Introduction

This document describes the Perl script html2ps – an HTML to PostScript converter. The program is developed on a Unix system (Sun Solaris 2.5.1), but earlier versions have been reported working on other platforms, such as Windows NT, MS-DOS and MacIntosh.

First, do not get intimidated by the size of this document. At least on a Unix system with Perl installed, you should be able to convert a local text-only HTML file directly without doing any preparation at all. Just execute the script with the HTML file as parameter and direct the output to a file or a printer, for example:

```
html2ps file.html > file.ps
```

or perhaps

```
perl html2ps file.html > file.ps
```

There are several possibilities to control the appearance of the output. Starting with version 1.0 of html2ps, most of this is done via configuration files, and not with command line options only. A sample configuration file 'sample' is included with the html2ps distribution (used by the installation script to convert this document to postscript). You can take a look at this file to see an example of a configuration file.

The command line options, and how to set their defaults using a configuration file, are described in the `option` block below. Due to the new concept of configuration files, some of the command line options in earlier versions have been removed. Hopefully, this will not cause too many problems for old users.

Jean-Philippe Argaud has written a GUI (xhtml2ps) for html2ps. You can find this in the directory 'contrib'. Both html2ps and xhtml2ps comes with the GNU General Public License, see the file 'COPYING' for details.

There is a mailing list for html2ps. If you want to be included on this list, send a mail with the word 'subscribe' on the subject line to html2ps-request@tdb.uu.se.

Installation

For certain tasks, such as rendering of inline images and retrievement of remote documents, html2ps depends on other program packages. First of all, you should make sure that you have all the necessary software is installed on your system. See below where these packages can be found.

Since html2ps is written in Perl, you must of course have Perl installed, version 5 or later is required.

If you want to make use of the support for inline images, you must have one of the packages ImageMagick, pbmplus or netpbm installed. If you do not have ImageMagick (or if it is not installed with jpeg support), you can use djpeg from the jpegsrc distribution to handle jpeg images. The recommended image package is ImageMagick together with its Perl module interface PerlMagick.

To be able to convert remote documents directly without downloading them first, you must have have some program installed for this purpose. The best is to either use the Perl module library libwww-perl or www.pl and network.pl, but you can for example also use lynx or url_get for this.

Ghostscript is also required to get full functionality (needed to generate DSC PostScript and cross references).

If you are on a Unix system, and have all necessary software installed, you can now execute the installation script 'install' to build a global configuration file, and install all files. On other platforms you will have to create a global configuration file manually, and insert its name into the html2ps script. This is described in next section.

Configuration files

Configuration files are used for layout control, resource information etc. Normally, there should always exist a global configuration file. In this file one typically specify things like: what image conversion packages are available on the system, the default paper size, the default text fonts and sizes, etc.

For Unix systems, the installation script 'install' can be used to automatically build a global configuration file with all necessary definitions, and install all files. The files replaced by the installation are saved. If you for some reason are not satisfied with the new version: execute the script 'backout' to reinstall your earlier version.

On non-Unix systems, you will have to manually create a global configuration file, and insert the name of this file into the html2ps script (close to the beginning, the line starting with "`$globrc=`"). The configuration file should contain a `package` block, and perhaps `paper` and `hyphenation` blocks, described below.

Each user can then have a personal configuration file (by default $HOME/.html2psrc) that complements/overrides the definitions made in the global file. It is also possible to specify alternative files on the command line, using the -f option.

File format

A configuration file can include other configuration files. This is done with:

```
@import "filename";
```

The rest of the configuration file consists of zero or more blocks. A block is given by a block name, followed by the block definition, as in:

```
BODY {
  font-size: 12pt;
  font-family: Helvetica;
  text-align: justify
}
```

The block definition, enclosed by curly braces: { }, consists of key-value pairs and/or other blocks. A key-value pair consists of the key name followed by a colon, followed by the value. Blocks and key-value

pairs are separated by semicolons. The semicolon may be omitted after a block.

Several blocks can share the same definition. The block names are then separated be commas, as in:

```
H2, H4, H6 { font-style: italic }
```

A comment in a configuration file starts with the characters "`/*`" and ends with "`*/`":

```
@html2ps {
  seq-number: 1;  /* Automatic numbering of headings */
}
```

Notations

Here are some definitions of terms used below:

Flag:
: A value of either 0 (absence, inactive etc) or 1 (presence, active etc).

Absolute size:
: A real number optionally followed by one of the following two-letter unit identifiers: cm (centimeters), mm (millimeters), in (inches), pt (points, 1pt = 1/72 inch), pc (picas, 1pc = 12pt). The default unit is centimeters.

Relative size:
: A size relative to current fontsize. The default and currently only recognized unit is em. One em equals the size of the current font. The value should be given as a real number, optionally followed by 'em', as in '0.25em'.

Whitespace:
: Any one of the characters: space, tab, newline, or carriage return.

CSS1 blocks

All blocks, except one: the `@html2ps` block, coincides with a subset of the CSS1 – Cascading Style Sheets level 1 – specification. (If this link is no longer valid, try looking for style sheet documentation starting at the W3C home page.)

