The **lwarp** package

LaTeX to HTML5

v0.31 — 2017/05/15

bd@BDTechConcepts.com

Abstract

The lwarp package allows LaTeX to directly produce HTML5 output, using external utility programs only for the final conversion of text and images. Math may be represented by SVG files or MathJax.

Documents may be produced by LaTeX, LuaLaTeX, or XeLaTeX. A `texlua` script removes the need for system utilities such as `make` and `gawk`, and also supports `xindy` and `latexmk`. Configuration is automatic at the first manual compile.

Print and HTML versions of each document may coexist, each with its own set of auxiliary files. Support files are self-generated on request. Assistance is provided for import into EPUB conversion software and word processors.

A modular package-loading system uses the lwarp version of a package for HTML when available. Several dozen LaTeX packages are supported with these high-level source compatibility replacements.

A tutorial is provided to quickly introduce the user to the major components of the package.

To update existing projects, see section 1, Updates.

Note that this is still a "beta" version of lwarp, and some things may change in response to user feedback and further project development.

Contents

List of Figures

List of Tables

Package 1

lwarp.sty

1 Updates

The following is intended for those updating existing projects which use lwarp, highlighting any special changes which must be made due to improvements or modifications in lwarp itself.

For a detailed list of changes, see the Change History on page 477.

v0.31:

- Improved compatibility with keyfloat, including the new `keywrap` environment.

v0.30:

⚠ lwarp-newproject

- lwarp-newproject has been removed, and its functions have been combined with lwarp.

  To modify existing documents, remove from the document source:

  ```
  \usepackage{lwarp-newproject}
  ```

  The lwarp package now produces the configuration files during print output, and also accepts the option `lwarpmk` if desired.

⚠ HTML setup changes.

- A number of macros related to HTML settings have been converted to options, and other macros and options have been renamed to create a consistent syntax:

Old Macro	New Package Option
`\HomeHTMLFileName`	`HomeHTMLFilename`
`\HTMLFileName`	`HTMLFilename`
`\useLatexmk`	`latexmk`
`\warpOSwindows`	`OSWindows`

Old Package Option	New Package Option
`lwarpmklang`	`IndexLanguage`
(new)	`xdyFilename`

Old Macro	New Macro
`\MetaLanguage`	`\HTMLLanguage`
`\HTMLauthor`	`\HTMLAuthor`
`\NewHTMLdescription`	`\HTMLDescription`
`\SetFirstPageTop`	`\HTMLFirstPageTop`
`\SetPageTop`	`\HTMLPageTop`
`\SetPageBottom`	`\HTMLPageBottom`
`\NewCSS`	`\CSSFilename`

- Per the above changes, in existing documents, modify the package load of lwarp, such as:

```
\usepackage[
    HomeHTMLFilename=index,
    HTMLFilename={},
    IndexLanguage=english
]{lwarp}
```

- The file `lwarp_html.xdy` has been renamed `lwarp.xdy`. To update each document's project:
  1. Make the changes shown above.
  2. Recompile the document in print mode. This updates the project's configuration files, and also generates the new file `lwarp.xdy`.
  3. The old file `lwarp_html.xdy` may be deleted.
- The new lwarp package option `xdyFilename` may be used to tell `lwarpmk` to use a custom `.xdy` file instead of `lwarp.xdy`. See section 6.11.
- Improvements in index processing:
  - `xindy`'s language is now used for index processing as well as glossary.
  - Print mode without `latexmk` now uses `xindy` instead of `makeindex`.
  - `texindy`/`xindy` usage depends on `pdflatex` vs `xelatex`, `lualatex`.
  - For `pdflatex` and `texindy`, the `-C utf8` option is used. This is supported in modern distributions, but a customized `lwarpmk.lua` may need to be created for use with older distributions.

v0.29:

- Add: `lwarpmklang` option for lwarp-newproject and lwarp. Sets the language to use while processing the glossary. (As of v0.30, this has been changed to the `IndexLanguage` option.)
- Fix: `\includegraphics` when no optional arguments.

v0.28:

