although it is missing hardware acceleration on the Raspberry Pi), but unfortunately Mozilla no longer supports embedding its engine into other browsers. Developers of independent browsers have no other choice today than to use the webkit engine.

If you have read so far you may already have discovered that despite its simplicity, kweb is highly configurable and extensible and offers some unique features which are described in the next two parts of this manual. In fact it is much more than just a browser and can do things which are not possible with any other browser.

Part Two

Configuring Kweb and its Environment



About Minimal Kiosk Browser (kweb and kweb3)

Version 1.7.0 for Raspbian Jessie

Kweb is a slim and efficient web browser based on webkit, written especially for the Raspberry Pi. It offers:

- HTML5 support
- PDF support via Evince, Xpdf or Mupdf
- asynchronous download using wget or the download manager uget
- full media support (audio, video, playlists) using omxplayer
- omxplayerGUI, a window based frontend for omxplayer
- web media support for HTML5 video or for websites supported by youtube-dl
- opening HTML, text and PDF documents, images (GIF, JPG, PNG) and all kinds of media directly from your file system
- a special command interface, which supports calling any kind of program from

Refresh pages: browser configuration | all other pages
Modify styles: edit CSS
Get help: kweb manual | omxplayerGUI manual
Check and install: program update | changelog
It's more convenient: open and edit settings as root

You can create your own editable :command pages

Reload this page after creating or deleting a :command page.

Your current settings are shown below and you can modify every single one of them. All changes take effect immediately. Check the manual for a detailed explanation of all options.

GLOBAL OPTIONS

Download directory, where the downloads, PDF files, playlists etc. go;
if empty, a folder 'Downloads' in the user's home dir will be taken (and created, if it doesn't exist).

Minimal Kiosk Browser Configuration

Your current browser settings are shown below. You can edit them and save a new configuration in the form below. Check the manual for a detailed explanation of all options.

Global Program Options
No options selected equals the default settings, when running the browser without a config file or command line options.
You can choose any combination, that makes sense (some are mutually exclusive, the last one will prevail).
-- Toolbar Presets
☐ Enable Full Zoom (Z)
☐ Enable JavaScript (J)
☐ Enable Cookies (E)
☐ Disable Download Manager (W)

Cache Settings
Only one option can be selected.
● Web Browser Cache Model (default)
○ Document Viewer Model, no Cache (U)
○ Document Browser Cache Model (D)

Window Size Setting
Only one option can be selected.
○ 640 x 480 (0)
○ 768 x 576 (1)
○ 800 x 600 (2)

List of Keyboard Commands: (ALT+)

	Toolbar Commands
o	Open File
b	Back
s	Stop Loading
r	Reload
h	Home
p	Play Web Video
+	Zoom +10%
-	Zoom -10%
z	Zoom 100%
g	General Zoom

Kweb's control panel page – everything in one place

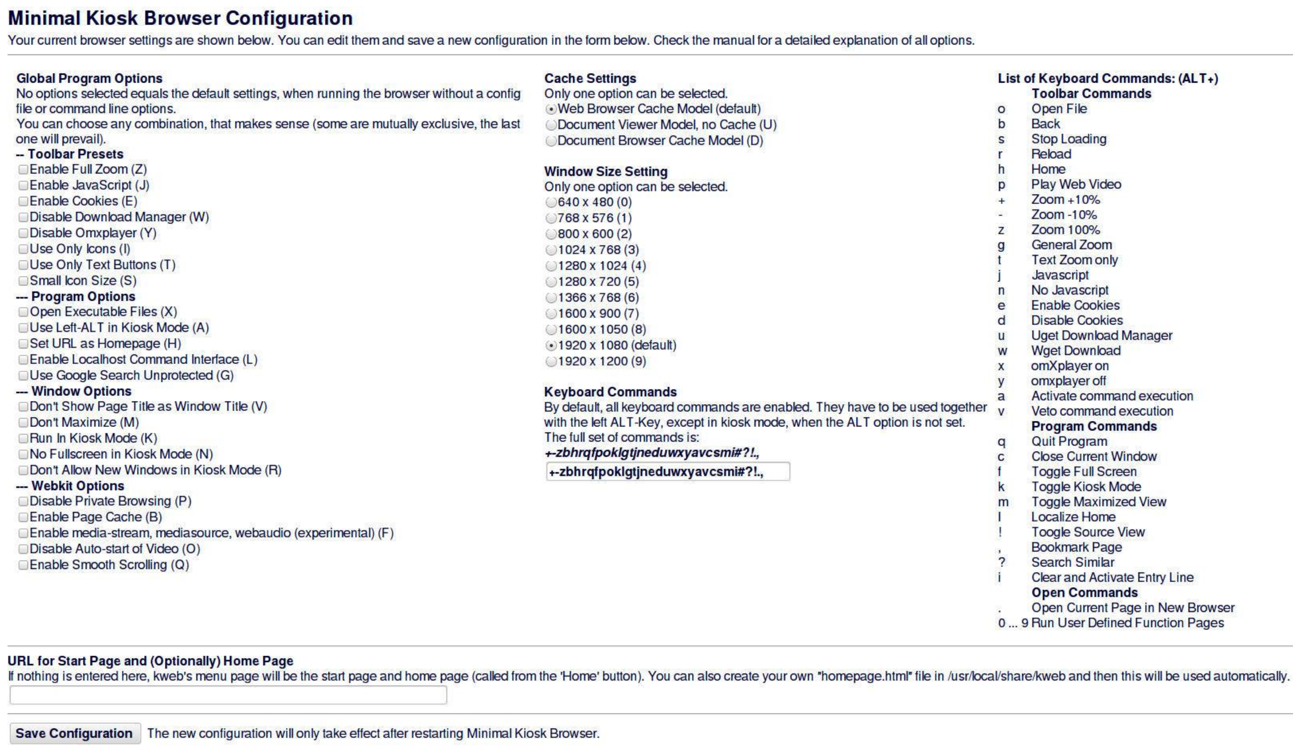
9. Configuring Kweb and its Helper Applications

a) Browser Configuration

You can call the browser configuration page from kweb's menu page or with the “:c” command.

Note: To modify anything “Commands” execution must be enabled in the toolbar.

The upper part of the configuration page looks like this (default settings):



Minimal Kiosk Browser Configuration

Your current browser settings are shown below. You can edit them and save a new configuration in the form below. Check the manual for a detailed explanation of all options.

Global Program Options
No options selected equals the default settings, when running the browser without a config file or command line options.
You can choose any combination, that makes sense (some are mutually exclusive, the last one will prevail).

-- Toolbar Presets
- ☐ Enable Full Zoom (Z)
- ☐ Enable JavaScript (J)
- ☐ Enable Cookies (E)
- ☐ Disable Download Manager (W)
- ☐ Disable Omxplayer (Y)
- ☐ Use Only Icons (I)
- ☐ Use Only Text Buttons (T)
- ☐ Small Icon Size (S)

--- Program Options
- ☐ Open Executable Files (X)
- ☐ Use Left-ALT in Kiosk Mode (A)
- ☐ Set URL as Homepage (H)
- ☐ Enable Localhost Command Interface (L)
- ☐ Use Google Search Unprotected (G)

--- Window Options
- ☐ Don't Show Page Title as Window Title (V)
- ☐ Don't Maximize (M)
- ☐ Run In Kiosk Mode (K)
- ☐ No Fullscreen in Kiosk Mode (N)
- ☐ Don't Allow New Windows in Kiosk Mode (R)

--- Webkit Options
- ☐ Disable Private Browsing (P)
- ☐ Enable Page Cache (B)
- ☐ Enable media-stream, mediasource, webaudio (experimental) (F)
- ☐ Disable Auto-start of Video (O)
- ☐ Enable Smooth Scrolling (Q)

Cache Settings
Only one option can be selected.
- [✓ Web Browser Cache Model (default)]
- ○ Document Viewer Model, no Cache (U)
- ○ Document Browser Cache Model (D)

Window Size Setting
Only one option can be selected.
- ○ 640 x 480 (0)
- ○ 768 x 576 (1)
- ○ 800 x 600 (2)
- ○ 1024 x 768 (3)
- ○ 1280 x 1024 (4)
- ○ 1280 x 720 (5)
- ○ 1366 x 768 (6)
- ○ 1600 x 900 (7)
- ○ 1600 x 1050 (8)
- [✓ 1920 x 1080 (default)]
- ○ 1920 x 1200 (9)

Keyboard Commands
By default, all keyboard commands are enabled. They have to be used together with the left ALT-Key, except in kiosk mode, when the ALT option is not set.
The full set of commands is:
+-zbhrqfpoklgtjneduwxyavcsmi#?!.,

`+-zbhrqfpoklgtjneduwxyavcsmi#?!.,`

List of Keyboard Commands: (ALT+)

	Toolbar Commands
o	Open File
b	Back
s	Stop Loading
r	Reload
h	Home
p	Play Web Video
+	Zoom +10%
-	Zoom -10%
z	Zoom 100%
g	General Zoom
t	Text Zoom only
j	Javascript
n	No Javascript
e	Enable Cookies
d	Disable Cookies
u	Uget Download Manager
w	Wget Download
x	omXplayer on
y	omxplayer off
a	Activate command execution
v	Veto command execution
	Program Commands
q	Quit Program
c	Close Current Window
f	Toggle Full Screen
k	Toggle Kiosk Mode
m	Toggle Maximized View
l	Localize Home
!	Toogle Source View
,	Bookmark Page
?	Search Similar
i	Clear and Activate Entry Line
	Open Commands
.	Open Current Page in New Browser
0 ... 9	Run User Defined Function Pages

URL for Start Page and (Optionally) Home Page
If nothing is entered here, kweb's menu page will be the start page and home page (called from the 'Home' button). You can also create your own "homepage.html" file in /usr/local/share/kweb and then this will be used automatically.

Save Configuration The new configuration will only take effect after restarting Minimal Kiosk Browser.

Each one of the global program options on the left side can be enabled or disabled. They are divided into subsections and most should be easy to understand (and if you don't understand them you won't need them, I suppose). The character to be used in a configuration string (for script or command line use) is shown in parentheses. Here's a list of the global options with some additional information:

Z = enable full zoom on start instead of text zoom only, which is the default

J = enable JavaScript (default is disabled)

E = enable cookies (default is disabled)

W = use wget for downloads (default is uGet)

Y = disable video and audio being played with omxplayer (by default it is enabled)

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

X = allow executables to be started from the "open" command

A = use left ALT-key for keyboard commands in kiosk mode also

H = use URL, if supplied, as home page instead of the default home page. 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?...” to that argument. This is a security risk and should be used with care. Normally command links only work inside “file://...” URLs. This is a special developers option; no kweb environment pages will work any more, if this options is enabled!

G = use Google as search engine instead of using it via startpage.com

V = don't show page title as title of current window (by default activated)

M = don't maximize kweb window on start (except in kiosk mode).

K = run browser in kiosk mode

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, slide shows etc.).

R = Don't allow new windows in kiosk mode – needed by some kinds of embedded applications.

P = disable private browsing. If private browsing is enabled (default setting) nothing is permanently stored in your file system.

B = enable page cache for history (faster “back”, but costs resources)

F = enable special webkit features, which are considered “experimental”: media-stream (http://www.w3.org/TR/mediacapture-streams/), mediasource (http://www.w3.org/TR/media-source/), webaudio (https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html).

O = Disable auto start of embedded video (enabled by default)

Q = >Enable Smooth Scrolling – may be heavy on the CPU (not accelerated)

There's one more option character, “C”, which does not appear on the configuration page because it can only be used inside a command line options string, never in a configuration file. It activates the command execution when the browser starts. This is only needed if your start page or home page are files inside your user directory; otherwise command execution is enabled by default. Developers of embedded applications will usually call kweb with a command line configuration for their applications and if the local HTML project pages are within the user directory they have to add this option to their command line string (usually a script).

In the middle column there are three different kinds of settings. In the first part you can define the cache settings of the browser and this will result in more or less resources used by the browser (especially memory). Only one option can be selected.

Note: If don't want your browser to write anything to your file system (to avoid SD card corruption, for example), select the “Document Viewer Model”, but it may slow down page loading.

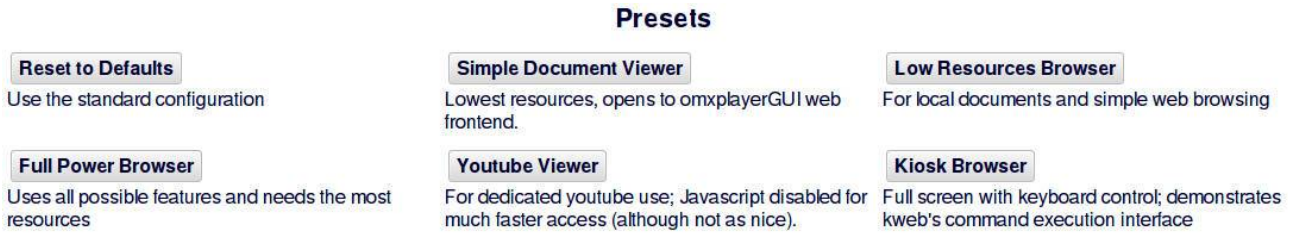
The next part sets a default window size. In combination with the “Don't Maximize” option it will set the size of your browser window when the browser opens. It's also important for embedded applications, when a certain screen size is needed (especially, when no window manager is used). Again only one of these options can be selected.

