A guide to using $\TeX 4\text{ht}$ as a tool for publishing $\LaTeX$-documents on the web.

Simen Kvaal

\begin{verbatim}
class point (x,y); real x,y;
    begin ref (point) procedure plus (P); ref (point) P;
      plus := new point (x+P.x, y+P.y);
end point;

point class polar;
    begin real r,v;
      ref (polar) procedure plus (P); ref (point) P;
      plus := new polar (x+P.x, y+P.y);
      r := sqrt (x+2+y+y+2);
      v := arctg (x,y)
end polar;
\end{verbatim}
A guide to using \TeX4ht as a tool for publishing \LaTeX-documents on the web.

Simen Kvaal

June 10, 2003

Contents

1 Introduction 3

2 The basics of \TeX4ht 4
  2.1 \TeX4ht in general . . . . . . . . . . . . . . . . . . . . . . . 4
  2.2 How to invoke \TeX4ht . . . . . . . . . . . . . . . . . . . . 4
  2.3 Hello, world! . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  2.4 Parameters to the \texttt{tex4ht} package . . . . . . . . . . . . . 6

3 The image-generating process 6
  3.1 Strategy for creating bitmaps . . . . . . . . . . . . . . . . . 6
  3.2 All the bad things: \texttt{tex4ht.env} . . . . . . . . . . . . . . . 7
  3.3 Making amends with \texttt{cscript.sh} . . . . . . . . . . . . . . 7

4 A recipe for \TeX4ht conversion of \TeX{} documents 8
  4.1 Step 0: Prerequisites . . . . . . . . . . . . . . . . . . . . . . . 8
  4.2 Step 1: Creating the scripts . . . . . . . . . . . . . . . . . . 9
  4.3 Step 2: Preparing your \LaTeX{} source . . . . . . . . . . . . . 9
  4.4 Step 3: Compiling and error checking . . . . . . . . . . . . . . 10
  4.5 Step 4: The web page . . . . . . . . . . . . . . . . . . . . . . 10

5 Short version of the recipe 11

6 Linking slides from \texttt{prosper} to \TeX4ht 12
  6.1 The \texttt{hyperref} package . . . . . . . . . . . . . . . . . . . 12
  6.2 The missing links inside \TeX4ht . . . . . . . . . . . . . . . . . 13
  6.3 An example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

7 Making a Simula report from your \LaTeX{} document 14
### Listings

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>cscript.sh</td>
<td>15</td>
</tr>
<tr>
<td>8.2</td>
<td>customlinks.tex</td>
<td>17</td>
</tr>
<tr>
<td>8.3</td>
<td>tex4ht.env</td>
<td>17</td>
</tr>
</tbody>
</table>
1 Introduction

\LaTeX\ is a very powerful and flexible typesetting package, which in addition to being freely available also is widely supported. Although the print quality of the output may be outstanding and despite the fact that high quality PDF versions of your document may be published, \LaTeX\ unfortunately lacks any direct support for creating documents readily published on the web, compatible with any graphical web-browser. (And we cannot really blame the authors of \TeX\ and \LaTeX: It would be like blaming green-grocer for not selling automobiles.)

In this document, we will go through the usage of \TeX\4ht to publish large documents on the web. \TeX\4ht is a flexible extension to \LaTeX, making the web publishing of complex \LaTeX\ documents relatively easy. It is a combination of a standard \LaTeX\ package and accompanying post-processing utilities.

We will develop a “recipe” for post-processing \LaTeX\ documents, with navigation through sections, consistent layout and robust graphics conversion. On our way there are some obstacles, but once overwon \html\ versions of your documents are readily created.

The reader is not assumed to go into detail in all the covered problems and topics, but is encouraged to spend some amount of time reading the different sections. For creating \html\ versions of \LaTeX\ documents however, only knowledge of the step-by-step recipe is needed. The really enthusiastic and impatient reader may skip directly to section five.

We will also discuss some aspects of the powerful \prosp\ \LaTeX\ package for creating delicate slides for presentations. More specifically, we want to be able to intertwine the web pages created with \TeX\4ht with the presentations, making it possible to refer to written material in an elegant manner on-screen.

Immediately after this introduction, we proceed with the basics of \TeX\4ht in section 2. The whole of section 3.1 is devoted to the process of converting special content to graphical material in the web pages, since this is the main chore with \TeX\4ht. In section 4 we state the long version of the “recipe” for creating good-looking \html\ documents from \LaTeX\ documents. Section 5 is the short version for the impatient reader, with short and concise instructions. As mentioned above, the reader may skip directly to this: Understanding the other material is not absolutely necessary. The section describes among other things how to obtain a package with all needed files for performing the wizardry.

