

makeHTML user's guide

Jan Theofel *

November 6, 1999

Abstract

makeHTML is a script tool for the generation of web pages. It is still under heavy development.

This document describes how to use makeHTML as user to create your own web pages with it. If the commands or examples in this user's guide don't work with your version of makeHTML, please check my web site for new versions of the documentation.

Contents

1	Introduction	2
2	Important notes	2
3	System requirements	3
4	First steps	3
4.1	The make file	3
4.2	The dummy and content files	4
4.3	Graphic files for the menu	5
4.4	Running makeHTML	5
5	General HTML settings	5
5.1	The HTML head	5
5.1.1	The HTML version	5
5.1.2	Meta tags	6
5.1.3	The title	6
5.2	General page settings	6
6	Creating menus	6
6.1	Selecting the menu style	7
6.2	Settings for the menu	7
6.3	Creating menu items	7
6.4	Creating files without a menu item	7

*Email: jan@theofel.de, URL for makeHTML: <http://www.theofel.de/oss/>

7	html style definitions	8
7.1	Fonts	8
7.1.1	Declaring fonts	8
7.1.2	Using font definitions	8
7.2	Tables	9
7.2.1	Declaring tables	9
7.2.2	Using table definitions	9
7.3	Images	10
8	Extended html possibilities	10
8.1	Using image maps	10
9	Java script possibilities	11
9.1	Highlight graphic links	11
9.2	Use your own java script	12
10	How to . . .	12
10.1	Contacting the author	12
10.2	New program versions	12
10.3	Report bugs	12
10.4	Submit feature ideas	12
10.5	Submit your script code	13

1 Introduction

There are many graphic free and proprietary HTML editors at the moment. I don't use them, because in my opinion the HTML hacker must have the full control about the commands which are used. An other reason against using them is that their output promises a pixel exact layout which fails with most browsers.

This is why I always did write my web pages in a standard text editor (for me the vim 5.3). But this is some how difficult and hard to manage for bigger projects.

So my idea was to create a tool which helps the webmaster of complex web pages, but which allows the full control about the generated source. The result of this idea was makeHTML which I hope will be a useful tool for you.

makeHTML is a very young project. It has not yet many features, but there it already has a very powerful and many more are planned (and perhaps realized with the time, when you read this document). Please check my web page <http://www.theofel.de/oss/> for new versions.

2 Important notes

It is important for you to know, that makeHTML ist still under heavy development. The file formats will change with the next version, which will come soon. It won't be very much, but also not so little. So please keep this in mind, when

you start projects with makeHTML in this version. (But don't worry: It will be **very** easy to adapt your old files!)

3 System requirements

You need a running Perl on your system. I'm running now Perl 5.005_03 with which it works fine. Newer versions should also be supported, but for older ones, I don't know. Please check yourself and inform me please, if it works even under older versions.

If it is not located in `/usr/bin/perl`, please change the first line of the script accordingly.

For this version you need also the GD module (`gd.pm`), which is a "a perl5 interface to THOMAS BOUTELL'S gd library" written by LINCOLN D. STEIN. I'm using version 1.18 at the moment which works fine. If you have an older version please check also for yourself and inform me please, if it works with your version.

Downloading:

You'll find Perl at Larry Wall's FTP site:

`ftp://ftp.netlabs.com/pub/outgoing/perl5.0`

or in the CPAN (Comprehensive Perl Archive Network):

`http://www.perl.com/cpan/`

The GD module you can find at:

`http://stein.cshl.org/WWW/software/GD`

4 First steps

4.1 The make file

The complete web page generation is declared in a make file.¹ This file contains all important information about which files to include and the values for the different parameters.

For the projects I realized by now, I always called the file `home.make`. (This should stand for "home page makefile".)

To make this file better readable you may use blank lines and lines which start with a `#` are remarks.

If you want to manage a bigger project using makeHTML, this make file can get very big. In this case you can split in in different files using the include command in the main make file: `include_other-file.make`

This is also very helpful, if you have different projects, which should have the same look or at least some common objects (e.g. they both use the fonts *Arial* and *Helvetica* as defaults).

¹Please note that this is not a makefile for the GNU make!