- `\HTMLAuthor {`⟨*name*⟩`}` assigns HTML meta author if non-empty. Defaults to `\theauthor`.
- Booean `HTMLDebugComments` controls whether HTML comments are added for closing `<div>`s, opening and closing sections, etc.
- Boolean `FormatEPUB` changes HTML output for easy EPUB conversion via an external program. Removes per-file headers, footers, and nav. Adds footnotes per chapter/section.
- Boolean `FormatWordProcessor` changes HTML output for easier conversion by a word processor. Removes headers and nav, prints footnotes per section, and also forces single-file output and turns off HTML debug comments.
- Boolean `HTMLMarkFloats` adds text marks around floats only if `FormatWordProcessor`. These make it easier to identify float boundaries, which are to be manually converted to word-processor frames.
- Updated for the new MathJax CDN repository.
- Adds tabulary.
- Supports the options syntax for graphics.
- Improved index references, now pointing exactly to their target.
- Adds glossaries. `lwarpmk` is modified to add `printglosssary` and `htmlglossary` actions.

v0.27:

- Improved documentation for MacOS install.
- Fix for microtype with XeLaTeX and LuaLaTeX.
- Fix for table footnote paragraph tags.
- Adds lettrine, ulem, and soul.

v0.26:

- Improved installation instructions for MiKTeX regarding generating the `lwarpmk` executable.
- Footnotes are now supported by LaTeX boxes instead of pagenotes. pagenote now works as per the print version. footnote, footnotehyper, footmisc, endnotes, marginnote, and sidenotes are also supported.

- LaTeX labels now are used to track the page numbers of `latexiamges`. This allows the correct inclusion of `lateximages` in footnotes, pagenotes, and endnotes.
- cutwin and floatflt are also supported.

v0.25:

- Fix: Allows graphicx and graphicsx before lwarp because XeLaTeX and LuaLaTeX use xunicode which uses graphics.
- Package support for framed, several theorem packages, and ellipses.

v0.24:

- tikz's `babel` library is load automatically as needed.
- subfig has been added, along with `lofdepth` and `lotdepth`.
- `picture` and `tikzpicture` now may be inline.

v0.22:

- Support has been added for tabular column types `D`, `!`, and `X`. Unknown column types are converted to `l`.
- Additional packages are supported, including abstract, dcolumn, tabularx, and varioref.

v0.21:

- Documentation for installing on Windows has been updated and improved.
- For Windows compatibility, the `lateximages` shell script has been replaced with a `lateximages.txt` file, which is parsed by `lwarpmk` to generate lateximages. This does not require any changes in the user's code.
- Windows `lwarpmk again` now functions.
- For improved error handling, lwarp now verifies the order in which packages are loaded, and signals an error for misplaced packages. inputenc, fontenc, newunicode, and fontspec must be loaded before lwarp, and the other packages which lwarp knows about must be loaded after.
- lwarp no longer requires a `\title` be assigned.

v0.20:

- The `makefile` and related infrastructure has been replaced by the `lwarpmk` utility. This provides increased portability, reduced dependencies, and much simpler installation and setup.
- The `lwarp-newproject` package is now used to locally create support files.

- The print and HTML versions of a document may co-exist with their own sets of auxiliary files.
- Package handling is now controlled by a modular system which looks for and loads an `lwarp-<package>` version if available.
- High-level source compatibility is provided for all supported packages, almost totally eliminating the need for `warpprint` and `warpHTML` environments.
- A large number of additional packages are supported.
- A new tutorial is included in the documentation, and many obsolete sections have been removed.
- `\NewHTMLdescription` sets the HTML meta description tag for each file. See section 6.8. (v0.30 changes this to `\HTMLDescription`.)
- ⚠ `\HTMLFilename` may now be empty, allowing filenames without a prefix. Lwarp no longer automatically appends a `-` character. For existing projects, add a `-` to the end of `\HTMLFilename`.
- ⚠ `\HomeHTMLFilename` and `\HTMLFilename` no longer use escaped underscore characters. Underscores may be used in filenames as-is. (Version 0.30 changes these to package options `HomeHTMLFilename` and `HTMLFilename`.)
- lwarp now tries to auto-detect the operating system, and `\warpOSwindows` is only needed if the auto-detection fails to detect Windows. (As of v0.30, `\warpOSwindows` has been converted to the `OSWindows` option.)
- Tabular column types `@`, `>`, and `<` are now supported.
- `BlockClass` and `\InlineClass` add an optional style.
- The `sidebar` and `example` environments have been moved to the test suite, and are no longer included in lwarp.

v0.19:

- ⚠ MathJax now may be used to display math via the `mathjax` option. See sections 6.2 and 6.12.5. To use MathJax with a pre-existing project, copy or link the file `lwarp_mathjax.txt` to the project's directory.
- `\rule` added, supporting width, height, raise, `\textcolor`.
- `\LateximageFontSizeName` provides user-adjustable font size for math and `lateximages`.
- `\minipagefullwidth` requests that the next minipage be full-width in HTML, but still the assigned width in print.
- `minipage` improved side-by-side rendering.
- CSS class `tablenotes` is provided for table note items.
- ⚠ `\warpprintonly` replaces `\rowprintedonly`, and `\warpHTMLonly` is added. These behave like the `warpprint` and `warpHTML` environments, and are generally useful, so they replace the previously table-specific syntax.