Section 6 deals with the aforementioned linking of \prosp\ documents to documents published with \TeX\4ht.

We also include section 7 describing how to add the official Simula cover to your \LaTeX\ document. It is included because one often needs to publish the report on the web at the same time as one creates the official report. It is therefore convenient to have the recipes for both processes at the same place.

In the 8th and last section I have included listings of important files, although one can download them from \cite{6}. See also the short version of the recipe in section 5.

We will assume — of course — that \TeX\4ht and \prosp\ are installed alongside \LaTeX\ on your system.
2 The basics of \TeX4ht

2.1 \TeX4ht in general.

We will here talk about \TeX4ht in general terms. For reference, I strongly recommend reading “The \L\A\TeX Web Companion” [1], as the documentation in this book is very informative. It is a very handy book on other matters as well. See also the official web site [2] for \TeX4ht. However, we will touch the important basics here, and everything needed to get a working knowledge of the system will be covered.

There are several other alternatives besides \TeX4ht for creating html out of \L\A\TeX, but \TeX4ht is definitely the most elaborate and gives the best results after some work. The main drawback is this amount of work. On the other hand, the benefits are much more important: Because \TeX4ht works on a low level (\L\A\TeX), and \L\A\TeX works on top of this, complex structures are easily handled and may be tailored to get the results one want. To compare, \LaTeX2html is merely a Perl script parsing \L\A\TeX code, and staggers when the author does “tricks,” as is the case in all projects bigger than your average “hello world!”-project.\footnote{Indeed, if you go through the simple “Hello world” \TeX4ht example outlined below and study the html code, you can see how \L\A\TeX{} commands such as \texttt{\LaTeX{}} are constructed internally, because \TeX4ht works on such a low level.}

How does \TeX4ht work? It is basically a combination of a \L\A\TeX package and a set of post-processing utilities. The \L\A\TeX{} package modifies the dvi output\footnote{dvi means “device independent.” \L\A\TeX generates a dvi file that can be converted to other device dependent media, such as PDF, PostScript and (you guessed it!) HTML.} from the compilation process, adding information for the post-processing utilities to digest. These utilities generate the html code and the pictorial representation of content such as included figures and mathematical formulae. We will have more to say on this later.

2.2 How to invoke \TeX4ht

To invoke \TeX4ht inside the \L\A\TeX document, one simply issues a \texttt{\usepackage\{tex4ht\}} command in the preamble immediately after the \texttt{\documentclass} statement. (There are several optional arguments, but we won’t worry about them now.) This makes the dvi-output all messed up, probably making it incompatible with your favorite dvi-driver (eg. \texttt{dvips}). This is because the dvi-file now is tailored to create html files (making the dvi device dependent). This is done with the external post-processing commands \texttt{tex4ht} and \texttt{t4ht}.

The \texttt{tex4ht} command takes the dvi and some other input files created by the \TeX4ht{} package and creates html-files and an idv-file. Try looking at this with e.g. \texttt{xdvi}, and you’ll discover that it really is a dvi: Each page holds one item of some kind. These are items that \TeX4ht cannot convert into html directly. So, the program \texttt{t4ht} is used to create bitmap versions of each item. This process relies on \textit{external utilities}, as explained below. Thus, \texttt{t4ht} might seem rather
fragile, but on the other hand it is highly customizable. Take a look at figure 1 as this illustrates the concept.

![Flow of files and information when using TEX4ht](diagram.png)

**Figure 1: Flow of files and information when using TEX4ht**

**Note:** If you want to create an ordinary non-HTML dvi after you have used TEX4ht (that is, if you remove the proper `\usepackage` command), you must delete the aux-file created when you were using TEX4ht.

The actual commands to use when creating your HTML version is:

```latex
latex document
latex document
latex document
tex4ht document
t4ht document
```

Note the several invocations of `latex`. This is because TEX4ht needs to get HTML tables, cross references and so on correct. Sometimes even more runs of `latex` are needed! The above process can be run with the single command `ht latex document`, and is usually all right for a start. If your document is very complex and your HTML-document turns out to lack some references for instance, then maybe you need one or two more `latex` runs before post-processing with `tex4ht`. And if you are using for example BibTEX, you almost certainly need to make your own sequence of commands.