4.2 The dummy and content files

Beside the make file you need some more files. The other important file is the dummy file. It defines the basic layout of your web pages:

```
<BODY_BGCOLOR="#FFFFFF">

<CENTER>
  <<TABLE_WIDTH="90%"_BORDER="0"_ALIGN="center"><TR><TD>

  <<IMG_SRC="pics/nav_left.gif"_ALT=""_BORDER="0">
  <<!--menu-->
  <<IMG_SRC="pics/nav_right.gif"_ALT=""_BORDER="0">
  <<BR><BR>
  <<!--content-->

  <</TD></TR></TABLE>
</CENTER>

<BR>
<HR_WIDTH="90%"_SIZE="1">
<CENTER>
  <<FONT_SIZE="-1"_NAME="Arial,_Helvetica">
  <<Your_footnote_which_copyright_and_other_information
  <</FONT>
</CENTER>

</BODY>
</HTML>
```

There are a few things to see in this example:

1. The HTML code contains no head. It will be created completely by the script.
2. You need a complete `<body>...</body>` tag with a closing `</html>` at the end.
3. In your body you can specify your page layout. For the both commands `<!--menu-->` and `<!--content-->` the menu and the content (see below) will be inserted by the script.

In this example we have a page which filles 90% of the width of the screen, a white background (and therefore a white frame of 5% at the left and at the right side too). The menu it a the top of the page. (This is the layout used for my Open-Source-Software pages where you also can find makeHTML.)

You also need the content files. Their HTML code is included for the `<!--content-->` command.

An very important principle of makeHTML is that the additional commands which are used have the form of `<!--...-->`, i.e. they are HTML comments.² This is true for both the dummy and the content files, so that you can simply open them in your web browsers to look at them.

4.3 Graphic files for the menu

To be able to run makeHTML correctly, you need some graphics for the menu.

Please prepare two graphics for every item you will later take in your menu. The first one is the normal graphic, the second it shown when the menu item will be selected later.

It is also possible to use decorative graphics at the start and end of the menu and between two menu items.

The graphic program of my choice is Gimp (The GNU Image Manipulation Program). If possible I suggest to use Script-Fus to generate the menu items. This is a good idea, because coming versions of makeHTML will support the execution of them for a generation of graphics. (So you needn't even to start Gimp for generation the graphics, the script will do it for you.)

4.4 Running makeHTML

When you have prepared everything, you can use the commands described in the following sections to control the generation of your web page. To start the generation please type `makehtml.pl_home.make`.

This of course only works if you can access `makehtml.pl` like this and if your file was named `home.make`.

5 General HTML settings

5.1 The HTML head

The head of the HTML code is generated completely by the script. Here are the settings you can make. There is added some additional java script to it. If you want to define your own java script code in the head please read 9.2.

5.1.1 The HTML version

To create web pages which fulfill the standards of the W3C, you must set the html version. This is done but using the command

```
set_html_<version>
```

Where `<version>` stands for the HTML version you use.

²This principle is only broken for table definitions at the moment, but this "bug" will be fixed soon.

5.1.2 Meta tags

The generation of meta tags is very simple. Simply use the syntax:

```
meta_<name>_<value>
```

to create the meta tag

```
<meta_type="name"_content="value">
```

If you want to support the DC standard, you might want to use the same value for different meta tags. With makeHTML you needn't copy'n'paste them but simply use

```
meta_name1_value1  
meta_name2_name1
```

5.1.3 The title

An other important part of the document is the title. It is defined by:

```
set_title_<text_of_the_title>
```

5.2 General page settings

There are some settings which you should specify for a correct working makefile. They have all the same syntax:

```
set_<var>_<value>
```

relgfxpath defines where the graphics can be found on the local computer where you run makeHTML.

start_count_gfx gives the number of graphics, which are in the HTML code before the menu starts. (The background graphic doesn't count, if one is specified!)

add_count_gfx gives the number of graphics between the menu and the content.

border defines the border size for your menu graphics.

dummy stands for the filename of the dummy file which is explained in 4.2.

6 Creating menus

One of the basic functions of makeHTML is the support for script generated menus which will take away many work from you. This feature will be extended to the possibility to manage complex web pages including different menus with sub menus and lots of the page internal links and files.