- cleveref is loaded `\AtEndPreamble` for improved reliability. See section 65.
- `\xfracHTMLfontsize` controls xfrac font size in HTML.
- Ti*k*z improved catcode handling.

v0.18:

- The verse package and the verse-related commands from the memoir package are now supported.
- Responsive web design has been improved for the sideTOC.
- `\includegraphics` now maintains relative sized for em, ex, and %.

v0.17:

- mdframed package is supported.

v0.16:

- Font and input encoding are now controlled by the user, and lwarp is loaded after fonts have been selected.
- Support for XƎLaTeX and LuaLaTeX. See section 6.1.

2 Introduction

The lwarp project aims to allow a rich LaTeX document to be converted to a reasonable HTML interpretation. No attempt has been made to force LaTeX to provide for every HTML-related possibility, and HTML cannot exactly render every possible LaTeX concept. Where compromise is necessary, it is desirable to allow the print output to remain typographically rich, and compromise only in the HTML conversion.

Several "modern" features of HTML5, CSS3, and SVG are employed to allow a fairly feature-rich document without relying on the use of Javascript. Limited testing on older browsers show that these new features degrade gracefully, although the SVG format for math may not be available on small cell phones.

`pdflatex`, `xelatex`, or `lualatex` is used, allowing lwarp to process the usual image formats. While generating HTML output, SVG files are used in place of PDF. Other formats such as JPG are used as-is.

SVG images may be used for math, and are also used for `picture`, and Ti*k*z environments, as this format has better browser and e-book support than MathML (as of this writing), while still allowing for the high-quality display and printing of images (again, subject to potentially bug-ridden[1] browser support).

Furthermore, SVG images allow math to be presented with the same precise formatting as in the print version. Math is accompanied by ALT tags holding the LaTeX source for the expression, allowing it to be copy/pasted into other documents.[2] Custom LaTeX macros may be used as-is in math expressions, since the math is evaluated entirely inside LaTeX.

The MathJax JavaScript display engine may be selected for math display instead of using SVG images. Subject to browser support and Internet access, MathJax allows an HTML page to display math without relying on a large number of external image files, one per math expression. lwarp maintains LaTeX control for cross-referencing and equation numbering / formatting.

The lwarp package allows LaTeX to directly generate HTML5 tags from a LaTeX source document, with only minor intervention on the user's part. A `texlua` program called `lwarpmk` is used to process either the print or HTML version of the document. A few external utility programs are used to finish the conversion from

[1] Firefox has had an on-again/off-again bug for quite some time regareding printing SVGs at high resolution.

[2] There seems to be some debate as to whether MathML is actually an improvement over LaTeX for sharing math. The author has no particular opinion on the matter, except to say that in this case LaTeX is much easier to implement!

a LaTeX-generated PDF file which happens to have HTML5 tags, to a number of HTML5 plain-text files and accompanying images.

lwarp automatically generates the extra files necessary for the HTML conversion, such as CSS and `.xdy` files, and configuration files for the utility `lwarpmk`. Also included is a parallel version of the user's source document, `<sourcename>-html.tex`, which selects HTML output and then inputs the user's own source. This process allows both the printed and HTML versions to co-exist side-by-side, each with their own auxiliary files.

When requesting packages during HTML conversion, `lwarp` first looks to see if it has its own modified version to use instead of the usual LaTeX version. These `lwarp-packagename.sty` files contain code used to emulate or replace functions for HTML output.

Enough functionality is provided to convert a typical article containing technical content. Not every package has been tested, but many of the most useful ones are known to work, either as-is or through emulation with replacement code. (See table 1 on page 27.)

Assistance is provide for modifying the HTML output to suite the creation of EPUB documents, and for modifying the HTML output to ease import into a word processor.

2.1 Supported packages and features

Supported classes include book, report, and article. memoir is planned, but in the meantime many of the packages used by memoir are already supported.