The lowest part is the selection of supported keyboard commands. By default all are used. For developers of embedded applications it may be useful to disable some or all commands, by clearing the input field or by reducing it to a few selected commands. The meaning of all possible keyboard commands is explained in the right part of the configuration page.

On the bottom you'll find an entry field for an URL. This will be your start page when the browser opens and can also be used as your home page, if the option “use URL as home page” on the left side is enabled.

If you are finished with setting your configuration, click the “Save Configuration” button. Next time you start the browser it will use these configuration settings.

In the lower part of the web page you will find some example configurations called “Presets”, which can be selected with a mouse click.



Presets

Reset to Defaults	**Simple Document Viewer**	**Low Resources Browser**
Use the standard configuration	Lowest resources, opens to omxplayerGUI web frontend.	For local documents and simple web browsing
Full Power Browser	**Youtube Viewer**	**Kiosk Browser**
Uses all possible features and needs the most resources	For dedicated youtube use; Javascript disabled for much faster access (although not as nice).	Full screen with keyboard control; demonstrates kweb's command execution interface

After selecting one of the example configurations you have to reload the page (from the toolbar, the right click pop-up menu or with ALT+r) to see the actual configuration. You can of course, also modify such an example configuration and save the result. Again the browser needs to be restarted to use the new configuration.

b) Helper Programs Settings

Kweb's helper applications (especially kwebhelper.py and omxplayerGUI) have even more fine tuning options than kweb itself. All helper configuration can be done through a web interface, kweb's settings page. It's also possible to define presets, which can be selected with a simple mouse click .

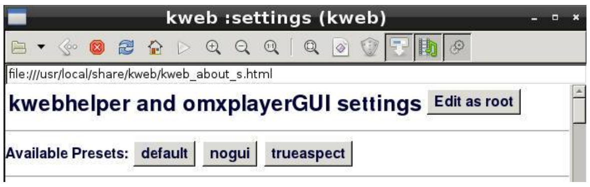
Any changes in the settings will take effect immediately; you don't have to start kweb again. But if an instance of omxplayerGUI is running while you are editing its settings, you will have to restart it to use the new settings.

To modify anything “Command” execution must be enabled with the right most toggle button in the toolbar (or with ALT+a).

Kweb's helper programs kwebhelper.py (responsible for PDF support, downloads and kweb's command execution interface) and omxplayergui.py use the same global settings file (another Python script, kwebhelper_settings.py). Therefore the configuration for both programs is managed by the same settings page, which can be opened via the “:s” command from the entry line. It's also available from the menu and control panel page.

Here we'll only describe the sections that are used for configuring global options, the PDF interface, download options and the command execution. All settings for omxplayerGUI (playing video and audio, including embedded web video) are described in detail in the omxplayerGUI manual.

There's one thing to consider when doing any kind of editing in this area. The settings are stored in the protected part of your file system (for security reasons) and any change requires root rights, which is realized via sudo. If you have set up your pi to ask for a password when using sudo (which you should, for security reasons), you will be asked for your password (inside a terminal) each time you change some settings value or issue certain commands. This is not very comfortable if you want to apply a lot of changes. But there is a solution for this: if you click the button “Edit Settings as Root” (in different places), a second, specially configured kweb browser will be started in root mode (via gksudo). You will be asked for your password only once and then may apply any number of changes.


kweb :settings (kweb)
file:///usr/local/share/kweb/kweb_about_s.html
kwebhelper and omxplayerGUI settings
Edit as root
Available Presets:
default
nogui
trueaspect

On top of the web page,you'll see a few buttons with “presets”, which can be chosen with a mouse click. After selecting a preset, you have to reload the web page to see which options have been set. Three presets are predefined: “default” (default settings), “nogui” (windowless mode for media) and “trueaspect” (a special mode for newest omxplayer versions). The differences only affect omxplayerGUI. You'll see later, how you can create your own presets.

Each of of the following options can be modified and (must be) saved separately.

GLOBAL OPTIONS

Download directory, where the downloads including PDF files, playlists etc. go;
if empty, a folder 'Downloads' in the user's home dir will be taken (and created, if it doesn't exist).

If you have a HDD connected to your Raspberry Pi you should use it to save your downloads to instead of using the SD card.

dldir:

save

PDF OPTIONS (kwebhelper)

Preferred pdf reader: either evince, xpdf or mupdf. If left empty, the program will try to find the best PDF reader
Selecting an installed program of your choice will speed it up a bit

pdfprog: save

Additional options for pdf program (must match the selected program!):

pdfoptions: Enter one element per line!

You must set a program of your choice above, when you want to set additional options here for the selected PDF program.

save

This will allow kweb to open pdf files on a local server as files instead of downloading them first; this will only work with "http://localhost" links

pdfpathreplacements: Enter one element per line in the form value1=value2

e. g.

http://localhost/PDF=/var/www/PDF

You

save

You can add more than one replacement path, one per line.

A very special option, disabled by default. If you have a local web server on your Raspberry Pi that 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. If your PDF files are served from “http://localhost:8073/Ebooks1” and they are placed in '/var/www/Ebooks1', for example, enter:
http://localhost:8073/Ebooks1=file:///var/www/Ebooks1
kwebhelper will then send the file paths directly to your PDF reader instead of downloading the files to the “Downloads” directory first.

DOWNLOAD OPTIONS (kwebhelper)

Defines if wget will run in a terminal (visual control) or in the background:

show_download_in_terminal: [✓ True] False save

Options for wget:

Do not remove the default options!

wget_options: Enter one element per line!

```
--no-check-certificate
--no-clobber
--adjust-extension
--content-disposition
```

save

Options for download manager uget:

uget_options: Enter one element per line!

```
--quiet
```

save

COMMAND EXECUTION OPTIONS (kwebhelper)

If this is set to "True", all Desktop (GUI) programs will be executed without starting a terminal first

check_desktop: [✓ True] False save

This will only work, if the name of the desktop file and the binary match (true for most programs).

Direct commands will be executed without starting a terminal first. Use it for background commands or programs with a GUI that are not desktop programs or if check_desktop is set to "False"

direct_commands: Enter one element per line!

```
kwebhelper.py
omxplayergui
kwebhelper_set.py
omxplayer
gksudo
xterm
screen
```

save

You may add other programs or scripts here, but do not not remove any of the preset programs!

Preferred terminal to run commands in, must be set ('xterm' or 'lxterminal')

preferred_terminal: `lxterminal` save

Set the following to False if you don't want to run 'sudo' commands inside a terminal,
but only if a password is not required (you may break command execution otherwise):

sudo_requires_password: [✓ True] False save

The commands from this page (and others) have to be executed as root. They will be run inside a terminal to let you enter your password.

If a password is not required (insecure), you may set this option to "False" to avoid a terminal popping up each time you save an option.

Set the following to "True", if you want to run all commands from a script file.
It may help with complex command links, but will require more disk accesses.

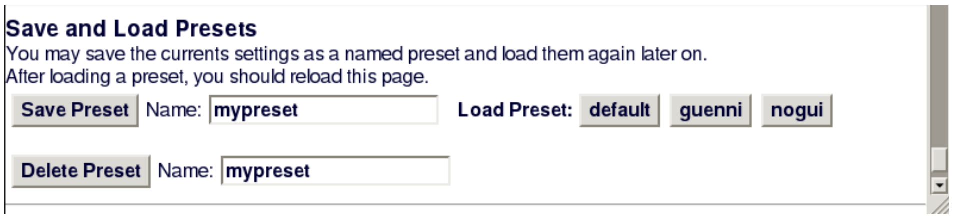
This option should not be used, because it results in a lot of disk write operations.

May be required for some special applications.

run_as_script: True [✓ False] save

All other options are only used by omxplayerGUI and are described in its manual.

After editing a number of options you can save the complete actual settings as a preset, that can later be loaded with a simple mouse click. To save a preset, scroll down to the bottom of the page, enter a name and click “Save Preset”.



Save and Load Presets
You may save the currents settings as a named preset and load them again later on.
After loading a preset, you should reload this page.

Save Preset Name: mypreset **Load Preset:** default guenni nogui

Delete Preset Name: mypreset

To overwrite the preset again later with different settings, simply use the same name. To delete a preset, enter its name and click “Delete Preset”.

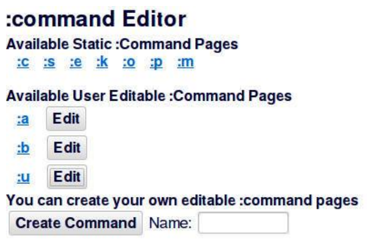
To load a preset, simply click on the button with its name. Reload the page afterwards to see the active settings. This function is also available at the top of the settings page.

10. Using the :command Editor

a) Introduction

The ":Editor" page gives access to a number of functions to modify or extend the kweb environment or to add complex user defined functions (also available as keyboard short-cuts ALT+0....9).

The upper part of the page shows an overview of predefined fixed and user editable :Command pages and also lets you add your own extensions. Most of this will be explained in the following section and an example of creating your own editable :Command will be described in part three.


:command Editor
Available Static :Command Pages
:c :s :e :k :o :p :m
Available User Editable :Command Pages
:a Edit
:b Edit
:u Edit
You can create your own editable :command pages
Create Command Name:

The second part of the :Editor page lets you create user definable functions (keyboard commands) and will be treated later on.

b) Understanding the :Command Interface

The ":x" commands (x may be a name of any length, but for the predefined ones we use a single character) which you can call from your entry line or simply open by clicking on an appropriate link inside your :Command editor page (':e'), are simply html files in a certain directory (/usr/local/share/kweb) using a special rule of name building. The ":e" command, for example, is realized by the file "kweb_about_e.html" inside that folder. If you want to add a static command page to your system, say ":me", you can simply create an HTML file named "kweb_about_me.html" and move it to that folder.

There are also user editable command files (bookmarks ':b', applications ':a', utilities: ':u'). These are created from simple text files, which you can edit with kweb's text editor. The web pages are automatically rebuilt if you save your additions or changes from this editor. You can also create your own editable :command pages from the editor page.

c) Editing Predefined :Command Pages

The next chapter will give a more detailed description about how you can edit and organize your bookmarks page. Here I will just demonstrate how you an add something to the "Utilities" and "Application" pages. As a simple example let us add another language for spell checking to the "Utilities" page. Click the "Edit" button at the bottom of the "Utilities" page. In the text editor scroll down to the line:
German (DE)=!de_DE
Enter a new line with (for example):
French=!fr_FR
Click "Save & Create" and from now on you can select French spell checking from the "Utilities" page (of course you need to have a French spell checking library installed).

Editing the "Application" page is similar. This page demonstrates kweb's ability to start other programs by just clicking a link (or button). You may remove commands you don't need or add new ones like this:
name=command
where "command" can be any kind of command line string as you would type it into a terminal.

d) Tutorial: Organizing your Bookmarks Page

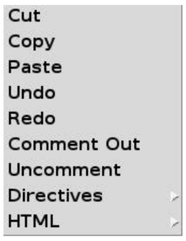
A very simple user editable command page is the bookmarks page (:b), which will be opened by default on the menu page. Usually you will add bookmarks directly from the browser using kweb's bookmark utility, but if you want to organize your bookmarks in a better way or even redesign it you have to use the "Edit" function at the bottom of the bookmarks page or the matching button on the ":Editor" page.

Kweb's special text editor will be started (as root with gksudo, which may ask for your password). The default bookmarks page looks like this:

```
#!mode=link
#!spacer= &nbsp;&nbsp;
#!target=top
#!pagetitle=Bookmarks
#<h2>Bookmarks</h2>
Raspberry Pi Forum=https://www.raspberrypi.org/forum
Omxplayer Download=http://omxplayer.sconde.net/
Youtube=https://www.youtube.com/
```

If you have already added some bookmarks, the list will be longer of course.

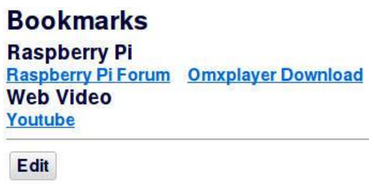
Ignore the first lines for now. We'll just show some editing features in this example to help you reorganize your bookmarks. You may manually add bookmarks by entering new lines (at the bottom or in between) like this:
name=URL
where 'name' can be anything and 'URL' must be a valid "file://" or "http://" URL. You can delete bookmarks by simply deleting the appropriate lines. You can reorganize them using "Cut" and "Paste" commands. At this point it's a good idea to have a look at the editor's right click pop-up menu.



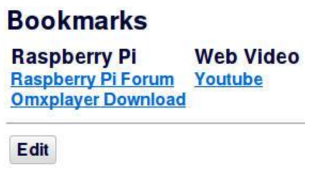
If you have lots of bookmarks it may be useful to organize them into categories. Set the cursor to the start of a new line, open the right click menu and click on "HTML" then from the sub menu select one of the "Category" entries. You will be asked to enter a title for the category. This will insert a (smaller or larger) headline, which will also create a horizontal divider. Repeat this step for as many categories as you like. You may also want to move your already existing bookmarks around using "Cut" and "Paste".