The following default settings for html2ps illustrate just about everything that currently can be used from the CSS1 specification:

```
BODY {
  font-family: Times;
  font-size: 11pt;
  text-align: left;
  background: white;
}

H1, H2, H3, H4, H5, H6 {
  font-weight: bold;
  margin-top: 0.8em;
  margin-bottom: 0.5em;
}
H1 { font-size: 19pt }
```

```
H2 { font-size: 17pt }
H3 { font-size: 15pt }
H4 { font-size: 13pt }
H5 { font-size: 12pt }
H6 { font-size: 11pt }

P, OL, UL, DL, BLOCKQUOTE, PRE {
  margin-top: 1em;
  margin-bottom: 1em;
}

P {
  line-height: 1.2em;
  text-indent: 0;
}

OL, UL, DD { margin-left: 2em }

TT, KBD, PRE { font-family: Courier }

PRE { font-size: 9pt }

BLOCKQUOTE {
  margin-left: 1em;
  margin-right: 1em;
}

ADDRESS {
  margin-top: 0.5em;
  margin-bottom: 0.5em;
}

TABLE {
  margin-top: 1.3em;
  margin-bottom: 1em;
}

DEL { text-decoration: line-through }

A:link, HR { color: black }
```

The program specific block `@html2ps`

This block is used to specify parameters that are specific to html2ps, and not covered by CSS1. The `@html2ps` block has several sub-blocks and key-value pairs, these are described in this section.

The `package` block

This block is used to specify which program packages are installed on the system. Typically, this is done in the global configuration file.

`PerlMagick`
: A flag specifying whether the Perl module PerlMagick is installed or not. The default is 0.

`ImageMagick`
: A flag specifying whether the ImageMagick package is installed or not. The default is 0.

`pbmplus`
: A flag specifying whether the pbmplus package is installed or not. The default is 0.

`netpbm`
: A flag specifying whether the netpbm package is installed or not. The default is 0.

`djpeg`
: A flag specifying whether djpeg is installed or not. The default is 0.

`Ghostscript`
: A flag specifying whether Ghostscript is installed or not. The default is 0.

`TeX`
: A flag specifying whether the TeX package is installed or not. The default is 0.

`dvips`
: A flag specifying whether dvips is installed or not. The default is 0.

`libwww-perl`
: A flag specifying whether the Perl module library libwww-perl is installed or not. The default is 0.

`jfriedl`
: A flag specifying whether the Perl scripts www.pl and network.pl (by Jeffrey Friedl) are installed or not. The default is 0.

`geturl`
: When neither of the Perl packages for retrieving remote documents are available, it is possible to use some other program like lynx or url_get. This value should be set to a command that retrieves a document *with a complete MIME header*, such as "`lynx -source -mime_header`" or "`url_get -h`".

`check`
: The name of a program used for syntax checking HTML documents. No default, a good choice is weblint.

`path`
: A colon separated list of directories where the executables from the program packages are. It is only necessary to include directories that are not in the PATH for a typical user.

The `paper` block

The paper size is defined in this block. The size can either be given as one of the recognized paper types or by giving explicit values for the paper height and width.

`type`
: Paper type, possible choices are: A0, A1, A2, A3, A4,...,A10, B0, B1,...,B10, letter, legal, arche, archd, archc, archb, archa, flsa, flse, halfletter, 11x17, and ledger (this set of paper types is taken from Aladdin Ghostscript). The default is A4.

`height`
: An absolute size specifying the paper height.

`width`
: An absolute size specifying the paper width.

The `option` block

This block is used to set default values for the command line options. The key in the key-value pair is the option name, in either its long or short form.

`twoup` (on the command line: `-2` or `--twoup`)
: Two column (2-up) output. The default is one column per page.

`base` (on the command line: `-b` *`URL`* or `--base` *`URL`*)
: Use URL as a base to expand relative references for in-line images. This is useful if you have downloaded a document to a local file. The URL should then be the URL of the original document.

`check` (on the command line: `-c` or `--check`)
: Check the syntax of the HTML file (using an external syntax checker). The default is to not make a syntax check.

`toc` (on the command line: `-C` *`string`* or `--toc` *`string`*)
: Generate a table of contents (ToC). The value should be a string consisting of one of the letters 'f', 'h', or 't', optionally combined with the letter 'b':
  - b The ToC will be printed first. This requires that Ghostscript is installed.
  - f The ToC will be generated from the links in the converted document.
  - h The ToC will be generated from headings and titles in the converted documents. Note that if the document author for some strange reason has chosen to use some other means to represent the headings than the HTML elements H1,...,H6, you are out of luck!
  - t The ToC will be generated from links having the attribute `rev=TOC` in the converted document.

`debug` (on the command line: `-d` or `--debug`)
: Generate debugging information. You should always use this option when reporting problems with html2ps.

`DSC` (on the command line: `-D` or `--DSC`)
: Generate DSC compliant PostScript. This requires Ghostscript and can take quite some time to do. Note that a PostScript file generated with this option cannot be used as input to html2ps for reformatting later.