Table 1 lists some of the various LaTeX features which may be used. *Supported* means that the package or macro may be used as-is, perhaps with minor limitations. *Emulated* means that the original package or macro is not used, but similar functionality is provided in a way which is intended to be compatible with the user's LaTeX code.

Table 1: LaTeX–HTML generation — lwarp package — Supported functions

Category	Status
Engines:	pdfLaTeX, XeLaTeX, LuaLaTeX
Classes:	book, report, or article. memoir is planned.

lwarp Supported Functions — continued

Category	Status
Sectioning:	Supported, with hyperlinks. Honors `tocdepth` and `secnumdepth`. Adds `filedepth` for splitting the HTML output. Files may be numbered sequentially or named according to section name. Common short words and punctuation is removed from the filenames.
Table of Contents, Figures, Tables:	Supported, with hyperlinks.
Title page:	`\maketitle`, `titlepage`, titling. Optional titling-based commands for published and subtitle.
`abstract`:	Supported
Cross-references:	Emulated, with hyperlinks.
hyperref:	Emulated. HTML hyperlinks are generated for TOC, LOF, LOT, `\nameref`, `\ref`, the cleveref commands, and index entries.
Footnotes:	footnote, footmisc, marginnote, sidenote, pagenote, endnotes.
Indexing:	`texindy` is used, with hyperlinks.
Glossary:	glossaries and `xindy` are used.
Bibliography:	Supported, without hyperlinks so far.
Math:	Supported. Converted to SVG images with HTML `ALT` tags containing the LaTeX source for the math expression. MathJax supported as an alternative. $\mathcal{AMS}$ environments are supported. User-defined macros are available during converson, due to native LaTeX processing.
Theorems:	Support for native LaTeX theorems, plus theorem, amsthm, ntheorem.
Floats:	Appear where declared. float, newfloat, caption and subcaption, subfig, capt-of, placeins, trivfloat, floatrow, keyfloat, wrapfig, cutwin, floatflt.

lwarp Supported Functions — continued

Category	Status
tabular:	Emulated. \multirow and \multicolumn are available, but cannot be used at the same time. Nested tables are not supported.
array:	Supported inside math environments, emulated elsewhere.
tabularx, tabulary, threeparttable, multirow:	Emulated.
longtable:	Emulated. Converted to a tabular. Captions supported. Extra headings and \kill lines must be enclosed in \warpprintonly{}
booktabs:	Emulated. \toprule and \bottomrule form black rules, \midrule forms silver rules, as demonstrated on this table. \cmidrule, demonstrated at this line, does not use width or trim options.
graphics, graphicx:	Emulated. \includegraphics supports width, height, origin, angle, and scale tags, and adds class. References to PDF files are changed to SVG, other image types are accepted as well. \rotatebox and \scalebox are supported as well as HTML can handle.
rotating:	Emulated. All objects are displayed unrotated.
Lists:	Supported
enumitem:	Supported, although spacing is still controlled by CSS.
Environments:	Standard LaTeX environments are supported.
picture and tikz:	Converted to an SVG image.
minipage:	Supported with some HTML5-imposed limitations. Nested minipages are supported. Footnotes appear at the bottom of the HTML page.
fancyvrb:	Supported except for verbatim footnotes.
framed, mdframed:	Supported

lwarp Supported Functions — continued

Category	Status
multicol:	Emulated, with CSS3. Converted to up to three columns with an optional heading, per browser support. Single-column if unsupported.
siunitx:	Supported except for `per-mode=fraction`.
xfrac:	Supported
Direct formatting:	`\emph`, `\textsuperscript`, `\textbf`, etc are supported. `\bfseries`, etc. are not yet supported. lettrine, ulem, and soul are supported.
Ordinals:	nth, fmtcount, and engord are supported.
Text ligatures:	Ligatures for symbols are supported. Ligatures for f, q, t are intentionally turned off because many simpler browsers do not display them correctly. Modern full-featured browsers re-create these ligatures on-the-fly.
Horizontal space:	HTML output for thin-unbreakable, unbreakable, `\enskip`, `\quad`, `\qquad`, `\hspace`.
Rules:	`\rule` with width, height, raise, text color.
HTML reserved characters:	`\&`, `\textless`, and `\textgreater` are converted to HTML entities.
xcolor:	Supported. Full package color names, any color models, and mixing is converted to hex web colors via `\convertcolorspec`. Patched commands are `\textcolor`, `\colorbox`, and `\fcolorbox`. `\pagecolor` is not supported.
Where:	
Supported:	The existing LaTeX package is used.
Emulated:	The LaTeX package is not used, but some/all of its functions are emulated. Null functions, lengths, and counters are provided for source compatibility.