For this example I've added two categories using the "Category (small)" entry from the HTML menu to the default bookmarks list. The whole text now looks like this:

```
#!mode=link
#!spacer= &nbsp;&nbsp;
#!target=top
#!pagetitle=Bookmarks
#<h2>Bookmarks</h2>
#<h5>Raspberry Pi</h5>
Raspberry Pi Forum=http://www.raspberrypi.org/forum
Omxplayer Download=http://omxplayer.sconde.net/
#<h5>Web Video</h5>
Youtube=https://www.youtube.com/
```



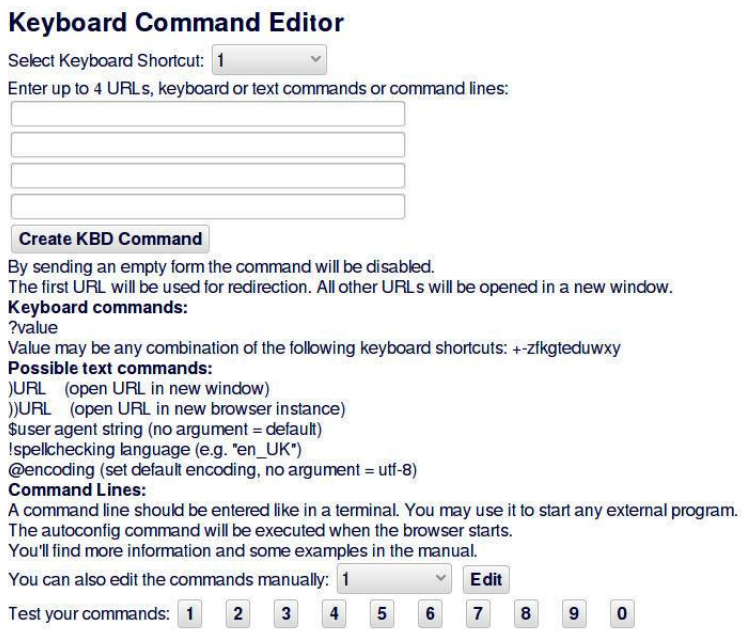
Click "Save & Create" to save your changes and to create the new bookmarks page. Reload the bookmarks page afterwards. It now appears as shown in the image.

We want to go one step further now and organize our bookmarks into multiple columns. We click the “Edit” button again and apply the following changes:
We change the second line to
```
#!spacer=<br>
```
After the line
```
#<h2>Bookmarks</h2>
```
we add a new line and insert “#!table=..” from the “Directives” menu. It will look like this:
```
#!table=1:1:0:0:4:top:center
```
we have to modify it a bit, so it looks like this:
```
#!table=1:8:0:0:4:top:left
```
We have created a table with 8 columns. To replace the current single content placeholder with the table we just created we have to add another directive from the menu
```
#!next
```
All following content will be placed in the first column. To go to the next columns we have to add another “#!next” directive before the content that should be placed in the second column. And I have added six more empty “#!next” commands at the bottom, because we have six more (empty) columns left.
```
#!next
#<h5>Web Video</h5>
Youtube=http://www.youtube.com/
#!next
#!next
#!next
#!next
#!next
#!next
```


Click “Save & Create” and reload the bookmarks page. It now shows our bookmarks organized in two columns. The other columns will appear when we start adding content to them.

e) Creating User Defined Functions (Keyboard short-cuts)

The second part of the :Editor page contains the “Keyboard Command Editor”, which allows you to create complex functions, available as keyboard short-cuts or links. Technically speaking these are simple HTML files (kweb1.html, kweb2.html etc.), which are loaded when you call the matching keyboard short-cut (ALT+1, ALT+2 etc.). But they are built in a special way: they run a number of hidden commands and then either return to your current page or go forward to a specific URL, which you have set up.


Keyboard Command Editor
Select Keyboard Shortcut: 1
Create KBD Command
Edit
Test your commands: 1 2 3 4 5 6 7 8 9 0

Enter up to 4 URLs, keyboard or text commands or command lines:

By sending an empty form the command will be disabled.
The first URL will be used for redirection. All other URLs will be opened in a new window.
Keyboard commands:
?value
Value may be any combination of the following keyboard shortcuts: +-zfkgteduwxy
Possible text commands:
)URL (open URL in new window)
))URL (open URL in new browser instance)
$user agent string (no argument = default)
!spellchecking language (e.g. "en_UK")
@encoding (set default encoding, no argument = utf-8)
Command Lines:
A command line should be entered like in a terminal. You may use it to start any external program.
The autoconfig command will be executed when the browser starts.
You'll find more information and some examples in the manual.
You can also edit the commands manually: 1

For each short-cut (0 – 9) you can set up to 4 URLs, commands, browser configuration strings or keyboard commands (limited subset). With one keyboard command or mouse click we can open multiple websites, start programs or configure the browser.

Let's start with a nice example:
We can select the keyboard short-cut first from the drop down menu, but we'll just leave it at the first value “1” for our exercise.

In the first line enter:
`https://www.youtube.com/`
in the second line:
`$Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36`
and in the third line:
`ytdl_server.py`
Then click: “Create KBD Command”.

If you now press ALT+1 or click the button “1” at the bottom, the following will happen:
The browser will open the youtube website, but before doing that it will set its user agent to a recent Google Chrome version (youtube sometimes doesn't like our browser and wants to tell us to install another one and so we fake a Chrome browser) and it will also start the youtube-dl-server for faster video access.

If you are happy with this function and want to always use it for youtube access but don't want to use the keyboard for it, we can do one more thing:
Open the bookmarks page and click on “Edit”.
In kweb's text editor find the line:
`Youtube=https://www.youtube.com/`
Change it to:
`Youtube=?1`
or to:
`Youtube=file:///usr/local/share/kweb/kweb1.html`
(the second method also works for right click and “Open in New Window”)
and click “Save & Create” at the bottom of the editor window.
Reload the bookmark page.

We have attached the keyboard function ALT+1 (or a direct link to the command page) to the youtube link in our bookmarks list and whenever we now click on this link we not only go to youtube, but also change the user agent and start the youtube-dl-server (if it's already running the new version will simply stop again).

As a second example let's create a Python working environment. Select “2” for the short-cut and enter the following lines:

```
https://docs.python.org/2.7/
)http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html
idle
```

After clicking “Create KBD Command” we now have a new keyboard command (ALT+2) which will open the Python (2.7) documentation, the Python Tkinter tutorial in a second window and start the Python editor “Idle”.

One last example. The Raspberry Pi forum is hard to read on a full HD screen, at least for old people like me with bad eyes. So let's create a command to go there with better settings:

```
https://www.raspberrypi.org/forums/
?egz+++++
```

This will execute a number of keyboard commands before entering the forum: enable cookies (to be able to login), enable global zoom and set the zoom to 150%.

If you need more than 4 entries, you can manually edit the text files from which the command pages are created, but this requires a deeper understanding of the way it works and you have to read part three of this manual for a full description.

f) Creating an Auto-Configuration Page

If you open the selection menu for the keyboard short-cut, you will find one entry at the bottom, which is not used for a keyboard short-cut at all: **autoconfig**

If you select this option, an auto-configuration page will created instead of a keyboard command. This page will always be run when the browser starts (if it exists), before it enters your start page or home page.

The possibilities are almost the same (there are a few restrictions) as with keyboard commands. You can use it to open multiple websites or even multiple browser instances, to change browser settings (user agent, for example), to set the zoom level or to start additional programs.

To delete the auto-configuration page again, just send an empty form: select "autoconfig" and click on "Create KBD Command" without entering anything into the four text fields.

11. Styling the Interface

a) Styling the Toolbar

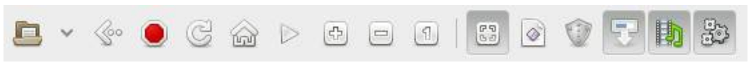
If you do not like the way the toolbar looks like you may modify it in lots of different ways.

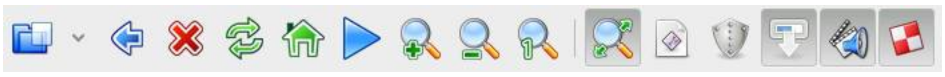
1) There are some options available on the configuration page:
I = use only icons without text
T = use only text buttons without icons
S = use always small icons.

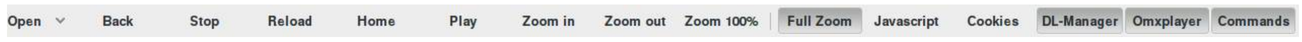
2) Using the Appearance application (lxappearance):
You may select a font and font size (for the text of the buttons), choose a symbol theme or set the icon size (if the "S" option is not used).

Note: Currently the Appearance application doesn't work correctly with GTK+3 any more. This will affect kweb3: the icon size is therefore fixed to small or large icons. Text styling also has some issues. And while changes in appearance are immediately applied to running GTK+2 applications, GTK+3 programs have to be restarted for the changes to take effect.

Examples using different symbol themes and icon sizes in kweb and kweb3 or simple text buttons:


Icons only, small icons, Adwaita theme


Icons only, large icons, Nuvoladesign theme



Text only

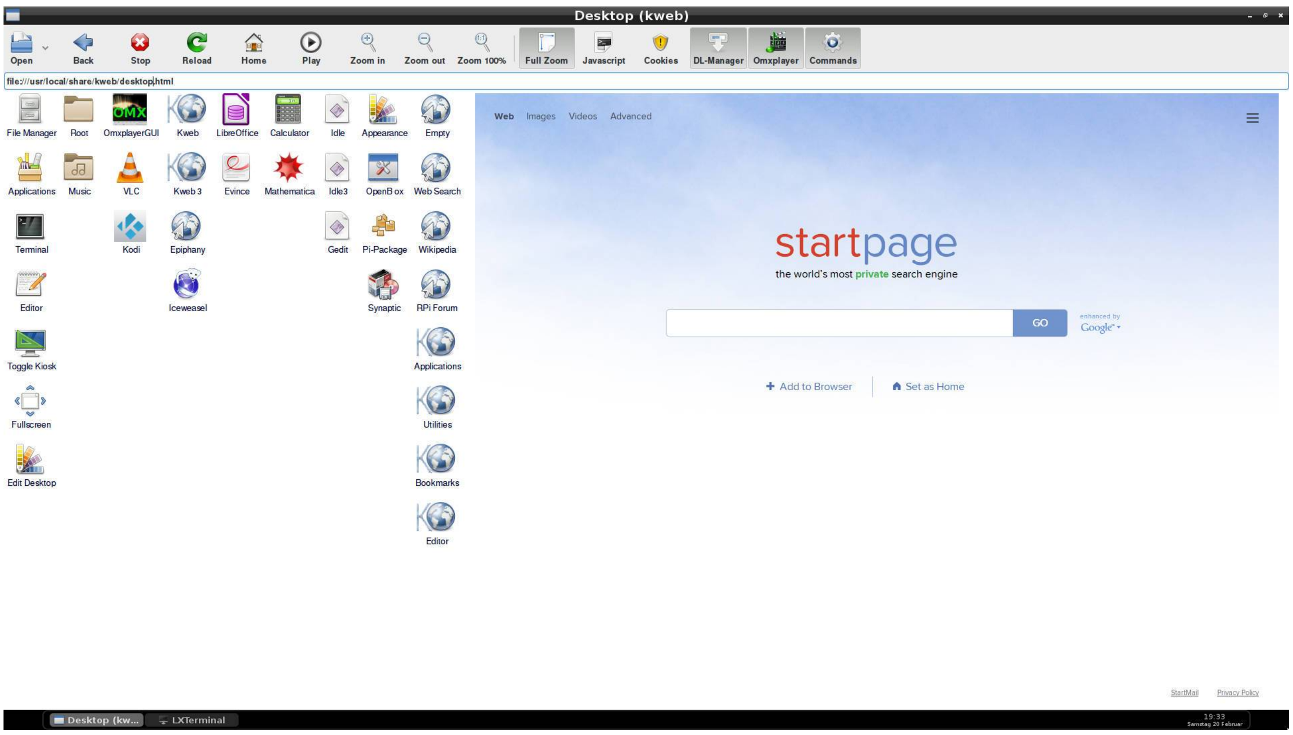
b) Styling the Environment

All web pages of the kweb environment use a common style (CSS file). This can be overwritten with four predefined styles by clicking on the appropriate buttons on kweb's "Utilities" page: "White" (default), "Grey", "Black" and "User".

The "user" style is by default the same as the "white" style. You can edit it ("Edit Style" from kweb's menu page) and modify fonts, font sizes and colours if you know a bit about CSS styling.

Part Three

Kweb's Application Interface and Application Examples



Example desktop page with embedded web area, from the tutorial in chapter 14. c)

12. Kweb's Text Editor and Text to HTML Compiler

a) Syntax of Kweb Source Text Files (Reference)

Introduction

The special kweb source text files are used to create simple (or more complex) HTML files that use the command links or hidden commands supported by kweb. They are extensively used in the kweb environment for user editable :command pages, user definable keyboard commands and the autoconfig page. They can also be used for developing embedded applications or kweb desktop pages.

The basic idea is that users can edit or create such pages without any knowledge of HTML and without understanding the special kinds of links supported by kweb. Developers of embedded applications can also use such files to insert the special command links or hidden commands into their HTML based application pages.