At the moment you can only define one main menu as follows:

6.1 Selecting the menu style

There are two different types of menus, the *horizontal* and the *vertical* ones. Please specify them by using:

```
set_menu [horizontal|vertical]
```

6.2 Settings for the menu

The following settings are declared like the settings described in 5.2:

gfxdummy specifies the dummy file between two menu items.

dummywidth / **dummyheight** gives the width / height of the menu dummy file. Use 0 for using auto detection of the size.³

gfxfilename sets a dummy filename for the menu item graphics. Use a * for the base filename as given in 6.3.

gfxhlfilename like **gfxfilename**, but is the dummy for the highlighted graphic (displayed when the menu item was selected).

gfxhl2filename like **gfxhlfilename**, but is the dummy for the graphic which is shown when the mouse moves over the menu item.

6.3 Creating menu items

You can't create a menu item without creating a file to which this menu item links. So they are both generated with one command:

```
menuitem <filename> <contentfile> <gfx1> <gfx2> <gfx3> <alt>
```

With the following parameters:

filename is the name of the file which will be created.

contentfile is the name of the file, from which the content is read.

gfx1 gives the basic graphic filename which is combined with the **gfxfilename** setting form above.

gfx2 / **gfx3** like **gfx1**. You can use a * here, if it is the same then **gfx1**.

alt specifies the alt-tag for the graphic.

6.4 Creating files without a menu item

If you want files which have the same menu like all the others but which doesn't appear in the menu itself, you can use the following command:

```
file <filename> <contentfile>
```

Where **<filename>** and **contentfile** have the same meaning like for the **menuitem** command.

³Autosize works only for GIF graphics at the moment.

7 html style definitions

7.1 Fonts

The font declarations are used to make one font style access able for many times. If you would use the HTML tag `<font_ face="...">...` you will have to change all font tags, when you want to change the default font for your page.

When using a font definition, you must only change the definition in the make file and run the script ones again. This can save a lot of time and makes sure that you don't forget to change some of the tags.

7.1.1 Declaring fonts

Declaring a font is very simple. Simply use the following syntax:

```
deffont
  name_std
  start_<font_ face="Arial, Helvetica">
  end_</font>
enddef
```

You must give the font a name, and a start and an end tag. In this case the standard font (named "std") is declared as a sans serif font. You can also use tags like ``, `<i>`, ... in the start tag and their opposites in the end tag.

7.1.2 Using font definitions

To use a font definition simply use the following to commands in your content file:

```
<!--startfont:std-->
  Your_text_in_font_std...
<!--endfont:std-->
```

Of course font definitions inside of others are allowed:

```
<!--startfont:std-->
  Your_text_in_font_std...
  <!--startfont:header-->
    Your_text_in_font_header...
  <!--endfont:header-->
  And_again_your_text_in_font_std...
<!--endfont:std-->
```

makeHTML checks if every font is ended correctly. If not, this produces a fatal error. An other fatal error occurs, when a font should be used which was not declared in the makefile.

PROPOSAL: Use one file to declare your fonts in and include this using the `include` command. This makes it easy to use the same fonts in different projects.

7.2 Tables

The advantage of table definitions is the same than it is for fonts. Here's how to declare and use them.

7.2.1 Declaring tables

The declaration of a table is nearly the same than for fonts:

```
deftable
  name std
  border 0
  cellspacing 10
  cellpadding 0
  tdstart <font face="Arial, Helvetica">
  tdend </font>
enddef
```

(Note that you can use the same names for tables which are already declared for a font.)

The elements which are declared are the following:

border is the size of the table border in pixels. Use 0 for an invisible table for the page layout.

cellspacing is the width of the lines between the table cells. (Which is invisible when you set **border** 0.)

cellpadding is the distance of the content of the table cells to the cell border.

tdstart will be inserted at the start of every table cell.

tdend will be inserted at the end of every table cell.

As the font replacements are done after the one of the tables, you can use definitions like this one:

```
deftable
  name std
  border 0
  cellspacing 10
  cellpadding 0
  tdstart <!--startfont:std-->
  tdend <!--endfont:std-->
enddef
```

7.2.2 Using table definitions

To use a table definition, you can use the same syntax as you already know for fonts:

```

<!--starttable:std-->
  Your table cells with "std" style...
  <!--starttable:inside-->
    Your table cells with "inside" style...
  <!--endtable:inside-->
  Again table cells with "std" style...
<!--endfont:std-->

```

7.3 Images

Let's assume, that you use one single graphic a few time, for exapmle a symbol for downloading. Here is a simple and effective way to reduce your work:

```
img_alias<IMG_SRC="file.gif" WIDTH="..." HEIGHT="..." ALT="...">
```

Put this command in one of your makefiles. Then you can use the following command to include the image in your HTML code:

```
<!--img:alias-->
```

TIP: This can also used as an short command for using the same HTML code a few times. (That fine for this version of makeHTML, but in the next one, there will be a special command for this, too.)