Supported packages include everything listed in the table of contents, plus each of the following in table 2, and probably others which have not yet been tested. Many

Table 2: Additional supported packages

babel, bm, calc, cleveref, csquotes, enumitem, fancyvrb, fileerr, newtxmath, siunitx, somedefs, tikz, trace, varioref, xspace

are simply nullfied during HTML output. Others are not affected by the output mode and thus work as-is.

These packages and features probably works with little or no change to the user's source code. Special environments are provided to mark blocks of code which are for print only, HTML only, or both, should it be necessary.

3 Alternatives

Summarized below are several other ways to convert a LaTeX or other document to HTML. Where an existing LaTeX document is to be converted to HTML, lwarp may be a good choice. For new projects with a large number of documents, it may be worth investigating the alternatives before decided which path to take.

3.1 Internet class

Cls `internet` The closest to lwarp in design principle is the `internet` class by Andrew Stacey (https://github.com/loopspace/latex-to-internet), an interesting project which directly produces several versions of markdown, and also HTML and EPUB.

3.2 TeX4ht

Prog `TeX4ht` http://tug.org/tex4ht/

This system uses native LaTeX processing to produce a DVI file containing special commands, and then uses additional post-processing for the HTML conversion by way of numerous configuration files. In some cases, lwarp provides a better HTML conversion, and it supports a different set of packages. TeX4ht produces several other forms of output beyond HTML.

3.3 Translators

These systems use external programs to translate a subset of LaTeX syntax into HTML. Search for each on CTAN (http:\ctan.org).

Prog `Hevea` **HEVEA:** http://hevea.inria.fr/ (not on CTAN)

Prog `TtH` **TTH:** http://hutchinson.belmont.ma.us/tth/

Prog `GELLMU` **GELLMU:** http://www.albany.edu/~hammond/gellmu/

Prog `LaTeXML` **LaTeXML:** http://dlmf.nist.gov/LaTeXML/

Prog `Plastex` **PlasTeX:** https://github.com/tiarno/plastex

Prog `LaTeX2HTML` **LaTeX2HTML:** http://www.latex2html.org/ and http://ctan.org/pkg/latex2html.

Prog `TeX2page` **TeX2page:** http://ds26gte.github.io/tex2page/index.html

Finally, GladTEX may used to directly insert LATEX math into HTML:

Prog GladTeX **GladTEX:** http://humenda.github.io/GladTeX/

3.4 AsciiDoc

AsciiDoc is one of the most capable markup languages, providing enough features to produce the typical technical-writing document with cross-references, and it writes LATEX and HTML.

Prog AsciiDoc **Asciidoctor:** http://asciidoctor.org/ (More active.)

Prog AsciiDoctor **AsciiDoc:** http://asciidoc.org/ (The original version.)

The Asciidoctor-LaTeX project is adding additional LATEX-related features.

Asciidoctor-LateX:
http://www.noteshare.io/book/asciidoctor-latex-manual
Prog Asciidoctor-LaTeX https://github.com/asciidoctor/asciidoctor-latex

3.5 Pandoc

Prog Pandoc

A markup system which also reads and writes LATEX and HTML.

Pandoc: http://pandoc.org/

(Watch for improvements in cross-references to figures and tables.)

3.6 Word processors

Prog Word
Prog LibreOffice
Prog OpenOffice

It should be noted that the popular word processors have advanced through the years in their abilities to represent math with a LATEX-ish input syntax, unicode math fonts, and high-quality output, and also generate HTML with varying success. See recent developments in Microsoft ® Word ® and LibreOffice ™ Writer.

3.7 Commercial systems

Prog Adobe
Prog FrameMaker
Prog InDesign
Prog Flare
Prog Madcap

Likewise, several professional systems exist whose abilities have been advancing in the areas of typesetting, cross-referencing, and HTML generation. See Adobe® FrameMaker®, Adobe® InDesign®, and Madcap Flare™.