`encoding` (on the command line: `-e` *`encoding`* or `--encoding` *`encoding`*)
: The document encoding. Currently recognized values are ISO-8859-1, EUC-JP, SHIFT-JIS, and ISO-2022-JP (other EUC-xx encodings may also work). The default is ISO-8859-1.

`rcfile` (on the command line: `-f` *`file[:file[:...]]`* or `--rcfile` *`file[:file[:...]]`*)
: A colon separated list of configuration file names to use instead of the default personal configuration file $HOME/.html2psrc. Definitions made in one file override definitions in previous files (the last file in the list has highest precedence). An empty file name (as in ':file', 'file1::file3', or 'file:') will expand to the default personal file. The environment variable HTML2PSPATH is used to specify the directories where to search for these files. (Note: this is *only* supposed to be used on the command line, not in a configuration file.)

`frame` (on the command line: `-F` or `--frame`)
: Draw a frame around the text on each page. The default is to not draw a frame.

`grayscale` (on the command line: `-g` or `--grayscale`)
: Convert colour images to grayscale images. Note that the PostScript file will be smaller when the images are converted to grayscale. The default is to generate colour images.

`help` (on the command line: `-h` or `--help`)
Show usage information.

`hyphenate` (on the command line: `-H` or `--hyphenate`)
Hyphenate the text. This requires TeX hyphenation pattern files.

`scaleimage` (on the command line: `-i` *num* or `--scaleimage` *num*)
Scale in-line images with a factor num. The default is 1.

`language` (on the command line: `-l` *lang* or `--language` *lang*)
Specifies the language of the document (overrides an eventual LANG attribute of the BODY element). The language should be given according to RFC1766 and ISO 639.

`landscape` (on the command line: `-L` or `--landscape`)
Generate code for printing in landscape mode. The default is portrait mode.

`scalemath` (on the command line: `-m` *num* or `--scalemath` *num*)
Scale mathematical formulas with a factor num. The default is 1.

`number` (on the command line: `-n` or `--number`)
Insert page numbers. The default is to not number the pages.

`startno` (on the command line: `-N` *num* or `--startno` *num*)
Specifies the starting page number, the default is 1.

`output` (on the command line: `-o` *file* or `--output` *file*)
Write the PostScript code to file. The default is to write to standard output.

`original` (on the command line: `-O` or `--original`)
Use PostScript original images if they exist. For example, if a document contains an image figure.gif, and an encapsulated PostScript file named figure.ps exists in the same directory, that file will be use instead. This only work for documents read as local files. Note: if the PostScript file is large or contains bitmap images, this must be combined with the -D option. In HTML 4.0 this can be achieved in a much better way with:

```
<OBJECT data="figure.ps" type="application/postscript">
 <OBJECT data="figure.gif" type="image/gif">
  <PRE>[Maybe some ASCII art for text browsers]</PRE>
 </OBJECT>
</OBJECT>
```

`rootdir` (on the command line: `-r` *path* or `--rootdir` *path*)
When a document is read from a local file, this value specifies a base directory for resolving relative links starting with "/". Typically, this should be the directory where your web server's home page resides.

`xref` (on the command line: `-R` or `--xref`)
Insert cross references at every link to within the set of converted documents.

`scaledoc` (on the command line: `-s` *num* or `--scaledoc` *num*)
Scale the entire document with a factor num. The default is 1.

`style` (on the command line: `-S` *string* or `--style` *string*)
This option complements/overrides definitions made in the configuration files. The string must follow the configuration file syntax. (Note: this is *only* supposed to be used on the command line, not in a configuration file.)

`text` (on the command line: `-T` or `--text`)
Text mode, ignore images. The default is to include the images.

`underline` (on the command line: `-u` or `--underline`)
: Underline text that constitutes a hypertext link. The default is to not underline.

`colour` (on the command line: `-U` or `--colour`)
: Produce colour output for text and background, when specified. The default is black text on white background (mnemonic: coloUr ;-).

`version` (on the command line: `-v` or `--version`)
: Print information about the current version of html2ps.

`web` (on the command line: `-W` *`string`* or `--web` *`string`*)
: Process a web of documents by recursively retrieve and convert documents that are referenced with hyperlinks. When dealing with remote documents it will of course be necessary to impose restrictions, to avoid downloading the entire web... The value should be a string consisting of one of the letters 'a', 'b', 'l', 'r', or 's', optionally combined with a combination of the letters 'p', 'L', and a positive integer:

  a Follow all links.
  b Follow only links to within the same directory, or below, as the start document.
  l Follow only links specified with "`<LINK rel=NEXT>`" in the document.
  p Prompt for each remote document. This mode will automatically be entered after the first 50 documents.
  r Follow only relative links.
  s Follow only links to within the same server as the start document.
  L With this option, the order in which the documents are processed will be: first all top level documents, then the documents linked to from these etc. For example, if the document A has links to B and C, and B has a link to D, the order will be A→B→C→D. By default, each document will be followed by the first document it links to etc; so the default order for the example is A→B→D→C.
  # A positive integer giving the number of recursive levels. The default is 4 (when the option is present).