### 2.3 Hello, world!

Let us see a small example using TEX4ht. This shows the default behaviour of the system, and is not particulartely convincing. Let this be the input LATEX document (hello.tex):

```
\documentclass{article}
```
\usepackage{tex4ht} 

\begin{document}
\section{Hello, world!}
This is a simple \LaTeX{} document.
\section{Ars magna}
This is easy:
\[
\begin{array}{c}
a^2 + b^2 = c^2 \\
\end{array}
\]
But what about
\[
\begin{array}{c}
\alpha^3 + \beta^3 = \gamma^3 \\
\end{array}
\]
\end{document}

Run \texttt{ht latex hello} and watch. Note how slow the process of generating bitmaps from the mathematical formulae is and how bad the maths look when hello.html is viewed with for example Mozilla. (The actual code is rather complicated, so we won’t display it here.) Apart from this, the document is quite nice. But with some simple adjustments to the configuration, we will be able to produce both quicker and better results. This is the topic of section 3.1.

\section*{2.4 Parameters to the \texttt{tex4ht} package}

When our recipe for creating web documents is finished, we won’t need to worry about the parameters for the \texttt{tex4ht} package. Let us mention some of the parameters anyway for completeness:

If parameters are supplied at all, the first one must either be the name of a configuration file (see [1]) or \texttt{html}. We will always use the latter. A common mistake is to forget that the first parameter to \texttt{tex4ht} is special.\textsuperscript{3}

Other parameters include:

- \texttt{png} or something similar to indicate the “planned” extension of images in the HTML document. (This is a rather confusing matter, since this only applies to some of the images generated.) We will always use \texttt{png} here. All our images will be of this type when we follow the recipe to be created.

- 1 through 4 to invoke certain pre-defined setups concerning spreading the html document over several files. Higher numbers produce a deeper nesting of documents. (1 divides at \texttt{part}, 2 at \texttt{chapter} et.c.) In addition, \TeX{}4ht supplies navigation buttons on each page. The layout of these can be customized; see [1].

\section{3 The image-generating process}

\subsection*{3.1 Strategy for creating bitmaps}

As explained in section 2, \TeX{}4ht needs external utilities for creating the bitmaps used in the web-documents. The default set up is a combination of

\textsuperscript{3}“Common mistake” of course means that I’ve been confused about it a lot.
dvips, gs (GhostScript) and ImageMagick’s convert utility [4]. Unfortunately this is a rather slow process, and not a very good one when it comes to antialiasing, and it’s definitely not suited for typesetting uses. However, the dvi2bitmap tool available freely from [3] is efficient and converts dvi files directly to png or gif files. Dvi2bitmap works almost like a normal dvi-driver, but has no support for so-called dvi-specials (such as included eps-figures et.c.) Thus, a combination of the two is required: We use dvi2bitmap whenever we encounter maths and special symbols as this is efficient and produce good results, but use the aforementioned dvips/gs/convert-process for items that dvi2bitmap cannot handle, such as included PostScript figures.

Note: In the trouble-shooting section on the official web page for TEX4ht [2] it is mentioned that you can directly configure TEX4ht inside your document to process e.g. \includegraphics in a special way. I have not been able to use the method successfully myself, though you might give it a try if you are not satisfied with the results.

3.2 All the bad things: tex4ht.env

The converting process is initiated in theTEX4ht configuration file tex4ht.env. From the man pages of TEX4ht one can locate this. It is useful to have a copy in your working directory and edit it when needed. TEX4ht will use whatever configuration file it finds first, and the current working directory is the first place it looks. You may also get the configuration file in the package from section 5.

The converting process is the lines in tex4ht.env starting with G. I have simply replaced those lines with a call to a small script. Thus, the converting-process in my tex4ht.env file looks like:

G./cscript.sh %%1 %%2 %%3 2

The configuration file format is ugly and crammy, and it takes some effort to get into it. (And to make things worse, blank lines are actually harmful, as the documentation so nicely puts it.) For the image-generating process however, all that is needed is to replace the few lines that begin with a G.