Although kweb source text files can be created and edited with any text editor like leafpad or gedit, it is recommended to use kweb's own editor kweb_edit.py (new in version 1.7.0); it offers some special functions and will automatically create the HTML file when you save the text file.

kweb text source files must use a ".txt" extension. The text2HTML compiler will create a file with the same name and replace the ".txt" extension with ".html".

Syntax

kweb's text to HTML compiler understands three different kinds of text lines:

Lines starting with "#!" are called "directives" and are used to set special options for the creation of HTML pages or to manage content insertion.

Lines starting with "#" followed by any kind of text will be inserted into the HTML page with the leading "#" removed. Such lines can also include any kind of HTML code (tags).

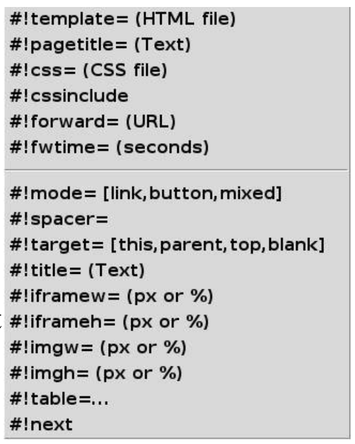
Lines starting with “&” or “@” are ignored (may be used for comments).

Lines of the form
name=value
will create kweb's special command links or hidden commands. We'll call them "instructions" from now on. “name” may be empty, e. g.
=value
This will create hidden commands or iframes containing another document.

Lines not matching any of these conditions will be ignored.

Directives

All directives are optional. If they are missing, default values are used. Kweb's text editor can insert any available directive from the right click pop-up menu.



```
#!template= (HTML file)
#!pagetitle= (Text)
#!css= (CSS file)
#!cssinclude
#!forward= (URL)
#!fwtime= (seconds)

#!mode= [link,button,mixed]
#!spacer=
#!target= [this,parent,top,blank]
#!title= (Text)
#!iframew= (px or %)
#!iframeh= (px or %)
#!imgw= (px or %)
#!imgh= (px or %)
#!table=...
#!next
```

The following directives should be used only once and best at the top of the text file:

#!template=
followed by the path to the HTML file used as template. It must be either a full path (starting with "/") or a file name if the file is in the same directory as the text file. If this directive is missing or has no value, a built-in template is used. This directive should only be used for development of embedded applications. For use in the kweb environment the built-in template should be used.

#!css=
followed by the path to the CSS file used for styling. It must be either a full path (starting with "/") or a file name if the file is in the same directory as the text file. If this directive is missing or has no value, no style will be used. This directive should only be used for development of embedded applications. For use in the kweb environment, styling will automatically be applied to :command pages and should not be changed.

#!cssinclude
This should only be used for embedded applications. The CSS file will be included into the HTML page as style tag.

#!forward=
followed by an URL. This will include page forwarding into the HTML page. It is optionally used for user definable keyboard command pages and can also be used for embedded applications. It's main use is to create HTML pages that execute a number of hidden commands and then go on to another web page.

#!fwtime=
followed by a number (seconds). Default value is "0" and should work in most cases. This value is used if you use page forwarding, and defines, how long the page should be shown, before forwarding occurs.

#!pagetitle=
followed by some text. This will include a title tag into your web page. You should use it if you create your own user editable :command pages for the kweb environment.

The following directives can be used multiple times:

#!mode=
followed by one of the following values: “link”, “button”, “mixed” (e. g. "#!mode=mixed")
It defines, how the command links should look like.
link: all commands are shown as normal links
button: all commands are shown as buttons
mixed: URL links are shown as links, all other commands are shown as buttons.
This can be changed at any time and will be used for the following "name=value" pairs.

#!spacer=
followed by nothing or some HTML entities or tags. It defines, what will be inserted between two command links. Spaces do also count. To have every link or button on a new line, use “<br>” as

spacer. The default value in kweb's predefined user editable command pages is " &nbsp;&nbsp; " which puts a certain distance between links or buttons.
This can be changed at any time and will be used for the following "name=value" pairs.

#!target=
followed by a certain value or empty. This will set the target of all following URL links (not used for other commands). Possible values are:
"this", "self" or empty: Open link in same frame (no target set).
parent: Open link in parent frame
top: open link in top most frame (replace current page).
blank: open link in new window.
Developers of embedded applications can also use any name of a named frame or iframe in their HTML template as target argument.
This can be changed at any time and will be used for the following "name=URL" pairs.

#!title=
followed by some text. This text will be shown in your HTML page when the mouse is moving over the link or button or image. Each "#!title" directive is only used once for the next "name=value" pair.
If the following command is a hidden command (“=value”), the text will be used as name of the resulting iframe. This way we can create named targets for use with the “#!target” directive.

#!iframew=
and
#!iframeh=
followed by a number (e.g. “200” or “200px”) or a percent value (e.g. “30%”). Default value for both is "2". This defines width and height of iframes used for hidden commands. Usually you don't want to change this. But there is one possible use in embedded applications: If a "hidden" command points to an URL you may want to display its content (not really "hidden") and you need a certain size for it. This can be used to cascade command pages or to construct toggle buttons without using JavaScript.
This can be changed at any time and will be used for the following "=value" instructions.

#!imgw=
and
#!imgh=
followed by a number (e.g. “200” or “200px”) or a percent value (e.g. “30%”). This defines the size (width or height or both) of image links. Default value for both is "0", which means that images are displayed with their original size.
This can be changed at any time and will be used for the following "image-path=value" instructions.

#!table=rows:columns:with:height:padding:vertical-alignment:horizontal-alignment
This directive is different from all others, as it creates a table, that has a content placeholder in each table cell. The argument is rather complex. Therefore the “#!table=...” item from the directives pop-up menu enters a default setting, which you can then modify:
```
#!table=1:1:0:0:4:top:center
```
rows, columns and padding must be numbers,
width, height may be either numbers (e. g. “100” or “100px”) or percent values (e. g. “30%”).
Vertical alignment may be: “top”, “middle” and “bottom.
Horizontal-alignment may be: left, center, right

The table must be inserted by a following “#!next” directive (see below) before we can add content into the table cells. It's also possible to create multiple tables before inserting them.
After inserting a table, all following content will be placed in the first table cell, until you use another “#!next” directive.

#!next
This directive may only be used for external templates that are built for it (see the "Templates" section below), after using the “#!table” directive or after inserting additional placeholders. It will collect all instructions built so far and insert them into the HTML template, replacing the first content placeholder. All following content will then be collected for the next placeholder. This can only work if the template or table contains multiple placeholders.
If you are using tables, each “#!next” directive goes to the next table cell, from left to right, and then goes to the next row. A table needs rows*columns-1 “#!next” directives to get completely filled.

Instructions

Instruction lines always have the form
name=value

"name" can be one of the following:
any text: will be used as name of the link or button created in your HTML file.

path or name of an image file, ending with ".png", ".jpg", ".gif" or “.ico”. This must be either a full path or the image file must be in the same directory as the text file. In this case, your link will be shown as that image. This allows the use of icons of any size.

Empty: If “name” is missing, "=value", a hidden command will be created. Hidden commands are iframes pointing to the URL created from the value. If value itself is an URL and not any kind of command, it will be a normal iframe displaying the content pointed to by the URL. But the normal use is to execute any kind of command, depending on the value given.

"value" may be one of the following:

an URL, starting with "http://", "https://" or "file://". A normal link pointing to that URL will be created. This is used by the bookmarks page, for example.
Links starting with 'rtp', 'rtmp', 'rtsp' or 'mmsh' are also allowed, but will be forwarded directly to omxplayerGUI.

A command line as you would type it into a terminal.
This will create a command link, that will execute the command line when you click the link (button, image), or a hidden command that will execute the command when the page is loaded. This way kweb can execute any kind of program or script either interactively (clicking a link or button) or automatically.

A question mark, followed by any number of keyboard command characters, e. g.
?z+++++
(This will set the zoom to 150%).
In hidden commands only the following keyboard values may be used: '+-zbhfpokgteduwxycqm#'

A text starting with one of the following characters: **")", "$","&","!","@"**
This will create a text-link, that executes the text in the same way, as when you enter it into kweb's entry line:
)URL - open URL in new window
))URL - open URL in new browser instance
!language-code(s) - e.g. "!en_GB" or "!en_GB,en_US"". Selects the language(s) for spell checking (dictionaries must be installed!).
$user-agent - user agent string to be sent by kweb
@encoding - default encoding to be used by kweb. The encodings are checked and eventually replaced by allowed encodings.
&CSS-file - must be either a "file:///.....css" link or a full path to a CSS file that will be applied as user CSS styling. Not allowed in hidden commands!

HTML templates

Developers of embedded applications can design a HTML page for their application and use it as template. A matching text file can be created and used to insert kweb's special commands into that template and create a new HTML file from it. They have to know, how this insertion works. The internal template used in the kweb environment looks like this:

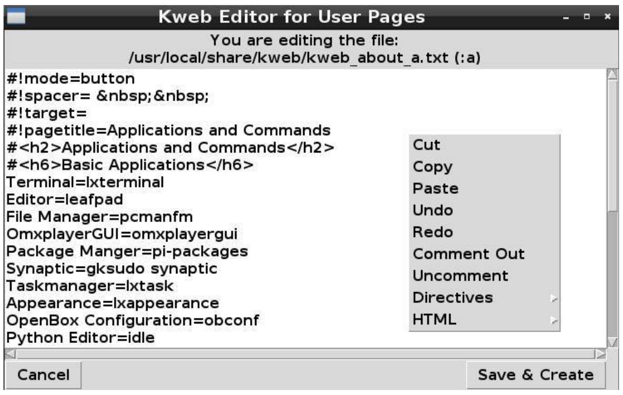
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta content="text/html; charset=UTF-8" http-equiv="content-type">
<meta http-equiv="cache-control" content="max-age=0" />
<meta http-equiv="cache-control" content="no-cache" />
<meta http-equiv="expires" content="0" />
<meta http-equiv="expires" content="Tue, 01 Jan 1980 1:00:00 GMT" />
<meta http-equiv="pragma" content="no-cache" />
<!--forward-->
<!--title-->
<!--style-->
</head>
<body>
<!--content-->
</body>
</html>
```

The <!--...--> tags are the values which are replaced by the text2HTML compiler:

<!--forward--> will be replaced with a page forwarding (if defined in the text file),
<!--title--> will be replaced with a title tag (if defined in the text file),
<!--style--> will be replaced with a CSS link or style tag (if defined in the text file),
<!--content--> will be replaced with the collected command links or hidden commands, including any additional HTML text.

It's possible to use multiple <!--content--> tags. Then the first one will be replaced when the compiler meets the first "#!next" directive, the next at the following "#!next" etc. For the last replacement you don't need a directive as the remaining collection of commands will be used.
 The replacement tags in the header are all optional; you don't have to use them. But at least one <!--content--> tag will be required or nothing will happen.

b) Kweb's Text Editor (User Guide)


Kweb Editor for User Pages
You are editing the file:
/usr/local/share/kweb/kweb_about_a.txt (:a)
Cut
Copy
Paste
Undo
Redo
Comment Out
Uncomment
Directives
HTML
Cancel
Save & Create