`duplex` (on the command line: `-x` *`num`* or `--duplex` *`num`*)
: Generate postscript code for single or double sided printing. No default, valid values are:

  0 Single sided.
  1 Double sided.
  2 Double sided, opposite page reversed (tumble mode).

The `margin` block

This block is used to specify page margins.

`left`
: An absolute size of the left margin, default is 2.5cm.

`right`
: An absolute size of the right margin, default is 2.5cm.

`top`
: An absolute size of the top margin, default is 3cm.

`bottom`
: An absolute size of the bottom margin, default is 3cm.

`middle`
: An absolute size for the distance between the columns when printing two columns per page, default is 2cm.

The `xref` block

At every hyperlink (to within the set of converted documents) it is possible to have a cross reference inserted. The `xref` block is used to control this function.

`text`
: This defines the cross reference text to be inserted; the symbol $N will expand to the page number, default is "[p $N]".

`passes`
: The number of passes used to insert the cross references. Normally, only one pass is run. But since the insertion of the page numbers may effect the page breaks, it might for large documents with many links be necessary with more than one pass to get the cross references right. The default is 1.

The `quote` block

Language specific quotation marks are defined in this block. These quotation marks are used with the HTML 4.0 element Q for short quotations. Quotation marks for a few languages are predefined, see below. It is possible to define different quotation marks for quotes within quotes.

A quotation mark is defined as a string, using the same encoding as the converted document (normally ISO-8859-1), and/or with character entities. Note that quotation mark characters for several languages are not included in ISO-8859-1, and their corresponding character entities were not been defined prior to HTML 4.0.

Quotation marks for a language can be defined explicitly in a sub-block of the `quote` block. One can also identify the set of quotation marks with another previously defined language, using a key-value pair. The sub-block/key name should equal the language code as defined in ISO 639. The language sub-block can have the following key-values:

`open`
: The quote opening character(s).

`close`
: The quote closing character(s). If undefined, it will equal `open`.

`open2`
: The quote opening character(s) for quotes within quotes. If undefined, it will equal `open`.

`close2`
: The quote closing character(s) for quotes within quotes. If undefined, it will equal `close`.

Example: English and Spanish use the same set of quotation marks – at least according to my book on typography. These (already known to html2ps) are defined with:

```
quote {
  en {
    open: "&ldquo;";
    close: "&rdquo;";
    open2: "`";
    close2: "'";
  }
  es: en;
}
```

The `toc` block

When a table of contents (ToC) is generated from document headings and titles, the appearance is controlled by this block.

`heading`
: A string with HTML code specifying a heading used on the first ToC page. The symbols $T and $A can be used for the document title and author respectively.

`level`
: The maximum heading level used for building the ToC. The default is 6, which means that all headings will generate ToC entries.

`indent`
: The ToC entries are indented proportional to the corresponding heading level. This value specifies the size of the indentation. The default is 1em.

`extrapage`
: A flag specifying whether an extra (empty) page should be printed between the ToC and the rest of the document, when necessary, to ensure that both will start on an odd page. This is typically desirable for double sided printing. The default is 1.

The `font` block

Currently, html2ps recognizes the fonts: Times, New-Century-Schoolbook, Helvetica, Helvetica-Narrow, Palatino, Avantgarde, Bookman, and Courier. To add a new font (family), choose a name (consisting of letters, digits, hyphens, and underscores) for the font. Then define a sub-block to the `font` block, with the same name as the chosen font name. This block can contain two key-value pairs:

`names`
: A string containing four PostScript font names, separated by whitespace, corresponding to the font styles normal, italic, bold, and bold-italic. If less than four names are given, the first is used for the missing names. Note that PostScript font names are case sensitive.

`files`
: A string of four file names, separated by whitespace, for files containing font definitions for the four font styles as specified above.

Example: A font 'myfont' has its four font styles defined in local files. To use this font in all tables in the converted documents, one can use something like:

```
TABLE { font-family: myfont }

@html2ps {
  font {
    myfont {
      names: "MyFont-Roman MyFont-Italic MyFont-Bold MyFont-BoldItalic";
      files: "/x/y/myfr.ps /x/y/myfi.ps /x/y/myfb.ps /x/y/myfbi.ps";
    }
  }
}
```

The `hyphenation` block

Hyphenation pattern files for different languages are specified in sub-blocks within this block. The blocks names should equal the language code as defined in ISO 639. These language blocks can contain the following two key-values:

`file`
: A hyphenation pattern file in TeX format for this language.

`extfile`
: A file containing a list of hyphenation exceptions for this language. The exception file should contain words, separated by whitespaces, with hyphens inserted where hyphenation is allowed, as in: "`in-fra-struc-ture white-space`".

For example, for English (with language code 'en') one can have a block like:

```
en {
  file: "/opt/tex/lib/macros/hyphen.tex";
  extfile: "/opt/tdb/lib/html2ps/enhyphext";
}
```

The `hyphenation` block itself can furthermore have these key-values:

`min`
: A positive integer defining the minimum number of letters a word must contain to make it a candidate for hyphenation. The Default is 8.

`start`
: A positive integer defining the minimum number of letters that must precede the hyphen when a word is hyphenated. The default is 4.