3.3 Making amends with cscript.sh

The Bash script cscript.sh ("c" for conversion) obviously takes four parameters: We pass the parameters %%1-%%3 on the command line in tex4ht.env. The three parameters are the name of the dvi file, the page number and the name of the desired output file, respectively. (This is a good thing with tex4ht.env: An external utility may get all the information it needs from it.) As the fourth parameter the script takes an integer between 0 and 3 indicating the desired output quality of the bitmaps: 0 is quick-and-very-dirty, 1 has no antialiasing while 2 and 3 employ antialiasing, with 3 the best and by all means the slowest.\footnote{It is not a true antialiasing-method: We simply create a huge bitmap and scale it down. (This is not suitable for text.) We really need a method that employs an adaptive sampling method. Quality setting 0 is the reverse, blowing up a way too small bitmap!}
The quality settings only apply to the `dvips-gs-convert`-process, as `dvipng`'s quality is always quite good. For example, the following call to `cscript.sh` reads page 42 of `mydvi.dvi` and creates the bitmap `outfilename.png` with the lowest quality:

```
./cscript.sh mydvi 42 outfilename.png 0
```

Several minor problems were encountered while investigating the conversion process, and all of them are fixed in the script. For example, the default for `dvipng` is to employ an alpha-channel (for semi-transparent pixels) when outputting `png` files. This is not compatible with Microsoft’s Internet Explorer browser and many imaging applications. The fix is to employ a white background instead, but this means that when choosing background colors in the `html` document, one needs to adjust `cscript.sh` directly.

`TeX4ht` is defining a lot of strings and macros, and really puts a strain on `TeX`'s capacity. If this is the case, be a wizard and increase it in your `texmf.cnf` file.

The program `dvipng` relies on the `kpathsea` library for finding fonts, but not all `TeX` distributions install this. If this is the case, then `dvipng` must use other means for locating the font files. This may turn out to be complicated. Luckily, the newest version (as of time of writing) can also employ the standalone version `kpsewhich` and this works fine. Note however, that the presence of either the library or the standalone is needed for the current script to work.

### 4 A recipe for `TeX4ht` conversion of `TeX` documents

Now we have come to the point of stating a working recipe for creating `HTML` versions of `LaTeX` documents with `TeX4ht`. We will first summarize the needed programs and files, and then state a development procedure. The recipe should be more than adequate for most projects, but when converting huge projects such as books with hundreds of pages, one might encounter bugs in `TeX4ht`. If this is the case, report the bug to the author of `TeX4ht` and/or work around the problem.\(^5\)

#### 4.1 Step 0: Prerequisites

Besides `LaTeX` with `TeX4ht` installed you will also need:

- `dvipng`, available for free from [3].
- `GhostScript` which usually is installed alongside `LaTeX`.
- `ImageMagick`‘s `convert` utility available from [4].

\(^5\)This guide is based on the work done converting a huge book written by several authors over a long time. This implied strange `LaTeX` vs. `TeX4ht` problems, but only minor. Thus, serious problems should be rare.
• A good web browser like Mozilla is really useful.

• The latest bug-fixes for \TeX\4ht, available from [2].

Locate the \texttt{tex4ht.env} configuration file and store it in a convenient place for editing as explained below. You can also use the one supplied in the file package mentioned in section 5.

Also locate your default \texttt{texmf.cnf} file and copy it into a convenient place, e.g., your working directory. During large projects \TeX\ will almost certainly run out of capacity. When this happens, locate the appropriate variable in the configuration file and modify it.

4.2 Step 1: Creating the scripts

Fetch the \TeX\4ht configuration file \texttt{tex4ht.env}. Edit the lines starting with a \texttt{G}. They should look like this:

\texttt{G./cscript.sh \%1 \%2 \%3 2}

You will need the aforementioned \texttt{cscript.h}, of course. Either get it from [6] or copy the listing at the end of this guide. (Note that you may need to change the location of the \texttt{bash} binary at the beginning.)

4.3 Step 2: Preparing your \LaTeX source

Most of the time, \TeX\4ht will accept every bit of \LaTeX code you have written. This is because \TeX\4ht works on a level just between \TeX and \LaTeX, so that every macro gets special treatment.

There is one difference, however, and it is an important one: Always place your macro definitions after the \texttt{\begin{document}}-command. This has subtle reasons, and I recommend that you look it up in [1] if you are curious.

It \textit{might} happen that one or two of your macros won’t compile. In that case, it is a bug in \TeX\4ht. Either find a workaround or report the bug if it is serious, although it is unlikely.\footnote{In my work with very complex documents I’ve encountered four small bugs which required only tiny modifications of the macro code. (Not the actual document code.)}

You might also get strange errors seemingly not related to a macro, but it probably is. Let’s say you have encountered an error at line 2651 in your document. It still might be your macro that isn’t working correctly; just pay attention to exactly \texttt{where} in the line the error occurs.