```
#!mode=button
#!spacer= &nbsp;&nbsp;
#!target=
#!pagetitle=Applications and Commands
#<h2>Applications and Commands</h2>
#<h6>Basic Applications</h6>
Terminal=lxterminal
Editor=leafpad
File Manager=pcmanfm
OmxplayerGUI=omxplayergui
Package Manger=pi-packages
Synaptic=gksudo synaptic
Taskmanager=lxtask
Appearance=lxappearance
OpenBox Configuration=obconf
Python Editor=idle
```

If you click on any of the “Edit” buttons at the bottom of user editable :Command pages or on the “:Editor” page, kweb's text editor will be opened (as root) and load the appropriate text. Although this text editor is really simple, it has two advantages compared to other editors:

1) When you save the modified text, the matching HTML page will automatically be generated.

2) From its right click pop-up menu you can insert all possible directives and a number of simple HTML entities.

Functions in the right click pop-up menu:

Cut, **Copy** and **Paste** work as in any other software (and the usual CTRL keyboard short-cuts are also supported).

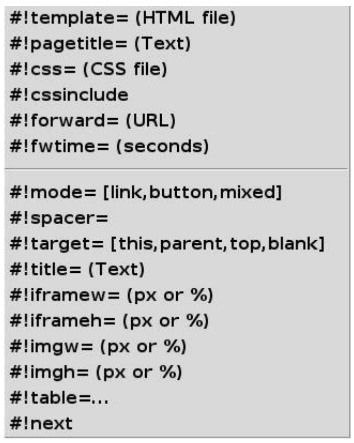
Undo: revert to the version before a last cut, paste, insert or delete actions. Multiple Undoes are possible. Can also be called with “CTRL+z”

Redo: revert the latest Undo step. Only possible if no manual changes have been applied after the latest Undo. Can also be called with “CTRL+SHIFT+z”

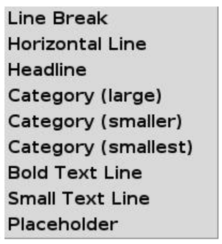
Comment Out: put a comment character “&” in front of a all selected lines. This will prevent these lines from being used.

Uncomment: remove the leading comment character “&” from all selected lines to enable their use again.

Directives: Opens a sub menu with a list of all possible Directives. The directives above the divider can be used only once, those below the divider can be used multiple times. If you select one of the items, it will be inserted into your text at the beginning of a line. If your cursor is already at the beginning of a line, it will be inserted there. If your cursor is somewhere inside a text line, a new line will be created after the current line. Most Directives require an argument that you must add.



HTML: opens another sub menu with some simple HTML entities to enter into your text. If your cursor is already at the beginning of a line, it will be inserted there. If your cursor is somewhere inside a text line, a new line will be created after the current line.



The last entry is special: it inserts an additional content placeholder into the template for use with the “#!next” directive.

All other entities insert HTML code functioning as a horizontal divider. If you select “Headline” or one of the following items, a small dialogue will be opened asking for the title text, that should be included, and text lines using different font sizes and styles will be inserted.

You can also call the editor from the command line (terminal!) to use it for your own applications:
```
[gksudo] kweb_edit.py [-f=size][text file path]
```
“size” (= font size) may be any value between 8 and 24.

The text file path must be either a full path or a file in your current working directory. The file name must have a “.txt” extension, which will be replaced with “.html” in the resulting HTML file.
If called as root (with gksudo), the working directory will be set to the directory of the kweb environment: /usr/local/share/kweb.

If called without a file path, a default content will be created as starting point and you can later select a file name when you save your file.

13. Creating Your Own :Command Pages

a) Introduction

If you enter something like
```
:name
```
into kweb's URL entry field, it is extended to
```
file:///usr/local/share/kweb/kweb_about_name.html
```
and kweb will try to load this local HTML page. The kweb environment uses one character names for all it's local pages, e.g. ":s" for the settings page, ":m" for its menu page etc. Only ":c" (browser configuration page) is handled differently, because this page is located in your user directory. You can easily extend this system, by moving your own HTML pages to that directory and follow the naming convention.

Note: Make sure, that you do not overwrite one of the system pages!

But it is much easier, to use kweb's ":command Editor" page (":e") to create your own :command pages without any knowledge of HTML coding. You can create user editable pages that contain links, command buttons or icons or hidden commands this way, similar to the predefined "Bookmarks", "Utilities" and "Applications" pages.

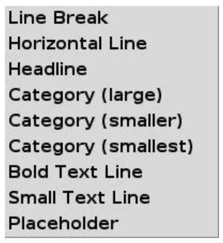
The following tutorial will show you, how you can add an Internet radio page to your kweb environment. It's just an example, but you can learn how to add a user editable page which fits your own needs.

b) Tutorial: Creating an Internet Radio Page

If like to use Internet radio, you might like to add a user editable page to the kweb environment which gives you easy access to your favourite radio streams and websites.

On kweb's menu page select ":Editor". In the "Name" field enter "r" and click "Create Command" (the page later can be opened with ":r" from the URL entry field).

Kweb's text editor will open with a few lines of default text. Modify them to look like this:
```
#!mode=mixed
#!spacer= &nbsp;&nbsp;
#!target=top
#!pagetitle=Internet Radio
#<h2>Internet Radio</h2>
```



Now we start adding sections using the right click pop-up sub-menu "HTML". Place the cursor at the bottom and select "Horizontal Line". Then select "Category (smaller)" and enter "Streams" in the text input field. We will get two new lines looking like this:
```
#<hr>
#<h5>Streams</h5>
```
Repeat these steps three more times, using the following category names: "Playlists", "Websites", "Applications", which will add the following lines_
```
#<hr>
#<h5>Playlists</h5>
#<hr>
#<h5>Websites</h5>
#<hr>
#<h5>Applications</h5>
```

Now we can start adding content to the sections. This is completely up to you. I'll just show a few examples. In the “Streams” category (below the category name) I've added:

```
Bob Dylan Radio=http://streaming208.radionomy.com:80/BobDylanRGRadio
Pink Floyd=http://listen.radionomy.com/shineonradio
```

In the “Playlist” section I'll add “file://”-type URLs to locally saved playlists:

```
Classic Rock Miami=file:///home/pi/CLASSIC%20ROCK%20MIAMI.pls
Rock the Folk=file:///home/pi/RocktheFolk.pls
```

Empty spaces in file paths must be replaced with “%20”.
In the “Websites” section I've added links to Internet radio websites:

```
Xiph=http://dir.xiph.org/
Radio Browser=http://www.radio-browser.info/gui/
Internet-Radio.com=http://www.internet-radio.com/
```

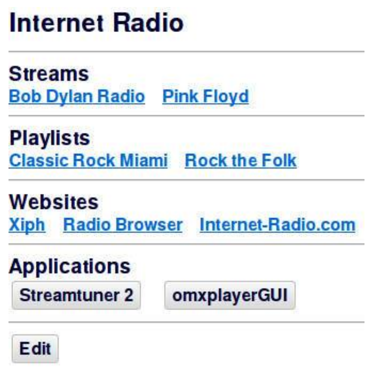
So far we have only used “name=URL” lines, which will simply create links. In the last section, “Applications”, we will add command buttons to start programs:

```
Streamtuner 2=streamtuner.pyz
omxplayerGUI=omxplayergui
```

(I have downloaded “Streamtuner 2” from Source Forge, renamed the “.pyz” file to “streamtuner.pyz”, made it executable and moved it to “/usr/local/bin”.)

Now click “Save & Create” and open the page in kweb with “:r”. You can also open (or reload) the “:Editor” page and find a new entry among the user-editable pages and click on the “:r” link.

All user-editable pages have an “Edit” button at the bottom. Clicking it will open kweb's text editor and you can add or remove content.



14. Desktop Home Page Examples

a) Introduction

When I started development of kweb I had one goal in my mind: I wanted to run a simple desktop environment using the browser as front end. In fact, I'm using it as such since version 1.0.

What does a simple Desktop require? A kind of file manager, the ability to access and run installed programs and a small toolbar to switch between windows of running applications. With a browser as front end the Internet is also at your finger tips. And I'm running a number of servers on my network (media servers, a server to manage and access satellite TV among them) which are also directly accessible this way.

You can easily test how this works without changing anything on your system; the usual LXDE desktop is still available and and we don't change any of its settings. The only thing you have to do is to install “tint2”, a very simple tool bar which won't interfere with your system:

```
sudo apt-get install tint2
```

Now either boot to the command line or – if you boot into the desktop and do not want to change that – close the desktop and return to the command line. Run the following command:

```
xinit ./ktop
```

This will start an X-Windows environment using the OpenBox window manager, tint2 as taskbar and kweb with its menu page as default home page. This is not the ideal environment, but it will show you what is possible.

How to run programs?

1) On the menu page, click on “Applications”. It already has a number of buttons for standard programs (terminal, editor, file manager etc.) and commands like shut down, reboot, update, upgrade etc. Just one mouse click will start the applications or run those commands. We've already shown how you can add you own commands to the application page using the edit function and adding lines like:

```
name=command
```

2) You can also use kweb's URL entry field as command line to start programs:

```
#lxterminal
```

will start the the terminal, for example.

3) You can use the Debian menu to start almost all installed GUI application. Right clicking into any empty screen area (e. g. besides the tint2 toolbar) will open it.

4) Start the file manager (button on the application page) and select “Applications” on the left side.

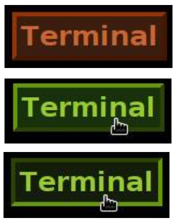
How to access files?

1) kweb can open a lot of file types by itself, using the open command: HTML and text files and some image types (JPEG, PNG, GIF) can be shown inside the browser, PDF files will be opened with your preferred PDF program and all types of media files (audio, video, playlists) will open in omxplayerGUI.
2) You can use LXDE's standard file manager (with a few restrictions). Just open it from the application page.

If you close the browser, the X-Windows desktop session ends and you will return to the command line.

This should give you an idea what is possible, but as I already noted, kweb's menu page is not the best choice to be used as desktop environment. The following examples may help you to design your own home page for desktop use or other kinds of interfaces.

b) Tutorial 1: Simple Buttons



This very simple desktop home page example uses coloured button-like links on a black background. The "buttons" are rather large, suitable to be viewed and used from a certain distance. The page is divided into five categories: "Services", "Internet", "Applications", "Tools" and "Configuration". If you don't like them, you can easily change their names in the HTML template file, but for this example we'll use them as they are.

Now we want to create the page. From a terminal run
```
gksudo kweb_edit.py
```
Kweb's editor will open and the initial text will look like this (default text for a new project):
```
#!mode=mixed
#!spacer=
#!target=top
#!pagetitle=
```
We have to edit it a bit. Change the first line to
```
#!mode=link
```
and the second to
```
#!spacer= &nbsp;&nbsp; 
```
(with an empty space at the end).
In the 4th line we'll add a page title, e.g.
```
#!pagetitle=My Home Page
```
We have to add one more directive. Place the cursor at the end, right click and from the pop-up menu select "Directives" and then "#!template=". This will insert the template directive and we will have to add the name of the template:
```
#!template=buttons_template.html
```

Now we are ready to add content to the first category ("Services"). This might be used for local web servers, software web frontends (like tvheadend, VDR, some torrent clients etc.). As an example I'll add some of my own stuff here:
```
Media Server=http://localhost:9079/
TV-Server=http://localhost:9081/
```
It completely depends on your own environment what you are entering here. Those are all links and it's important to have http:// or https:// in front, otherwise they won't be recognized as such.
If we are through with the first category, we add one more directive (from the pop-up menu or manually):
```
#!next
```
Now we can start with the second category, "Internet", by adding links like that
```
Youtube=https://www.youtube.com
RPi Forum=https://www.raspberrypi.org/forum
```
etc. This is not your bookmarks collection, so you should only add the websites that you visit very often. To get easy access to your full bookmarks list, you could add:
```
Bookmarks=file:///usr/local/share/kweb/kweb_about_b.html
```
If you have finished entering your most important websites, add another
```
#!next
```

directive. The next category ("Applications") will be used to add the programs or scripts you always want to have available. They are entered as "name=command line", e. g.
```
Terminal=lxterminal
Editor=leafpad
File Manager=pcmanfm
Kodi=kodi-standalone
```
etc. GUI programs that have to be started as root, should use "gksudo", e.g.
```
Synaptic=gksudo synaptic
```
Programs which are not found directly have to be entered with full path. My favourite Pinball game, for example, is opened with
```
Pinball=/usr/local/bin/indiecity/InstalledApps/mame4all_pi/Full/mame pbactio2
```
Programs have to be entered as you call them from inside a terminal. If arguments require quotes or double quotes, you'll also have to enter them here.

Note: Instead of entering everything by hand, you can open kweb's "Application" page, click on "Edit" and use copy and paste for some of the programs.

To go on to the next section, "Tools", we have to enter another
```
#!next
```
directive. We add stuff in a similar way as in the last section, e.g.
```
Top=top
Taskmanager=lxtask
Shutdown=sudo shutdown -h now
Reboot=sudo reboot
Update=sudo apt-get update
Upgrade=sudo apt-get upgrade
```
One more
```
#!next
```
directive will lead us to the last category, "Configuration". Here are some matching example entries:
```
Edit Bootconfig=gksudo leafpad /boot/config.txt
Raspi-Config=sudo raspi-config
Appearance=lxappearance
OpenBox Configuration=obconf
Control Panel=file:///usr/local/share/kweb/kweb_about_p.html
Edit Homepage=gksudo kweb_edit.py homepage.txt
```

As this is the last category we do not have to enter another "#!next" directive. Now it's time to save the text and create our home page. Click on "Save & Create". A file browser will open, pointing to the directory "/usr/local/share/kweb". Do not change this! (We have been using no full paths so far.) As file name enter "homepage.txt" and click "Save". If you restart the browser, your new home page will be opened. To start it from the command line (not from ta terminal!), you can use:
```
xinit ./ktop
```

You can edit the home page at any time using:
```
gksudo kweb_edit.py homepage.txt
```

Until now we have only used either links or commands, but it is also possible to add browser configuration options. Go to kweb's "Utilities" page, click "Edit" and use copy and paste to insert some of the example options from there, like setting a user agent or similar.

It's also possible to add some of the powerful user definable functions to your home page. In chapter 10. d) I have shown how to create a short-cut for ALT+1, which sets your user agent, starts the youtube-dl-server and then goes to youtube.com. To add this to your home page, e. g. in the

“Internet” section, just enter a line like that:
```
Youtube=file///user/local/share/kweb/kweb1.html
```

This is a very simple kind of home page, but carefully designed it should give you immediate access to 98% of places (services, Internet) and programs you are using with just one mouse click. Programs you seldom use are still available using the file manager.
Of course the categories used may not be the best choice for everyone, but it's easy to change them. Run
```
gksudo leafpad /usr/local/share/kweb/buttons_template.html
```
find the category names (they are table headers), change them to anything that suits your needs and save the file.

c) Tutorial 2: Desktop with Icons and Integrated Web Area

In this example I'll show you how to create a home page that looks a little bit more like a desktop, using named icons, but with a very special feature: A part of the desktop will be used to display selectable web content. This example is configured for a full HD screen. The complete example is part of the kweb package and you can open the text file from a terminal with:
```
gksudo kweb_edit.py desktop.txt
```
If you just want to have a look at the result, click on “Save & Create”. The editor will close and compile the page. You can open it from kweb with “file:///usr/local/share/kweb/desktop.html”. If you click on “Edit Desktop” (last icon in the first column), kweb's editor will open again with the text file, from which this page has been been created.

Note: All directives and HTML content in this example have been created by using the editors right click pop-up menu. No special knowledge of HTML coding is required.

Now we'll have a closer look at the text and how this page is built. The top of the text looks like this:
```
@ kweb desktop example page
@ Tutorial: Kweb Manual, chapter 14. c)