`end`
: A positive integer defining the minimum number of letters that must follow the hyphen when a word is hyphenated. The default is 3.

The `header` block

This block is used to specify page headers. It is possible to define left, center, and right headers. Different headers for odd and even pages can be specified. Some symbols can be used that will expand to document title, author, date, etc. See below.

`left`
: A left aligned header. If the `alternate` flag in this block is set to 1, this will be the right header on even pages.

`center`
: A centered header.

`right`
: A right aligned header. If the `alternate` flag in this block is set to 1, this will be the left header on even pages.

`odd-left`
: A left aligned header on odd pages.

`odd-center`
: A centered header on odd pages.

`odd-right`
: A right aligned header on odd pages.

`even-left`
: A left aligned header on even pages.

`even-center`
: A centered header on even pages.

`even-right`
: A right aligned header on even pages.

`font-family`
: The font used for the header, default is Helvetica.

`font-size`
: The font size for the header, default is 8pt.

`font-style`
: The default is "normal".

`font-weight`
: The default is "normal".

`color`
: The header color, default is black.

`alternate`
: A flag indicating whether the headers given by the `left` and `right` keys should change place on even pages. Typically used for double sided printing. The default is 1.

The `footer` block

This block is used to specify page footers. It is possible to define left, center, and right footers. Different footers for odd and even pages can be specified. Some symbols can be used that will expand to document title, author, date, etc. See below.

`left`
: A left aligned footer. If the `alternate` flag in this block is set to 1, this will be the right footer on even pages.

`center`
: A centered footer.

`right`
: A right aligned footer. If the `alternate` flag in this block is set to 1, this will be the left footer on even pages.

`odd-left`
: A left aligned footer on odd pages.

`odd-center`
: A centered footer on odd pages.

`odd-right`
: A right aligned footer on odd pages.

`even-left`
: A left aligned footer on even pages.

`even-center`
: A centered footer on even pages.

`even-right`
: A right aligned footer on even pages.

`font-family`
: The font used for the footer, default is Helvetica.

`font-size`
: The font size for the footer, default is 8pt.

`font-style`
: The default is "normal".

`font-weight`
: The default is "normal".

`color`
: The footer color, default is black.

`alternate`
: A flag indicating whether the footers given by the `left` and `right` keys should change place on even pages. Typically used for double sided printing. The default is 1.

The `frame` block

The appearance of the optional frame (drawn on each page) is controlled by this block.

`width`
: The width of the frame, default is 0.6pt.

`margin`
: The size of the frame margin, default is 0.5cm.

`color`
: The colour of the frame, default is black.

The `justify` block

This block specifies the maximum amount of extra space inserted between words and letters when text justification is in effect.

`word`
: Maximum amount of extra space inserted between words. The default is 15pt.

`letter`
: Maximum amount of extra space inserted between letters within words. The default is 0pt.

The `draft` block

It is possible to have some text written in a large font diagonally across each page. Typically this is a word, written in a very light colour, indicating that the document is a draft.

`text`
: The text to be printed, default is "DRAFT".

`print`
: A flag specifying whether the draft text should be printed or not. If unspecified, the draft text is printed when the document head contains `<META name="Status" content="Draft">`.

`dir`
: Specifies print direction, 0=downwards, 1=upwards.

`font-family`
: The default is Helvetica.

`font-style`
: The default is "normal".

`font-weight`
: The default is "bold".

`color`
: The default is "F0F0F0".

The `colour` block

The 16 colour names from the HTML 3.2 specification are recognized by html2ps. Use this block to extend this list of colours. This is done with key-value pairs, where the key is the colour name, and the value is the colour given as a hexadecimal RGB value, for example: "`brown: A52A2A;`".

Key-value pairs in the `@html2ps` block

`html2psrc`
: The name of the default personal configuration file. The default is $HOME/.html2psrc.

`imgalt`
: Specifies which text should be written as a replacement for in-line images when the IMG element has no ALT attribute. The default is `"[IMAGE]"`.

`datefmt`
: The symbol $D can be used in page headers and footers to insert the current date/time; the value of the `datefmt` key specifies the format used. The syntax is the same as in the strftime(3) routine. The default is `"%e %b %Y %R"`, which gives a date string like "9 Aug 1997 18:24".

`locale`
: The locale (language code) used for formating language dependent parts of the date/time in `datefmt`. If unspecified, the value is taken from environment variables, see setlocale(3). No default.

`doc-sep`
: A string of HTML code that will be inserted between the documents when more than one are converted. The default is `"<!--NewPage-->"`, which will cause a page break. You may use

(almost) any HTML code, for example "<HR><HR>" or "<IMG src=...>".

`ball-radius`
: The radius, given as a relative size, of the balls used in unordered lists. The default is 0.25em.

`numbstyle`
: Page numbering style, 0=arabic, 1=roman. The default is 0.

`showurl`
: When this flag is set to 1, the URL for external links are shown within parentheses after the link. The default is 0.

`seq-number`
: When this flag is set, the headings in the document will be sequentially numbered: H1 headings will be numbered 1, 2,..., H2 headings 1.1, 1.2, etc. The default is 0.