Now I have said a lot about errors, and you might get the impression that errors is something you’ll have to deal with a lot. This is not so, but it is good to be prepared and not to have to spend hours at errors with an easy solution.

Now for the preamble: You’ll need to invoke the \TeX\4ht package. Use this snippet:

\texttt{\usepackage[html,png]{tex4ht}}
If you want to split your document into several HTML files, add an integral parameter after `png` and a comma. (See the above part on parameters to \texttt{X4ht}.) For a big book project, 3 is typically a good value.

This is probably everything that is needed: One line of code in addition to moving your macros to after `\begin{document}`.

You may also want to set an option that forces every tiny inline equation to become a bitmap. By default \texttt{X4ht} creates HTML code for as many inline equations as possible, with varying degree of success. This option is described in [2] in detail:

\input{PicMath}{\EndPicMath}

Place the option after `\begin{document}`. Similar settings exist also for other math environments.

4.4 Step 3: Compiling and error checking

You are now ready to compile. You can for example run the command

\texttt{ht latex basefilename}

as described previously. You may also compose your own arrangement of \texttt{latex}, \texttt{tex4ht} and \texttt{t4ht} commands if the previous doesn’t fix all the tables et c. (This is typical in a big project.)

Note that in \texttt{script.sh} you specify the bitmap quality. Due to the long conversion times for included PostScript figures, I recommend that you start with a quality setting of 0 until you are happy with the look of the rest of the HTML document. The quality setting will not affect the rest of the HTML generating process.

You can of course preview your HTML pages in a web browser. The base file has the same name as your document.

4.5 Step 4: The web page

After you are satisfied with the results, copy all the .html and .png files to your published web area.

\textbf{Note:} When the surfer enters your directory, and if you do not have an `index.html`, he/she will see all the files in the directory. If you don’t want that, create an `index.html` to prevent this.

\textbf{Note:} Also note that all the files created resides in your working directory and should be placed in \textit{the same} published directory. You can specify options to place the .html-files in a special place, but you must modify \texttt{script.sh} yourself to place the images there as well.
5 Short version of the recipe

In this section we present a condensed version of the previous recipe, without all the gory details. Assuming that \LaTeX, \TeX4ht, ImageMagick and GhostScript are installed, you may download a file package from [6] containing the other essentials. These include:

- A precompiled binary for dvi2bitmap.
- The cscript.sh script.
- A version of tex4ht.env adapted to use cscript.sh.
- The customlinks.tex macros for customized HTML links.

If you have skipped directly to this section without reading about the background for these components, you will not necessarily understand what they do or what they are needed for. Consult the other sections if you are bewildered.

Now for the recipe for converting your \LaTeX\ document into HTML:

1. Download the file package tex4ht-files.tar.gz and decompress it into your working directory. (That is, the directory from where you run the latex-commands.)

2. Modify your source code:
   - Add \usepackage[html,png]{tex4ht} to your preamble, that is before the \texttt{\begin{document}} command.
   - Move all macro definitions (for example \texttt{\newcommand} and \texttt{\newenvironment} definitions) to after the \texttt{\begin{document}} command.

3. Compile your document:
   - You may run \texttt{ht latex document}, where \texttt{document} is the file name of your \LaTeX\ document without extension.
   - You may also run your own set of commands if the above does not seem to cope with all your tables or similar:
     
     \begin{verbatim}
     latex document
     latex document
     latex document
tex4ht document
t4ht document
     \end{verbatim}
     This is the sequence actually run with \texttt{ht latex document}. Run additional \texttt{latex} commands, \texttt{bibtex} commands or whatever before the \texttt{tex4ht} and \texttt{t4ht} commands.

4. If compilation succeeded, your HTML version is stored in \texttt{document*.html} and the graphic content is stored in \texttt{document*.png}. Additional cascading style sheet specifications reside in \texttt{document*.css}. All these files must be published in the same directory on the web for the document to be displayed correctly.
6 Linking slides from prosper to \TeX4ht

This section deals with the issue of creating hypertext links when using the package \texttt{prosper} \cite{5}, and more specifically hypertext links to HTML documents created with \TeX4ht. This is not meant as a guide to \texttt{prosper}, as the documentation that comes with the package is a good guide in itself, and since it is not very difficult to learn anyway. But we mention that \texttt{prosper} allows you, as a \LaTeX user, to create high-quality presentation slides merely with simple \LaTeX code. The pdf file that results from this process is comparable to those created with Microsoft’s \texttt{PowerPoint}. The most important advantages of using \texttt{prosper}, is that it is free and that it supports all typical \LaTeX structures, such as mathematics. (We all know that \texttt{PowerPoint} is bad at this.)