#!mode=mixed
#!spacer=
#!target=top
#!pagetitle=Desktop
```
Lines starting with “@” are comments. The first instructions set some default values. The next line is
```
#!css=about.css
```
and lets the page use the common default style of all pages of the kweb environment and so also supports its simple “skinning” from the “Utilities” page (white, grey, black, user).
The following line
```
#!imgh=48px
```
sets the image height of all following images to 48 pixels. For this example I have mostly used 48x48 pixel icons from the Adwaita theme, but also some applications icons that may have a different size. This setting will make sure, that all icons on the desktop have the same height.
The next directive
```
#!table=1:10:100%:100%:5:top:center
```
creates a table consisting of 10 columns stretched across the whole screen area. Each table cell (column) has its own content placeholder. We will use the first 9 columns for desktop icons and the last one for an embedded web area.
The next line inserts the table and all content following it will be placed in the first table cell:
```
#!next
```

Now we'll start adding content to the first column:
```
@elements of column 1
#!title=File Manager
/usr/share/icons/Adwaita/48x48/apps/system-file-manager.png=pcmanfm
#<br><small>File Manager</small><br>
#<br>
```
The first directive will create a tool tip text which appears when the mouse is hovering over the icon. The second line creates an icon which will start a program (LXDE's file manager pcmanfm). The left side is a the full path the icon being used and the right part after the equal sign is the command to be executed.
The last two lines contain HTML code created from the editors popup menu, using the entities "Small Text Line" (to give the icon a name displayed below it) and "Line Break" (to add more distance between icons).

All icons on the desktop page are built the same way. Let's have a look at the next icons of the first column:
```
#!title=Application Manager
/usr/share/icons/Adwaita/48x48/categories/applications-other.png=pcmanfm
menu://applications/
#<br><small>Applications</small><br>
#<br>

#!title=Terminal
/usr/share/icons/Adwaita/48x48/apps/utilities-terminal.png=lxterminal
#<br><small>Terminal</small><br>
#<br>

#!title=Text Editor
/usr/share/icons/Adwaita/48x48/apps/accessories-text-editor.png=leafpad
#<br><small>Editor</small><br>
#<br>
```
The first one will open the file manager's application page, the other ones the default terminal and text editor.

The next two icons make use of kweb's keyboard controls to toggle kiosk mode (hide or show the toolbar) or between full screen and window mode:
```
#!title=Toggle Kiosk Mode
/usr/share/icons/Adwaita/48x48/apps/preferences-desktop-display.png=?k
#<br><small>Toggle Kiosk</small><br>
#<br>

#!title=Toggle Fullscreen
/usr/share/icons/Adwaita/48x48/actions/view-fullscreen.png=?f
#<br><small>Fullscreen</small><br>
#<br>
```

The last icon of the first column will open kweb's text editor to modify the content of the desktop page:
```
#!title=Edit Desktop
/usr/share/icons/Adwaita/48x48/apps/preferences-desktop-theme.png=gksudo
kweb_edit.py desktop.txt
#<br><small>Edit Desktop</small><br>
#<br>
```
(four lines in the original text; the second line was too large for the manual's layout!).
To start adding content to the second column we have to add another:
```
#!next
```

I've used the second column for places, using the file manager with matching arguments:
```
@elements of column 2
#!title=Open Root as Root
/usr/share/icons/Adwaita/48x48/places/folder.png=gksudo pcmanfm /
#<br><small>Root</small><br>
#<br>

#!title=Downloads
/usr/share/icons/Adwaita/48x48/places/folder-download.png=pcmanfm Downloads
#<br><small>Downloads</small><br>
#<br>
```

The third column is used to access media players:
```
#!next
@elements of column 3
#!title=OmxplayerGUI Standalone
/usr/share/pixmaps/omxplayergui.png=omxplayergui
#<br><small>OmxplayerGUI</small><br>
#<br>

&#!title=VLC Media Player
&/usr/share/icons/hicolor/48x48/apps/vlc.png=vlc
&#<br><small>VLC</small><br>
&#<br>

&#!title=Kodi Media Player
&/usr/share/kodi/media/icon48x48.png=startkodi
&#<br><small>Kodi</small><br>
&#<br>
```
You will note, that the last two blocks start with a “&”. This is used for disabling (“commenting out”) content. If you have installed VLC player on your system, you can select the matching four lines and call the “Uncomment” function from the pop-up menu. This will remove the leading “&” from all selected lines.
If you have a kodi version for Raspbian installed, you can do the same with the last four lines. If you are not using one of the versions I have supplied, you also have to replace “startkodi” with “kodi-standalone”.

I've used the fourth column for browsers (or other internet applications):
```
#!next
@elements of column 4
#!title=Kweb Full Power Browser
/usr/share/pixmaps/minimalkioskbrowser.png=kweb -ZJEHPBFQ5+-
zbhrqfpoklgtjneduwxyavcsmi#?!., file:///usr/local/share/kweb/kweb_about_m.html
#<br><small>Kweb</small><br>
#<br>

#!title=Kweb3 Full Power Browser
/usr/share/pixmaps/minimalkioskbrowser.png=kweb3 -ZJEHPBFQ5+-
zbhrqfpoklgtjneduwxyavcsmi#?!., file:///usr/local/share/kweb/kweb_about_m.html
#<br><small>Kweb 3</small><br>
#<br>

#!title=Epiphany Browser
/usr/share/icons/Adwaita/48x48/apps/web-browser.png=epiphany-browser
#<br><small>Epiphany</small><br>
#<br>

&#!title=Iceweasel Browser
&/usr/share/pixmaps/iceweasel.png=iceweasel
```

```
&#<br><small>Iceweasel</small><br>
&#<br>
```
The first two start kweb or kweb3 with “Full Power Browser” settings, using kweb's menu page as home page. The last block is commented out again, because not everyone has installed Iceweasel-

The fifth column is used for typical office applications and document readers:
```
#!next
@elements of column 5
#!title=Office Suite
/usr/share/icons/gnome/48x48/apps/libreoffice-base.png=libreoffice
#<br><small>LibreOffice</small><br>
#<br>

#!title=Evince PDF Viewer
/usr/share/icons/hicolor/48x48/apps/evince.png=evince
#<br><small>Evince</small><br>
#<br>
```

The sixth column might be used for mathematical or scientific applications:
```
#!next
@elements of column 6
#!title=Calculator
/usr/share/icons/Adwaita/48x48/apps/accessories-calculator.png=galculator
#<br><small>Calculator</small><br>
#<br>

#!title=Mathematica
/usr/share/icons/hicolor/64x64/apps/wolfram-mathematica.png=mathematica
#<br><small>Mathematica</small><br>
#<br>
```

Column seven is used for software development:
```
#!next
@elements of column 7
#!title=Python 2.7 Editor
/usr/share/icons/Adwaita/48x48/mimetypes/text-x-script.png=idle
#<br><small>Idle</small><br>
#<br>

#!title=Python 3.4 Editor
/usr/share/icons/Adwaita/48x48/mimetypes/text-x-script.png=idle-python3.4
#<br><small>Idle3</small><br>
#<br>

&#!title=Source Code Editor
&/usr/share/icons/Adwaita/48x48/mimetypes/text-x-script.png=gedit
&#<br><small>Gedit</small><br>
&#<br>
```

I've used the eighth column for system settings and application managers:
```
#!next
@elements of column 8
#!title=Appearance Settings
/usr/share/icons/Adwaita/48x48/apps/preferences-desktop-theme.png=lxappearance
#<br><small>Appearance</small><br>
#<br>

#!title=Openbox Configuration
/usr/share/pixmaps/obconf.png=obconf
```

```
#<br><small>OpenBox</small><br>
#<br>

#!title=Raspberry Pi Package Manager
/usr/share/icons/Adwaita/48x48/apps/system-software-install.png=pi-packages
#<br><small>Pi-Package</small><br>
#<br>

&#!title=Synaptic Package Manager
&/usr/share/pixmaps/synaptic.png=gksudo synaptic
&#<br><small>Synaptic</small><br>
&#<br>
```

Column nine is different. We use it for links to HTML content (web or local) that should be displayed in the web area we'll create in column 10. It starts with:

```
#!next
@elements of column 9: Links to web area
#!target=area
```

The last directive is used to let all following links point to the web area we we'll create in column 10.

The first icon in this column points to an “empty” page, which we'll use if no web content should be displayed on our desktop:

```
#!title=Empty
/usr/share/icons/Adwaita/48x48/apps/web-
browser.png=file:///usr/local/share/kweb/empty.html
#<br><small>Empty</small><br>
#<br>
```

The next 3 icons point to websites you might often use:

```
#!title=Web Search
/usr/share/icons/Adwaita/48x48/apps/web-browser.png=https://www.startpage.com/
#<br><small>Web Search</small><br>
#<br>

#!title=Wikipedia
/usr/share/icons/Adwaita/48x48/apps/web-browser.png=https://www.wikipedia.org/
#<br><small>Wikipedia</small><br>
#<br>

#!title=Raspberry Pi Forum Forum
/usr/share/icons/Adwaita/48x48/apps/web-
browser.png=https://www.raspberrypi.org/forum
#<br><small>RPi Forum</small><br>
#<br>
```

The last four icons give access to some pages of the kweb environment:

```
#!title=Kweb Bookmarks Page
/usr/share/pixmaps/minimalkioskbrowser.png=file:///usr/local/share/kweb/kweb_about_b.html
#<br><small>Bookmarks</small><br>
#<br>

#!title=Kweb Utilities Page
/usr/share/pixmaps/minimalkioskbrowser.png=file:///usr/local/share/kweb/kweb_about_u.html
#<br><small>Utilities</small><br>
#<br>

#!title=Kweb Applications Page
/usr/share/pixmaps/minimalkioskbrowser.png=file:///usr/local/share/kweb/kweb_about_a.html
#<br><small>Applications</small><br>
```

```
#<br>

#!title=Kweb :Editor Page
/usr/share/pixmaps/minimalkioskbrowser.png=file:///usr/local/share/kweb/kweb_about_e.html
#<br><small>Editor</small><br>
#<br>
```

Now we'll create the web area in column 10:

```
#!next
@elements of column 10: Web Area
#!iframew=1200px
#!iframeh=100%
```

The last two lines define the size (width in pixels, height in %) of the web area (iframe).

```
#!title=area
```

When used with embedded frames (iframes), the "`#!title`" directive is not used for tool tips but gives the iframe a name which can be used as target (we have already used this target name in column 9).

```
=file:///usr/local/share/kweb/empty.html
```

This creates the web area (iframe) with the size and target name defined above, pointing to an empty page by default. If you click on any of the icons of column 9, the matching web (or local HTML) content will appear inside the web area.

Note: not all websites allow being displayed inside an iframe!

This rather extensive example has been added for teaching purposes, but if you like it, you may well use it as starting point to create your own desktop page. In this case you should create a copy using another file name, because the file "desktop.txt" will be overwritten if you install an update of kweb:

```
cd /usr/local/share/kweb
sudo cp desktop.txt mydesktop.txt
```

or any name you like. You might also use the file name "homepage.txt" if you want your desktop file to be taken as default home page.
Now open it with kweb's editor's

```
gksudo kweb_edit.py mydesktop.txt
```

and modify the file name in the "Edit Desktop" icon definition.
Now you can start adding, modifying or removing content.

You can also add icons pointing to documents. For PDF files or all kinds of media files (audio, media, playlists) you can add lines like that:

```
icon path=file:///home/pi/my.pdf
```

For other kinds of data, you have to include the program to open it, e.g.:

```
icon path=libreoffice /home/pi/mydoc.odt
```

This should be enough to get you started with your own experiments.

d) Adding Background Images

For some people a desktop is not a desktop if it doesn't use an image or "wallpaper" as background. We can easily add this to the desktop example from the last chapter.
You should first decide, which image you want to use. On top of dark images, you'll need white text, for example, on top of light images a dark text colour. We have to create a modified CSS file. If you use a light image, create a copy of the white.css file (white background, dark text):

```
cd /usr/local/share/kweb
sudo cp white.css bg.css
```

For use with dark images you should start with a copy of “grey.css” or “black.css”.

Now we have to edit the file “bg.cs” a bit. Let's assume, you want to use a file “/home/pi/wallpaper.png” as background image. We open the css file with leafpad:

```
gksudo leafpad /usr/local/share/kweb/bg.css
```

In the section starting with “body” (near the top), remove the line starting with “background-color:”. Add the following lines instead

```
background-image: url(file:///home/pi/wallpaper.png);
background-size: 100%;
background-attachment:fixed;
```

And save the file.

Now open your desktop.txt page with kweb_edit.py, as shown before, and modify one directive at the beginning to:

```
#!css=bg.css
```

Click “Save & Create” and your desktop page should now use the selected background image.

e) Using Kweb as Desktop Replacement

If you have built your own desktop page (following one of the tutorials above, for example) you can use it as an alternative to the usual LDXE desktop. We do not fiddle around with any LXDE settings; the normal desktop can still be used and started with “startx”.

Note: All these examples require “tint2” to be installed.

Set your Raspberry Pi to boot boot to the command line. If your desktop file is named “homepage.html” and resides in the kweb directory, you can simply start it by running

```
xinit ./ktop
```

from the command line. This will start X-ORG, the window manager openbox, the small taskbar tint2 and kweb with your desktop home page. If you close kweb, you will return to the command line (and all applications opened from your desktop page will also be closed).

If you use a different name for your desktop page, you have to activate the “Set URL as home page (H)” option on the browser configuration page (and all other options you might want to use) and add the full path or “file://”-URL in the entry field at the bottom, e. g.