`forms`
: This flag is used to specify whether FORM elements in the document should be processed or ignored. Some forms may be suitable for printing out and be filled out (with a pen), others are not. The default is 1.

`textarea-data`
: When a TEXTAREA element contains prefilled data, the text will be used as labels if this flag is set, otherwise ignored. The default is 1.

`page-break`
: Set this flag to 0 to suppress the normal behavior of generating page breaks form the comment `<!--NewPage-->` etc, as specified below. The default is 1.

`expand-acronyms`
: A flag specifying whether acronyms, given by the ACRONYM element, should be expanded or not. The default is 0.

`collapse-br`
: The default is 1.

`spoof`
: Some web servers return different documents depending on which user agent is used to retrieve the document. You can fool the web server that a certain browser is used, by setting this value to the identification used by the browser, such as "Mozilla/4.0". This only works if you are using one of the Perl packages to retrieve remote documents.

Symbols

The following symbols can be used in page headers/footers:

$T Current document title.
$A Author of current document, as specified with `<META name="Author" content="...">` in the document head.
$U The URL or file name of current document.
$N Page number.
$H Current document heading (level 1-3).
$D Current date/time. The format is given by the `datefmt` key.

To avoid symbol expansion, precede the dollar sign with a backslash, as in "\$T".

Some hints how to use configuration files

I imagine that a typical use of configuration files can be something along the following lines.

System specific definitions (e.g. specification of available program packages) and global defaults (paper type etc) are defined in the global configuration file.

If there is more than one user of the program on the system, each user can also have a personal configuration file with his/hers own personal preferences. (On a single user system one can use the global configuration file for this purpose as well.)

One may also develop a collection of configuration files for typical situations. These files are placed in a directory that is searched by html2ps (the search path is defined with the environment variable HTML2PSPATH). For example, to print a document as slides – in landscape mode, with large text in Helvetica, and a thick frame – one can create a configuration file, called 'slides' say, containing:

```
@html2ps {
  option {
    landscape: 1;
    frame: 1;
  }
  frame { width: 3pt }
}
BODY {
  font-family: Helvetica;
  font-size: 20pt;
}
H1 { font-size: 35pt }
H2 { font-size: 32pt }
H3 { font-size: 29pt }
H4 { font-size: 26pt }
H5 { font-size: 23pt }
H6 { font-size: 20pt }
PRE { font-size: 18pt }
```

Then use the command:

```
html2ps -f slides ...
```

to convert the document. Note that with this command the file 'slides' is used *instead* of the personal configuration file. If you want both to be used, giving precedence to definitions made in the file 'slides', use the command:

```
html2ps -f :slides ...
```

(The page breaks between the slides can for example be generated by adding '`<HR class=PAGE-BREAK>`' to the HTML document.)

For features that are frequently turned on and off, and that cannot be controlled by command line options, it may be a good idea to create small configuration files as "building blocks". For example a file 'A4' for printing on A4 paper (if you have some other default paper type):

```
@html2ps { paper { type: A4 } }
```

and a file 'hnum' for automatic numbering of headings:

```
@html2ps { seq-number: 1 }
```

Combining this with the previous example: to convert a document for printing on A4 sized slides with all headings numbered, use the command:

```
html2ps -f :slides:A4:hnum ...
```

HTML 4.0 support

Many of the new features in the HTML 4.0 draft are, at least partly, supported by html2ps. Here follow a brief description of some of these.

OBJECT

The OBJECT element is recognized; the supported object types are images, PostScript documents (eps), and HTML documents. For other types, the content of the OBJECT element is rendered, as specified in HTML 4.0.

TABLE

Most of the extended table element is supported. This includes the elements THEAD, TFOOT (used to get a running table head and foot when a table is broken across pages), TBODY, COLGROUP, and COL. It is also possible to get horizontal alignment on a specific character (align=char).

INS and DEL

The new elements INS and DEL, for marking text as inserted and deleted respectively, are recognized. Inserted and deleted text is indicated with a vertical bar in the left margin. By default, deleted text is rendered as struck-through; it is possible to have deleted text not shown at all, by specifying

```
DEL { display: none }
```

in a configuration file.

ACRONYM

The expansion of acronyms, given with the ACRONYM element, will be shown within parenthesis following the acronym if one specifies:

```
@html2ps { expand-acronyms: 1 }
```

in a configuration file. By default, acronyms are not expanded.

The id attribute

The id attribute is recognized as an anchor for a link (only relevant when cross references are inserted, and when the postscript code is converted to PDF).

Named character entities

All character entities in HTML 4.0, except ‌, ‍, ‎, and ‏, are supported.

Internationalization

There is not yet much support for internationalization, as specified in HTML 4.0. A few character encodings, other than the old default ISO-8859-1, are supported. The element Q for short quotations is implemented (quotation marks are chosen based on current language). It is also possible to make language dependent hyphenation, using hyphenation pattern files from the TeX distribution. The `lang` attribute (recognized for a few elements) is used to determine the language of the document, or for a particular section.