Why bother with these hyperlinks?

Imagine yourself at a conference, meeting, or whatever, displaying your slides for the audience. Maybe you are presenting your latest findings in time-travel technology or mind transfer serums. Then, at some point, you refer to some of your articles showing detailed calculations or witty quotes. With the hyperlinks inside your slides, you may click on this anchor and, behold, the web browser pops up displaying the exact contents of your article! Neat.

Moreover, with the knowledge of how to put hyperlinks into \texttt{prosper}, you may also link to regular web pages with relevant content. This might also be valuable. (But then, you don’t need most of the material in this section.)

6.1 The \texttt{hyperref} package

The \texttt{hyperref} package provides a flexible interface to hypertext links. It is included when you include \texttt{prosper}. This means that most of the options offered by \texttt{hyperref} is set “behind the curtains,” but for most uses this should not be a hindrance. A thorough description of \texttt{hyperref} is found in \cite{1}, and we will only deal with the basics here.

A plain, vanilla hypertext link is inserted with the \texttt{\href} command:

\begin{verbatim}
\href{url}{text}
\end{verbatim}

Here, \texttt{url} is the \textit{fully expanded} URL to the desired page, and \texttt{text} is the anchor shown visually. Note that characters such as \# and \~ should not be escaped. Note also that the link must be fully expanded. If not, the link will typically get a \texttt{file:} in front. This unfortunately makes the document less portable. To overcome this problem a bit, a command \texttt{\hyperbaseurl} is provided.\footnote{Due to a minor (?) bug, inserting a tilde in the address results in an error. Try to work around this in some way, e.g. by using \texttt{\string\~} to insert a tilde.} Set this to e.g. \url{http://www.something.com/user/} and you can omit the full expansion of the hyperlinks. (You may still expand external links, of course.)

Because of the colorizing parameters being obscured by the early inclusion of \texttt{hyperref}, one has to use other means for creating colored links. One way is
whose usage is the same as the original \href command.

6.2 The missing links inside \TeX\4ht

Now, we turn to the creation of human readable anchors inside the HTML files generated with \TeX\4ht. We wish to be able to direct our hypertext links inside the \texttt{prosper} presentation to a given chapter, section or page. In HTML, this is done with \texttt{<a name='whatnot'>} tags; anchor tags. \TeX\4ht creates such tags by default for internal cross linking between equation references, figure references and so on. Unfortunately, the format of these are not known a priori, and we must supply reliable tags ourselves.

This might seem clumsy at first: why not use the tags already present? The simple implementation we choose and the usage proves the method to be fine, though. Analyzing \TeX\4ht’s behaviour may be a tedious task, and that’s why we decide on the presented technique.

The main ingredients of the implementation is to exploit \TeX\4ht’s “hooks.” When the author issues e.g. a sectioning command, configurable hooks generate HTML material in the \texttt{tex4ht} post-processing stage. This allows us to create anchors with literal description of the section. The needed configurations and macros are collected in \texttt{customlinks.tex}, whose usage is described next.

To invoke the custom hooks, simply \texttt{\input} the macro file immediately after the \texttt{\begin{document}} command. On every chapter, section and subsection, a HTML anchor is created. One can also issue custom-made anchors with the commands \texttt{\customlink} and \texttt{\pagelink}:

- \texttt{\pagelink} creates an anchor to the current page on the form \texttt{pnnn}, where \texttt{nnn} is the page. This feature seems rather silly, but it might prove useful.
- \texttt{\customlink\{name\}} creates an anchor named \texttt{name} at the current position in the web document.

Note: When using these commands, do not insert empty lines on both sides of them. If you do, it will result in an empty but visible paragraph in the document. (The anchors still work, of course, whether you remember this or not.)

Whenever a custom hook is encountered, the macros issue a message to the log. To extract the interesting messages, issue the command

\texttt{cat doc.log | grep Customlink >links.txt}

to generate a file containing a description of the links. Typically a line reads:

--- Customlinks says: Link to section 2.1: doc.html#sec2-1
So, when one needs to link to section 2.1, use this link.

The `customlinks.tex` contains only basic \LaTeX programming, so extending it to ones own need should be no difficult task. As for now, starred sections are not implemented, for example.