8 Extended html possibilities

There are some more HTML possibilities supported by makeHTML.

8.1 Using image maps

An image map is an image with "click sensitive" areas, i.e. if you click on them, this links to another file.

Again, the declaration is very simple:

```

imagemap
  name imagemapname
  <kind1><link1><coords1><alt1>
  <kind2><link2><coords2><alt2>
  ...
endimagemap

```

The name is again the same as for fonts or tables. Important are the following lines (you may use as many as you want). The define the areas on which you can click in the image.

For **kind** you may use **rect**, **circle** or **polygon**. The **link** sets the file to which the link should shown to and **alt** is put in the **alt** parameter (so it is displayed when you move your mouse over this area of the image).

Only **coords** is a little bit more difficult. It depends on the **kind**:

rect coords are the corners of a rectangular as x_1, y_1, x_2, y_2 . All the values are separated with a comma. Please don't use blanks here!

circle Here this are three values: $x, y, radius$, again separated by a comma without blanks.

polygon Here you give the points of a polygon as $x_1, y_1, x_2, y_2, x_3, y_3, \dots$. The last pair of x/y must be the same like the first one. Again please separated by commas without blanks.

9 Java script possibilities

9.1 Highlight graphic links

In the menu the items are automatically highlighted, when you move your mouse over them. This is also possible for other graphics which are links.

First you must include the needed graphics in your java script section in the head of the file:

```
defjavascriptgfx
  files_index.html
  filename_/pics/*.gif
  name1_pic1
  name2_pic2
  name1_hl_pic1_hl
  name2_hl_pic2_hl
enddef
```

The setting `files` tells the script in which files to include the pictures in the java script section. Please give the full filename here, don't use `?` and `*` which is not yet supported. If you need them in many files, add them all departed by a blank.

The `filename` gives the dummy filename of the graphics, where `*` is later replaced by the basic filename.

These two settings are followed by a list of graphics which are to include. Please list first the graphics which are not highlighted in the form

```
<name>_<basefilename>
```

and then same graphics, but highlighted.

To highlight a graphic link now on "mouse over", simply use this syntax:

```
<A_HREF="aktuelles.html">!--highlight:name,name\_hl-->>
<IMG_SRC="..."_WIDTH="..."_HEIGHT="..."_BORDER="..."></A>
```

where `name` is the non highlighted filename declared above and `name_hl` the same graphic, but highlighted.

9.2 Use your own java script

If you want to use your won java script code in the files, simply use:

```
javascript
//Your java script code...
endjavascript
```

This will be included in all files at the moment.

10 How to ...

10.1 Contacting the author

If you want to contact me, the author of makeHTML, you have the following possibilities:

Email:	jan@theofel.de
Homepage (German):	http://www.theofel.de/
Homepage (English):	http://www.theofel.de/english/
Snail mail:	Jan Theofel Uhlandstr. 20 71106 Magstadt Germany

10.2 New program versions

You can find new version of makeHTML under the following URL:

<http://www.theofel.de/oss/>

10.3 Report bugs

If you find a bug in makeHTML, please first check the file **known-bugs**. If it is not in this list, please check the latest version of makeHTML and the bug list for this version. If there is also no solution / note about the bug you found, please mail me at jan@theofel.de. Please describe exactly what you did, what went wrong and if possible the error which is displayed. Please send only the important parts of your source inline (included in the mail) and not your hole packed project as attachment!

10.4 Submit feature ideas

If you have an idea how to make makeHTML even better, please read first the file **to-do-list** and if your missed feature is not found in there, check the latest online version. If you still don't find a hint for your feature, please mail me at jan@theofel.de.

10.5 Submit your script code

If you have implemented some features in makeHTML and want to send me this code, please contact me before at jan@theofel.de. This is my first real open source project and I don't have many experiences in this area. ;-)