Document encodings

It is possible to convert documents using some other character encodings than the old default ISO-8859-1. Currently the Japanese encodings EUC-JP, SHIFT-JIS, and ISO-2022-JP (7-bit) are recognized. EUC encoded documents in other languages, such as Chinese (EUC-CN), and Korean (EUC-KR) will probably work as well. I have no such font available, so I have not been able to test this myself.

You must of course have access to PostScript fonts for these languages and encodings. For Japanese you can use the free Wadalab fonts and associated tools. These fonts can be used together with Ghostscript to print on a printer that does not have such fonts.

Of the three Japanese encodings, EUC-JP works best with html2ps. The 7-bit encoding ISO-2022-JP, found in several documents on the web, is a bad choice for HTML documents. This is because the encoding uses characters that have a special function in HTML, such as '<'. Often this results in incorrect HTML code, and html2ps as well as browsers will of course have problems rendering these documents.

There is not yet any automatic detection of the encoding, this must be explicitly specified with the option -e, or in a configuration file. If you for example would like to convert a document with EUC-JP encoding, you can use a configuration file like:

```
BODY { font-family: eucjp }

@html2ps {
  option {
    encoding: "EUC-JP";
  }
  font {
    eucjp {
```

```
        names: "Ryumin-Light-EUC-H";
      }
    }
  }
```

The name `Ryumin-Light-EUC-H` should be changed into the PostScript font name for the font to be used, here with EUC-JP encoding. The name `eucjp` can be changed to something else, as long as it is the same in the `BODY` and `font` blocks. You may not (yet) use a font having vertical writing mode.

The support for Japanese has only recently been added, and so far only been tested by me. Unfortunately, I do not know any Japanese, and cannot verify that what comes out is correct. Also, the line breaking is probably not correctly done.

Please let me know if you are aware of any free PostScript fonts for Chinese, Korean, etc.

The quotation element Q

The element Q, that is used for short quotations, is language sensitive; so the quotation marks are chosen based on current language.

Here follow a few quotes that illustrate the languages that have their quotation marks predefined. See the `quote` block how to extend the list of quotation marks. Note: since the Q element is currently not implemented in most browsers, you will probably not see any quotation marks at all if you are reading this in a browser (this document is best printed with html2ps ;-).

“A horse! a horse! my kingdom for a horse!” – *Richard III* by W. Shakespeare.

”Blott Sverige svenska krusbär har” – Carl Jonas Love Almqvist (Swedish poet).

»Der er et yndigt land« – National anthem of Denmark.

«Seiern er vår!» – Almost any Norwegian nowadays...

”Sukuvirttä suoltamahan, lajivirttä laulamahan” – *Kalevala*, national epos of Finland.

“... y veámonos, como dijo un ciego a otro” – *Don Quijote de la Mancha* by Cervantes.

« Je pense donc je suis » – René Descartes.

„Zum Denken sind wenige Menschen geneigt, obwohl alle zum Rechthaben“ – Arthur Schopenhauer.

«Eppur si muove» – Galileo Galilei.

Hyphenation

The character entity can be used to explicitly specify possible hyphenation points.

It is also possible to get automatic hyphenation of the text. This requires hyphenation pattern files from the TeX distribution. A hyphenation pattern file for English is included with html2ps. Pattern files for other languages are available from the CTAN archives in, for example

- England
- Germany
- USA
- Japan

The hyphenation patterns for words containing only ASCII characters should work correctly. But it may be problems with other words, since the authors of these hyphenation pattern files use different (but in TeX equivalent) syntax to represent non-ASCII characters.

Please note that although TeX hyphenation patterns are used, the algorithm used is not as sophisticated as in TeX. In fact, you should not expect TeX quality in *any* part of this program!

Hyphenation patterns for important languages such as Swedish, Icelandic, Danish, Norwegian and Finnish, as well as a few exotic languages such as English, Spanish, German and French have been tested and seem to work correctly :-).

Known bugs

The following bugs are with respect to HTML 3.2 – at the time of writing the current W3C Recommendation for HTML. The support for HTML 4.0 is described in another section of this document.

The IMG attributes ALIGN=LEFT and ALIGN=RIGHT are not recognized. This means that text flow around images does not work. Also, the IMG attributes WIDTH and HEIGHT are ignored.

The TYPE attribute in unordered lists UL is not supported.

A few deviations from the TABLE element as specified in HTML 3.2: It is not possible to set the border width of a table. The WIDTH and HEIGHT attributes of the TH and TD elements does not work correctly (in my opinion, these attributes are bugs in the HTML specification itself!). The CELLSPACING attribute is not implemented as described in the specification; instead the value of the CELLPADDING attribute is increased by half the value of CELLSPACING.

Rendering HTML tables well is a non-trivial task. For "real" tables, that is representation of tabular data, html2ps usually generates reasonably good output. When tables are used for layout purposes, the result varies from good to useless. This is because a table cell is never broken across pages. So if a table contains a cell with a lot of content, the entire table may have to be scaled down in size in order to make this cell fit on a single page. Sometimes this may even result in unreadable output.

Page breaks are occasionally done in bad places: for example directly after a (long) heading, and before the last line in a paragraph.