The file `customlinks.tex` is available alongside this document from [6]. In addition, it is included in the file package described in section 5.

### 6.3 An example

On the web page [6] you may view an example showing how to link \texttt{prosper} documents to \texttt{html} documents and \texttt{html} documents created with \texttt{TeX4ht} in particular.

The example consists of several files:

- `dracula.tex`: This is the document that is being converted with \texttt{TeX4ht}.
- `dracula.html`: This is the document after conversion with \texttt{TeX4ht}.
- `customlinks.tex`: The previously mentioned file that defines \texttt{TeX4ht} hooks for customized anchors inside the \LaTeX document.
- `slides.tex`: \texttt{Prosper} slides utilizing the anchors generated during compilation of `dracula.tex`.
- `slides.pdf`: The compiled \texttt{prosper} presentation; clickable and all.

Take a look at the source codes. Hopefully this will in conjunction with the above paragraphs be informative.

### 7 Making a Simula report from your \LaTeX document

In this section we state the procedure for adding the official Simula report cover to your \LaTeX document. We include this description inside this document, because usually when creating Simula reports, one also publish it on the web in \texttt{HTML} format — thus having the recipes all in one place might be a good idea.

Adding the Simula cover is really simple, but remember that some \LaTeX packages that you use might turn out to be incompatible with the Simula cover style. We will return to this issue later.

For more information on the usage and bug reporting, see the Simula intranet [7].

To add a Simula-cover, three simple steps are needed:

1. Include the \texttt{simulacover} package just after the \texttt{\documentclass} declaration:

   \begin{verbatim}
   \usepackage{simulacover}
   \end{verbatim}
2. If you need or want another way to display author names etc. on the cover, add one or more of the following commands:

   - \simdoctitle{Alternate title}
   - \simdoctext{Alternate author description}
   - \simdoctype{For example ‘memo’}
   - \simdoctext{This goes on the front cover as well}
   - \simdocnumber{Report number}

3. Add the command

   \makesimulacover

   just before the \maketitle command. This will create the cover with the author names, the title of the document and so on.

   The simulacover package is installed on Simula, and is available on all local machines. If you want to compile your document on a laptop or another machine running \LaTeX{}, get a copy of the latest\textsuperscript{8} version of the package and put it into the same directory as your document. \LaTeX{} will then find the package and include it properly. There is no need for actually installing it to your \LaTeX{} system.

   If it turns out that for some reason, your document does not compile correctly or does not compile at all, the problem is probably that inclusion of some package interferes with simulacover. Currently, the fancybox package is known to exhibit this behaviour.

   A simple solution to this problem is to create a new document containing only the Simula cover, and no text. Create a PostScript version of this and add the two first pages to the PostScript version of your original document.

   Alternatively, work around the incompatibilities in some other way; using different packages or similar.

8 Listings

Here are the listings for various files mentioned. They can also be downloaded from \cite{[6]}. 

8.1 cscript.sh

```bash
#!/bin/bash

echo 'CSCRIPT.SH -- convert dvi to bitmap.'
```

\textsuperscript{8}Note: The version on the Simula intranet web site is at the time of writing outdated! We suggest that one locates the package on one of Simula’s Linux stations and copies this to one’s own computer.
exit status and their meanings:
0 - success
1 - bad command line

proper invocation:
cscript.sh thedvifile thepage theoutputfile quality

quality is a parameter describing the quality of the converting process
when using the horribly slow plan-b-method
0 - dirty, 1 - low, 2 - medium, 3 - high.

# check if command line arguments exists
if [[ ! "$1" || ! "$2" || ! "$3" || ! "$4" ]]
then
echo "CSCRIPT.SH Error: bad command line"
echo " proper invocation: CSCRIPT.SH thedvifile thepagenumber theoutputfile quality"
exit 1
fi

# check quality settings ...
quality=$4
if [[ "$quality" -lt 0 ]]
then
  quality=0
fi
if [[ "$quality" -gt 3 ]]
then
  quality=3
fi

echo "CSCRIPT.SH> quality set to $quality"

# try to convert with dvi2bitmap.
rm -f $3
success=0
if ./dvi2bitmap --font-mode=ibmvga --page-range=$2 --magnification=4 -s 4 --output-type=png --resolution=110 --process=transparent=false --output=$3 $1
then success=1
fi