```
file:///usr/local/share/kweb/desktop.html
```

Save it and then you can also use the “xinit ./ktop” command to start it.

You can also create your own start script. This is useful if you want to use different home pages when you start kweb as desktop environment or from the LXDE environment, for example.

The following script acts a bit differently from “ktop”. X-ORG is not closed when you close kweb. You have to select “Exit” from the openbox right click menu to close it. Open leafpad or nano and enter:

```
#!/bin/sh
tint2 &
openbox --config-file ~/.config/openbox/rc.xml --startup "kweb -HZQU4+-
zbhrqfpoklgtjneduwxyavcsmi#?!., file:///usr/local/share/kweb/desktop.html"
```

Save the file as “/home/pi/kdesk” and make it executable:

```
chmod +x kdesk
```

Now you can start it from the command line with:
```
xinit ./kdesk
```

You may use your own selection of options, of course. Just make sure, that all keyboard short-cuts (`+-zbhrqfpoklgtjneduwxyavcsmi#?!.,`) are part of the options string.

If you want to boot into your kweb desktop (booting to the command line must be selected in raspi-config!), run:
```
sudo nano /etc/rc.local
```
Just before the last line ("exit 0") add two lines:
```
sleep 10
su -l pi -c "xinit ./kdesk"
```
(or "ktop" at the end if you want to use it) and save the file.
The "sleep" command has to be used in Jessie, as rc.local may be executed too early by systemd at boot time. If "sleep 10" doesn't work, try a higher number.

15. Using Kweb as Front End for Embedded Applications

Introduction

Normally you'll look for a (kiosk) browser as a front end for an embedded application if this application is backed by an http (application) server, which does the actual work, while the browser supplies the user interface. Older Popcorn Media Players are an example of such configurations.

With kweb, this is a bit different. Of course, it can also be used in this way (which gives unlimited possibilities), but because of kweb's special features you can realize a lot of applications without needing a server at all. You simply create one or more html files and use kweb's built-in command options (as special kinds of links, buttons etc.) to execute your application commands. Sometimes you may have to add a few simple bash or Python scripts, which may be used by your application, perhaps running in the background. It's even possible to run commands without user interaction (e. g. when loading a page). Switching between different settings for the helper programs (presets) is also possible as part of your application.

Kweb supports three special kinds of links, which can execute commands as if they were issued from the command line, execute keyboard short-cuts or configure the browser as you could do it by entering special text commands into the browser's URL entry field. If you don't use links but (small, invisible) iframes pointing to the command URLs (as "src" attribute), commands can be executed automatically when a page loads or the "src" attribute is manipulated by JavaScript.

Kweb also supports special kinds of web forms which can be used to let the user configure command arguments (e. g. from a menu or text input field).

This all works with plain HTML files. You do not need any server side language like PHP. There's one great difference, though: the commands executed don't return something (don't go to a new web page as a result of your command). But it is possible to simulate such behaviour by using (bash or Python) scripts that use xdotool to manipulate the browser. In some simpler cases you can also use page forwarding instead.

One more interesting thing might be mentioned here, which already has been used in a number of applications. I call it "simulated web video". Instead of using the limited HW acceleration for video built into the webkit3 engine (used by epiphany and kweb3), omxplayer can be used in such a way that it looks like the video is playing inside the browser window, giving much better video performance and quality.

Although some simple things can be created without any knowledge of HTML by using kweb's text editor (as has been shown in previous chapters), a developer of an embedded application will usually create more complex HTML pages and then include kweb's special features (as links, iframes, forms etc.). This can be done in two ways: either manually or by including placeholders and using kweb's text editor to create the special content and insert it into the HTML pages, which are used as templates in this case.

Starting in Kiosk Mode

You'll start your application from the command line (of course you can also boot into it) with:
xinit ./kiosk
where "kiosk" (you can use any name you want) is a small, executable script file in the root of your

user directory. A very simple example may look like that:

```
#!/bin/sh
matchbox-window-manager &
kweb -KHCUA+-zbhrqfpoklgtjneduwxyavcsmi#?!., file:///usr/local/share/kweb/kweb_about_a.html
```

This requires the small and simple window manager matchbox-window-manager to be installed. It's a good solution if only one window at a time (on top, full screen) will be used by your application. The second line of the scripts starts it (in the background). Most applications will need a window manager to run.

The third line of the script starts kweb with a certain set of options:
Use kiosk mode, use URL as home page, enable command interface, no cache (we are using files only) and a list of all keyboard commands allowed in combination with the ALT key. It's up to you to reduce the keyboard commands to a subset you want to use or to disable them completely. As home page we use the application page example that comes with the kweb installation.

The “C” option is the only one, that only works from a command line and not from a configuration file. It definitely enables command execution. In most cases this is not really needed, because since version 1.7.0 command execution is enabled by default, whenever it is considered to be secure. But it is good practice to always add the “C” option for embedded applications.

You may also have to create a settings configuration best suited to your application (and save it as a preset).

For security reasons application commands can only be used from inside HTML files, not from web pages served via HTTP. There is an option that allows to overrides this and maps the command interface to a server running on localhost (only). You can open “http://”-web pages from links in HTML files but not “file://”-links from inside web pages served via “http://”. The best way to add web content to your application is therefore by using iframes inside HTML files.

This should give you a basic idea how to create kiosk applications based on files only. Of course, you can use other window managers as well, depending on the needs of your application.

There's one such example installed with kweb into your user directory, named “ktop” (short for kweb desktop). It is used to open kweb in desktop mode and looks like this:

```
#!/bin/sh
openbox --config-file ~/.config/openbox/LXDE-rc.xml --startup tint2 &
kweb
```

It starts the window manager OpenBox, using it's default LXDE settings (so we can use both the appearance and OpenBox tool for styling it), tint2 as task bar and finally kweb without a special configuration (it will take its configuration from a configuration file if it exists). From the command line start it with:

```
xinit ./ktop
```

Remote Control of Kweb for Presentation Applications

kweb can now be controlled from a script using xdotool. This will only work with the matchbox-window-manager; other window managers I've tested don't work well with xdotool. You can send all keyboard commands to kweb and also URLs which it should display. Here's a simple example script, which goes to a certain vimeo.com video page, starts the video (with omxplayerGUI) by issuing the “play” command, waits for the video to finish and then closes the browser.

```
#!/bin/sh
matchbox-window-manager &
kweb -KAC+-zbhrqfpoklgtjneduwxyavcsmi#?!., &
sleep 5
xdotool key alt+i type "http://vimeo.com/110416910
"
sleep 5
echo "" > ~/.omxplayergui.run
xdotool key alt+p
while [ -f ~/.omxplayergui.run ]
do
sleep 2
done
xdotool key alt+q
```

Start it from the command line with
```
xinit ./scriptname
```
(the script must be in the root of your home directory).
The script creates a file “~/.omxplayergui.run” which is deleted by omxplayerGUI when it finishes. This can be used by the script to decide when the video has stopped playing.

Check the xdotool documentation for more details

It's also possible to start scripts using xdotool from inside a web page (hidden commands). Usually such scripts will run inside a terminal, but we can avoid that by adding them to the “direct_commands” list on the settings page.

Kweb's Special Command Links and Command Forms

Kweb supports three kinds of command links and a number of special kinds of command forms for more complex operations.

Command links and command forms all start with a dummy URL “file:///homepage.html?”, a file that does not even exist and will never be opened by the browser. And the “command links” and the submit button of a command form will not open a new page at all (the page remains as it is), but execute a command (run a script, start a program, execute keyboard commands, modify browser settings etc.).

Keyboard Commands

The first kind of command links executes browser commands, the same you can call from the keyboard (see the list of keyboard short-cuts). The following link (appearing as a button) will close the browser (end your application) when clicked:
```
<a href="file:///homepage.html?kbd=q"><button>Quit Browser</button></a>
```
All possible keyboard commands may be used (not all of them make sense), but you can restrict their use with the command line options (see the example above).
The href attribute of the link always starts with
```
file:///homepage.html?kbd=
```
followed by any number of command characters (keyboard short-cut).
Keyboard command links only work if the matching keyboard short-cuts are enabled!

Command line commands

The second kind of command link is simply a command line command like this one:
```
sudo shutdown -h now
```
turned into a link like that:
```
<a href="file:///homepage.html?cmd=sudo%20shutdown%20-h%20now">Shutdown</a>
```
which will create a link named "Shutdown" and clicking on that link will shut down your Raspberry Pi.
The href attribute of such links always starts with:
```
file:///homepage.html?cmd=
```
followed by the command line with all spaces replaced with "%20".

Special characters may have to be escaped with their %-code. And (quoted) arguments containing spaces must be handled differently. For example the following command line
```
gksudo "kweb_edit.py -f=14 desktop.txt"
```
has to be converted to the following href attribute:
```
file:///homepage.html?cmd=gksudo%20%22kweb_edit.py+-f%3D14+desktop.txt%22
```
That's not always an easy task and I recommend using kweb's text editor to create the appropriate links.

Text commands

The third kind of command links inserts text into kweb's URL entry field (even if it's hidden in kiosk mode) and executes them in the same way as if you enter them manually (see the chapter about kweb's multifunctional entry field for more details). The following example link sets the user agent of kweb to Firefox if you click the button:
```
<a href="file:///homepage.html?txt=$Mozilla/5.0%20%28X11%3B%20Ubuntu%3B%20Linux
%20i686%3B%20rv%3A43.0%29%20Gecko/20100101%20Firefox/43.0"><button>Firefox
Desktop</button></a>
```

The href attribute of such links always starts with:
```
file:///homepage.html?txt=
```
followed by the text with appropriate %-escaping of special characters (including spaces). I recommend using kweb's text editor to create such links.

Hidden commands

All three kinds of command URLs can be executed automatically if you use them as the src-attribute of an iframe. Here's an example iframe that starts youtube-dl-server automatically if the page containing it is loaded:
```
<iframe src="file:///homepage.html?cmd=ytdl_server.py" width="2" height="2"
name=""></iframe>
```

Hidden commands are a powerful tool. They are extensively used by kweb's keyboard command editor. Kweb's text editor will create them for you if you use instructions without a name field (=value).

Simple command forms

Simple command forms may include only one form field, named either "cmd", "kbd" or "txt", and one submit button with an empty "name" attribute. The following example will create a menu which lets you select and execute a program:

```
<form name="execute" accept-charset="utf-8"
enctype="application/x-www-form-urlencoded" method="get"
action="file:///homepage.html">
<select name="cmd">
<option value="lxterminal">Terminal</option>
<option value="leafpad">Text Editor</option>
<option value="omxplayergui">omxplayerGUI</option>
</select>
<input value="Execute" type="submit"><br>
</form>
```

And here's another example using the “txt” command to select one or more spell checking languages:

```
<form name="language" accept-charset="utf-8"
enctype="application/x-www-form-urlencoded" method="get"
action="file:///homepage.html">
<select name="txt">
<option value="!en_GB">English</option>
<option value="!de_DE">German</option>
<option value="!fr_FR">French</option>
<option value="!fr_FR,de_DE,en_GB">All together</option>
</select>
<input value="Select Language" type="submit"><br>
</form>
```

Kweb's special command forms

Kweb supports a special kind of command form which may use multiple form fields. It must be built following 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">
```

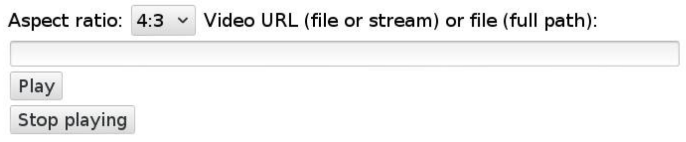
Form elements whose names start with "quoted" result in quoted arguments.
Form elements whose names start with "dquoted" result in quoted arguments using double quotes.
Both result in a single argument, even if it contains spaces.
Submit buttons should not have a name.

The following HTML code example results in an interactive interface for playing videos with omxplayer in a certain area of the screen. With this trick you can simulate running videos inside the browser.

```
<form accept-charset="utf-8" enctype="application/x-www-form-urlencoded"
method="get" action="file:///homepage.html" name="omxwindow">
<input name="cmd" value="formdata omxplayer --no-keys --no-osd --win"
type="hidden">
Aspect ratio: <select name="dquoted">
<option value="310 150 741 473">4:3</option>
<option value="310 150 8385 473">16:9</option>
</select> flush
Video URL (file or stream) or file (full path):<br>
<input size="60" name="quoted"><br>
<input value="Play" type="submit"><br>
</form>
<form accept-charset="utf-8" enctype="application/x-www-form-urlencoded"
method="get"
action="file:///homepage.html" name="stop">
<input name="cmd" value="formdata killall omxplayer.bin" type="hidden">
```

```
<input value="Stop playing" type="submit">
</form>
```

Inside the browser the form looks like this:


Aspect ratio: 4:3 Video URL (file or stream) or file (full path):
Play
Stop playing

As you can see from this example, it is possible to put multiple arguments into one form element, separated by spaces, while a (d)quoted element returns one argument, even if it contains spaces.