Todo

Actually, what *really* needs to be done is a complete rewriting of the code. It started out as a small hack – originally a sed script – and has now grown substantially, but it is still a hack... It is quite unlikely that this rewriting will take place, though – at least not in the near future.

A more realistic project is to implement the parts of HTML 4.0 that are relevant for printing. Much is already implemented, but there remain a lot to do regarding internationalization.

Support for style sheets has also high priority. Currently, a small part of CSS1 is supported, but only in the form of personal style sheets (using configuration files); style information within HTML documents is ignored.

Being a (currently inactive) PhD student in mathematics, I hope to be able to implement the suggested standard for expressing mathematical expressions in HTML documents. However, rendering mathematical formulas well is a very difficult task indeed.

All new features added have made the script quite large, and significantly slower than earlier versions. I will try to find some way to speed it up somewhat.

I am grateful for all suggestions how to improve html2ps. In fact, many of the current features have been implemented after suggestions from other users.

Miscellaneous

Environment variables

HTML2PSPATH
: This variable specifies the directories to search for configuration files. It should be a colon separated list of directory names. Use a dot '.' to denote the current directory. An empty directory name (as in ':dir', 'dir1::dir3', or 'dir:') will expand to the directory where the global configuration file is. The default value is '.:', that is: search the current directory first, and then the global one.

DSC compliance

By default html2ps produces PostScript code that is *not* DSC compliant. The practical implication of this is that it will not be possible to use the code with PostScript filters for n-up printing, reordering of pages, etc. It is also not possible to jump to a certain page directly in previewers such as Ghostview.

The advantage with the non-DSC code is that it can be 'reused' by html2ps: you can rebuild an already generated PostScript file using new command line options and configuration files. This is done by running html2ps with the new options, and with the old PostScript file as input. This can save a lot of time and bandwidth when converting remote documents.

It is possible to generate DSC compliant PostScript by using the option -D, but this requires that Ghostscript is installed, and it can take quite some time to do. Note: if you are producing PostScript files for others to download and print, it is **strongly** recommended that you generate DSC compliant code (i.e. convert *with* the -D option).

Extensions

A few extensions to HTML 3.2 and 4.0 are recognized by html2ps:

A page break can be forced by including any of these in a document:

```
<HR class=PAGE-BREAK>

<?page-break>

<!--NewPage-->
```

If you want html2ps to ignore certain parts of a document, use the following construction:

```
<DIV class=NOPRINT>
  This will not be printed...
</DIV>
```

This can be used to avoid getting navigation bars etc in the printout.

The BANNER, FIG, and MATH elements from the expired HTML 3.0 specification are also supported. Note that the support for MATH is far from complete, and requires that TeX and dvips are installed.

Conversion to PDF

The PostScript code generated by html2ps contains calls to the `pdfmark` operator, that are instructions to a PostScript-to-PDF converter (such as version 5.0 or later of Aladdin Ghostscript, or Adobe Acrobat Distiller). These instructions generate hyperlinks in the PDF document, from the hyperlinks in the original HTML documents. A PDF outline entry, or bookmark, is also generated from the HTML headings.

By default the links in the PDF document will be represented by a box with a border. You may instead have the link text coloured, by for example specifying '`A:link { color: red }`' in a configuration file, and use the option -U when converting.

Where to get other software

Here follow some addresses where one can obtain software used by html2ps. Several of these programs are available from many other program archives all over the world. Note also that it is not necessary to install all these packages.

- Perl
- ImageMagick and PerlMagick
- pbmplus
- netpbm
- jpegsrc
- weblint
- libwww-perl
- www.pl and network.pl

- Ghostscript

Reporting problems

Please note that html2ps comes with *no* support (this is unpaid work done on my spare time). However, I *am* interested in making the program as useful as possible, so do not hesitate to report any problems. But please carry out the following steps first:

1. Check that the HTML documents you are converting have correct syntax – rendering valid documents is difficult enough, do not expect *anything* from documents with invalid syntax. To check the syntax, there are validation services available, and also online syntax checkers such as weblint.
2. Execute html2ps with the option -d. This writes out debugging information to the terminal, as well as to a file 'html2ps.dbg'.

If you, after inspecting the results from these two steps, believe that there is a problem with html2ps; mail the content of the file 'html2ps.dbg' together with a description of the problem to html2ps-bugs@tdb.uu.se. If you fail to produce the file 'html2ps.dbg', instead please include as much details as possible: the version of html2ps, The Perl version, your platform, etc.

If possible, also include the HTML document, or a URL – preferably containing a minimal example illustrating the problem.

Acknowledgments

There are several persons that have been helpful with the development of this program – by supplying patches, suggesting improvements, etc. I have most certainly forgot a few names; please forgive me if you are missing your name in this list: John Bowe, Neil Bowers, Reinhard Denner, Craig Oda, Thomas Mohr, Dave Raggett, John Sinues, Larry W. Virden, and Chenxi Wang. My colleagues at the Department of Scientific Computing have also been a great help by testing early versions of the program.

Jan Kärrman
Dept. of Scientific Computing
Uppsala University
Sweden
jan@tdb.uu.se