# check if the outputfile was generated.
# if not, create image with dvips/gs/convert.
if [[ ! -e "$3" ]]
if (( success = 1 ))
then
echo "CSCRIPT.SH> dvi2bitmap made it!"
# enhance the bitmap
convert -unsharp 1.0x0.5 $3 fish_og_slips.gif
convert fish_og_slips.gif $3
else
  echo "CSCRIPT.SH> dvi2bitmap failed! running the dvips/gs/convert-process..."
dvips -Pcmz -Pamz -mode ibmvga -D 220 -f $1 -pp $2 > TEMPORARY.ps
res_str='-r25x25'
scale_str='440%
if [[ "$quality" = 0 ]]
then
res_str='-r110x110'
scale_str='100%
fi
if [[ "$quality" = 1 ]]
then
res_str='-r60x60'
scale_str='58.75%
fi
if [[ "$quality" = 2 ]]
then
res_str='-r220x220'
scale_str='50%
fi
if [[ "$quality" = 3 ]]
then
res_str='-r330x330'
scale_str='33.33%
fi

8.2 customlinks.tex

\newcommand{\logcomment}{--- Customlinks says: }
\newcommand{\chapterlink}{\HCode{<a name='ch\arabic{chapter}'></a>}}
\newcommand{\sectionlink}{\HCode{<a name='sec\arabic{chapter}-\arabic{section}'></a>}}
\newcommand{\subsectionlink}{\HCode{<a name='subsec\arabic{chapter}-\arabic{section}-\arabic{subsection}'></a>}}
\newcommand{\subsubsectionlink}{\HCode{<a name='subsubsec\arabic{chapter}-\arabic{section}-\arabic{subsection}-\arabic{subsubsection}'></a>}}
\newcommand{\reportchapterlink}{\typeout{\logcomment Link to chapter \thechapter: \FileName\#ch\arabic{chapter}}}\newcommand{\reportsectionlink}{\typeout{\logcomment Link to section \thesection: \FileName\#sec\arabic{chapter}-\arabic{section}}}\newcommand{\reportsubsectionlink}{\typeout{\logcomment Link to subsection \thesubsection: \FileName\#subsec\arabic{chapter}-\arabic{section}-\arabic{subsection}}}\newcommand{\reportsubsubsectionlink}{\typeout{\logcomment Link to subsubsection \thesubsubsection: \FileName\#subsubsec\arabic{chapter}-\arabic{section}-\arabic{subsection}-\arabic{subsubsection}}}\newcommand{\reportcustomlink}[1]{\typeout{\logcomment Custom link named '#1': \FileName\##1}}\newcommand{\customlink}[1]{\HCode{<a name='#1'></a>}\reportcustomlink{#1}}\newcommand{\thispage}{\arabic{page}}\newcommand{\reportpagelink}{\typeout{\logcomment Page-link at page \thispage: \FileName\#p\thispage}}\newcommand{\pagelink}{\HCode{<a name='p\thispage'></a>}\reportpagelink}\\Configure{chapter}{\chapterlink\reportchapterlink}{}{}{}\Configure{section}{\sectionlink\reportsectionlink}{}{}{}\Configure{subsection}{\subsectionlink\reportsubsectionlink}{}{}{}\Configure{subsubsection}{\subsubsectionlink\reportsubsubsectionlink}{}{}{}
\% To generate a report of links generated:
\%
\% cat bok.log | grep Customlinks > links.txt
\%\%
\% Note that when inserting "custom links," don't insert blank lines
\% after the \customlink-command. This will generate an extra (empty
\% but visible) paragraph!
%
8.3 tex4ht.env

% tex4ht.env / tex4ht
% %
% \%Notes:
% % 1. empty lines are harmful
% % 2. place this file in your work
% % directory and/or root directory
% % and/or in directory 'tex' of your
% % choice. In the latest case, compile
% % tex4ht.tex with #define HTFDIR env.
% %
References


[2] Official web site for \TeX{}4ht:
http://www.cis.ohio-state.edu/~gurari/TeX4ht/ and
http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn2.html

[3] Official web site for dvi2bitmap:
http://www.astro.gla.ac.uk/users/norman/star/dvi2bitmap/

[4] Official web site for ImageMagick:
http://www.imagemagick.org/

[5] Official web site for prosper:
http://prosper.sourceforge.net/

[6] The author of this document’s personal web site:
http://www.simula.no/~simek/
Link to file package in section 5:
http://www.simula.no/~simek/tex4ht-files.tgz

[7] Simula cover \LaTeX{} style at The Simula Research Lab. intranet:
https://www.simula.no/intranet/support/unix/software_pages/latex.shtml