A Few Tricks and Tips

Running a video when an HTML page is loaded:
Add an invisible, small iframe to your page, whose 'src' element contains the URL of the video to play, e. g.
`src="file:///home/pi/myvideo.mp4"`
How this video will be played (full screen or inside a window), will depend on your settings for omxplayerGUI.

Executing a command when an HTML page is loaded:
This is done in a similar way; we use an iframe again, but this time the 'src' gets a command link:
`src="file:///homepage.html?cmd=omxplayergui.py%20--preset=nogui%20av%20/home/pi/myvideo.mp4"`
This will start the same video, but this time we send a command to omxplayerGUI and include the option to use the preset "nogui", which means, that our video is played full screen without opening a GUI (kweb's old method).
It's also possible to automatically play video in a certain screen area with omxplayer and make it look like embedded web video, while in fact it's running in a screen overlay:
`scr="file:///homepage.html?cmd=omxplayer%20--win%20100,100,900,550&20/home/pi/myvideo.mp4"`

Running your own script commands in the background:
If you need some small shell or Python scripts for your application, that should run in the background without a terminal being opened, you have to add them (full path like '/home/pi/myscript.sh') to the list of **direct_commands** in your settings.

Sending the content of a multi line HTML text area to an application script:
kwebhelper.py will replace all Carriage Returns and Line Feeds with "%0D" and "%0A" in a decoded command string. This makes it possible to send the content of a multi line HTML textarea via command line to your script as a command line argument.

Switching settings from inside an application:
Run a command like that from an iframe src:
`src="file:///homepage.html?cmd=sudo%20kwebhelper_set.py%20loadpreset%20nogui"`
The new settings will be used for the next command or media action (and all others following them). It's possible to switch between presets more than once.

Adding more responsiveness with JavaScript.:
If the commands do not open other programs and don't run in a terminal, the user will see no result. And any command execution will not result in another web page being opened, as when you click on a normal form submit button. To display any kind of result to the user inside your web page, a bit of JavaScript. could be used, which reloads the page or opens another one or modifies the src of an

iframe with a certain delay, after the user has clicked an action button or a hidden command has been performed. JavaScript. could also be used to run a whole series of commands in the background within certain intervals, simply by modifying the scr attribute of an iframe. Of course you have to enable JavaScript. in your options if you want to use it in your project.

Booting into your application:
Set up your system to boot to the command line, not to the desktop, using raspi-config. You can always start the desktop later with the “startx” command.

For the next step there might now exist better solutions on Raspbian Jessie, but I still use rc.local to start a kweb application automatically. The only thing that has to be considered is that rc.local may be executed too early by systemd during the boot process and we have to add a delay

Run
```
sudo nano /etc/rc.local
```
Add the following lines just in front of the line with “exit 0”
```
sleep 10
su -l pi -c "xinit ./kiosk"
```
and save the file with “CTRL+o”.

If you have changed the default user from “pi” to something else, you have to replace “pi” with your user name. Instead of “kiosk” you may use any matching file name (script must exist in the root of your user directory!). To boot into the predefined kweb desktop mode, replace “./kiosk” with “./ktop”.

If kweb running your application terminates, you will return to the command line.

Appendix

Kweb's History

When I started to use my first 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 ….

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 HTML5 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 (kweb's command execution links). kwebhelper.py is a kind of replacement for the plug-in support which all or most browsers on the Raspberry Pi are missing. It could be configured by the user in many ways to match his needs.

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

When the Raspberry Pi Foundation published the new version of the epiphany browser with hardware acceleration for web video and JPEG decoding and JIT compiling for JavaScript., I had to make a decision: should I use the new, accelerated webkit3 engine provided by the Foundation, which had a lot of bugs, or the older but stable webkit1 engine from the Raspbian repository? I decided to go both ways. From this moment on, there were two kweb versions: kweb, using GTK+2 and the webkit1 engine, and kweb3, using GTK+3 and the new webkit3 engine.

Meanwhile we have reached the 25th public release version of kweb. Some things are much more sophisticated than in the beginning and a lot more Python code has been added, including a GUI for omxplayer. The kweb environment makes it very easy now for the user to configure the browser and its helper programs. But the browser itself is still a very small program (about 1600 lines of source code and the binary is only about 47 KB).

For the current 1.7 version both manuals have been revised again and extended considerably.

Kweb Package Components

kweb
The browser itself, a compiled C program, using GTK+2 for the GUI and webkitgkt-1.0 as web engine.

kweb3
Another version of the browser, a compiled C program, using GTK+3 for the GUI and webkitgkt-3.0 as web engine.

kwebhelper.py
A Python script which is responsible for calling other programs to support kweb: PDF support, downloads and the built-in command execution interface are handled that way. It also contains the browser configuration manager.

omxplayergui
A Python program, compiled to a binary, responsible for all media handling. It contains GUIs for an audio player and a video player, but can also use the old kweb method of playing videos always full screen without any GUI. OmxplayerGUI is also installed and can be used as a standalone media player with a simple GUI frontend which now includes playlist management.

kwebhelper_settings.py
This small Python script contains all global variables (= configuration options) of kwebhelper.py, omxplayergui and ytdl-server. It is run each time one of these programs is started and overwrites all global variables. The experienced user may edit this file (as root), to create his own configuration. But it is much easier to use kweb's settings page for it.

kwebhelper_set.py
Another Python script that manages helper settings, handles presets and creates web pages from the settings file and user editable templates (:Commands). It manages files in the protected area of the file system and therefore has to run as root (via sudo). You may be asked for a password each time a command is executed that uses kwebhelper_set.py if you have set up your system this way.

kweb_bookmark.py
A small GUI program written in Python which is called by kweb when you want to add bookmarks to kweb's bookmark page.

kweb_edit.py
A simple text editor, written in Python, with a few special features to create and manage user-editable pages. It works as a frontend for the text to HTML compiler built into kwebhelper_set.py.

ytdl_server.py
A small web server written in Python, which provides faster access to youtube-dl to extract the web video URLs and play them with omxplayerGUI.

ginstall-ytdl, update-ytdl and makebin-ytdl
Three scripts to help installing and updating a special version of youtube-dl from github, which speeds up its use considerably and is also needed by the youtube-dl-server. The use of makebin-ytdl is optional, but when you have used it once, you must run it again after each update. It creates an all-in-one youtube-dl program file, similar to the normal distribution, but uses “pyc”-files instead of py-scripts to allow faster loading of it's (600+) modules.

dbuscontrolm.sh
This is a modified version of a shell script from the omxplayer github, supporting multiple omxplayer instances with separate dbus names. It handles all dbus commands to omxplayer.

Static files
The kweb installer creates a folder
`/usr/local/share/kweb`
and copies a number of files into this folder or creates them there. This includes text and HTML files for the :commands, manuals and much more. This is also the place where settings templates and user editable pages will be stored (by kwebhelper_set.py).
To remove all content from this folder and the folder itself, you have to use the “removeall” script when you want to remove kweb from your system completely.

List of Keyboard Commands

ALT+

	Toolbar Commands
o	Open File
b	Back
s	Stop Loading
r	Reload
h	Home
p	Play Web Video
+	Zoom +10%
-	Zoom -10%
z	Zoom 100%
g	General Zoom
t	Text Zoom only
jj	JavaScript
n	No JavaScript
e	Enable Cookies
d	Disable Cookies
uu	Uget Download Manager
w	Wget Download
x	omXplayer on
y	omxplayer off
a	Activate command execution
v	Veto command execution
	Program Commands
q	Quit Program
c	Close Current Window
f	Toggle Full Screen
k	Toggle Kiosk Mode
m	Toggle Maximized View
l	Localize Home
!	Toogle Source View
,	Bookmark Page
?	Search Similar
i	Clear and Activate Entry Line
	Open Commands
.	Open Current Page in New Browser
0 ... 9	Run User Defined Function Pages

Encodings

Values on the left side of the table can be entered into kweb directly (as "@encoding"). Values on the right side can also be used in kweb's text editor. They will be converted to the corresponding left value.

utf-8	unicode-1-1-utf-8, utf-8, utf8
ibm866	866, cp866, csibm866, ibm866
iso-8859-2	csisolatin2, iso-8859-2, iso-ir-101, iso8859-2, iso88592, iso_8859-2, iso_8859-2:1987, l2, latin2
iso-8859-3	csisolatin3, iso-8859-3, iso-ir-109, iso8859-3, iso88593, iso_8859-3, iso_8859-3:1988, l3, latin3
iso-8859-4	csisolatin4, iso-8859-4, iso-ir-110, iso8859-4, iso88594, iso_8859-4, iso_8859-4:1988, l4, latin4
iso-8859-5	csisolatincyrillic, cyrillic, iso-8859-5, iso-ir-144, iso8859-5, iso88595, iso_8859-5, iso_8859-5:1988
iso-8859-6	arabic, asmo-708, csiso88596e, csiso88596i, csisolatinarabic, ecma-114, iso-8859-6, iso-8859-6-e, iso-8859-6-i, iso-ir-127, iso8859-6, iso88596, iso_8859-6, iso_8859-6:1987
iso-8859-7	csisolatingreek, ecma-118, elot_928, greek, greek8, iso-8859-7, iso-ir-126, iso8859-7, iso88597, iso_8859-7, iso_8859-7:1987, sun_eu_greek
iso-8859-8	csiso88598e, csisolatinhebrew, hebrew, iso-8859-8, iso-8859-8-e, iso-ir-138, iso8859-8, iso88598, iso_8859-8, iso_8859-8:1988, visual
iso-8859-8-i	csiso88598i, iso-8859-8-i, logical
iso-8859-10	csisolatin6, iso-8859-10, iso-ir-157, iso8859-10, iso885910, l6, latin6
iso-8859-13	iso-8859-13, iso8859-13, iso885913
iso-8859-14	iso-8859-14, iso8859-14, iso885914
iso-8859-15	csisolatin9, iso-8859-15, iso8859-15, iso885915, iso_8859-15, l9
iso-8859-16	iso-8859-16
koi8-r	cskoi8r, koi, koi8, koi8-r, koi8_r
koi8-u	koi8-ru, koi8-u
macintosh	csmacintosh, mac, macintosh, x-mac-roman
windows-874	dos-874, iso-8859-11, iso8859-11, iso885911, tis-620, windows-874
windows-1250	cp1250, windows-1250, x-cp1250
windows-1251	cp1251, windows-1251, x-cp1251
windows-1252	ansi_x3.4-1968, ascii, cp1252, cp819, csisolatin1, ibm819, iso-8859-1, iso-ir-100, iso8859-1, iso88591, iso_8859-1, iso_8859-1:1987, l1, latin1, us-ascii, windows-1252, x-cp1252
windows-1253	cp1253, windows-1253, x-cp1253
windows-1254	cp1254, csisolatin5, iso-8859-9, iso-ir-148, iso8859-9, iso88599, iso_8859-9, iso_8859-9:1989, l5, latin5, windows-1254, x-cp1254
windows-1255	cp1255, windows-1255, x-cp1255

windows-1256	cp1256, windows-1256, x-cp1256
windows-1257	cp1257, windows-1257, x-cp1257
windows-1258	cp1258, windows-1258, x-cp1258
x-mac-cyrillic	x-mac-cyrillic, x-mac-ukrainian
gbk	chinese, csgb2312, csiso58gb231280, gb2312, gb_2312, gb_2312-80, gbk, iso-ir-58, x-gbk
gb18030	gb18030
big5	big5, big5-hkscs, cn-big5, csbig5, x-x-big5
euc-jp	cseucpkdfmtjapanese, euc-jp, x-euc-jp
iso-2022-jp	csiso2022jp, iso-2022-jp
shift_jis	csshiftjis, ms932, ms_kanji, shift-jis, shift_jis, sjis, windows-31j, x-sjis
euc-kr	cseuckr, csksc56011987, euc-kr, iso-ir-149, korean, ks_c_5601-1987, ks_c_5601-1989, ksc5601, ksc_5601, windows-949
replacement	csiso2022kr, hz-gb-2312, iso-2022-cn, iso-2022-cn-ext, iso-2022-kr
utf-16be	utf-16be
utf-16le	utf-16, utf-16le

Donations

I have often been asked if I would accept donations for my work. I'm retired (that may tell you how old I am) and don't (have to) work any more for earning money. My work for the Raspberry Pi is a free gift to the community.
But I'm also working for a small German charity organization, “Uranos e. V.”, which provides the server space for all my Raspberry Pi projects (and pays for the traffic). So it might be a good idea, to support this organization with a small donation if you like my work. They can use every support they can get.

Here's the link to the donation page:

http://uranosev.de/spenden.html

This is a German website, but there is a short English text at the bottom of the donation page.