3.8 Comparisons

AsciiDoc, Pandoc, and various other markup languages typically have a syntax which tries to be natural and human-readable, but the use of advanced features tends to require many combinations of special characters, resulting in a complicated mess of syntax. By contrast, LaTeX spells things out in readable words but takes longer to type, although integrated editors exist which can provide faster entry and a graphic user interface. For those functions which are covered by the typical markup language it is arguable that LaTeX is comparably easy to learn, while LaTeX provides many more advanced features where needed, along with a large number of pre-existing packages which provide solutions to numerous common tasks.

Text-based document-markup systems share some of the advantages of LaTeX vs. a typical word processor. Documents formats are stable. The documents themselves are portable, work well with revision control, do not crash or become corrupted, and are easily generated under program control. Formatting commands are visible, cross-referencing is automatic, and editing is responsive. Search/replace with regular expressions provides a powerful tool for the manipulation of both document contents and structure. Markup systems and some commercial systems allow printed output through a LaTeX back end, yielding high-quality results especially when the LaTeX template is adjusted, but they lose the ability to use LaTeX macros and other LaTeX source-document features.

The effort required to customize the output of each markup system varies. For print output, LaTeX configuration files are usually used. For HTML output, a CSS file will be available, but additional configuration may require editing some form of control file with a different syntax, such as XML. In the case of lwarp, CSS is used, and much HTML output is adjusted through the usual LaTeX optional macro parameters, but further customization may require patching LaTeX code.

The popular word processors and professional document systems each has a large base of after-market support including pre-designed styles and templates, and often include content-management systems for topic reuse.

4 Installation

Table 3 shows the tools which are used for the LaTeX to HTML conversion. In most cases, these will be available via the standard package-installation tools.

4.1 Installing the lwarp package

There are several ways to install lwarp. These are listed here with the preferred methods listed first:

Pre-installed: Try entering into a command line:

Enter ⇒ `kpsewhich lwarp.sty`

If a path to `lwarp.sty` is shown, then lwarp is already installed.

TeX Live: If using a TeX Live distribution, try installing via `tlmgr`:

Enter ⇒ `tlmgr install lwarp`

MiKTeX: If using MiKTeX, try using the package installer to install the package `lwarp`. Also update the package `miktex-misc`, which will install the `lwarpmk` executable.

Operating-system package: The operating-system package manager may already have `lwarp`, perhaps as part of a set of TeX-related packages.

CTAN TDS archive: lwarp may be downloaded from the Comprehensive TeX Archive:

1. See http://ctan.org/pkg/lwarp for the lwarp package.
2. Download the TDS archive: `lwarp.tds.zip`
3. Find the TeX `local` directory:

   TeX Live:

   Enter ⇒ `kpsewhich -var-value TEXMFLOCAL`

   MiKTeX:
   In the "Settings" window, "Roots" tab, look for a local TDS root.

   This should be something like:

   `/usr/local/texlive/texmf-local/`

4. Unpack the archive in the TDS local directory.

Table 3: Required software programs

Provided by your LaTeX distribution:

From TeXLive: http://tug.org/texlive/.

LaTeX: `pdflatex`, `xelatex`, or `lualatex`.

The lwarp package: This package.

The `lwarpmk` utility: Provided along with this package. This should be an operating-system executable in the same way that `pdflatex` or `latexmk` is. It is possible to have the lwarp package generate a local copy of `lwarpmk` called `lwarpmk.lua`. See table 4.

luatex: Used by the `lwarpmk` program to simplify and automate document generation.

xindy: The xindy package is used by lwarp to create indexes. On a MiKTeX system this may have to be acquired separately, but it is part of the regular installer as of mid 2015.

latexmk: Optionally used by `lwarpmk` to compile LaTeX code. On a MiKTeX system, Perl may need to be installed first.

pdfcrop: Used to pull images out of the LaTeX PDF.

Poppler PDF utilities:

pdftotext: Used to convert PDF to text.

pdfseparate: Used to pull images out of the LaTeX PDF.

pdftocairo: Used to convert images to SVG.

These might be provided by your operating-system package manager.

From Poppler: poppler.freedesktop.org.

For MacOS®, see https://brew.sh/, install Homebrew, then

Enter ⇒ `brew install poppler`

For Windows, see:
https://sourceforge.net/projects/poppler-win32/ and:
http://blog.alivate.com.au/poppler-windows/

Perl:

This may be provided by your operating-system package manager, and is required for some of the Poppler PDF utilities.

perl.org, strawberryperl.com

Automatically downloaded from the internet as required:

MathJax: Optionally used to display math. Automatically loaded from the MathJax website when needed.
From: mathjax.org

5. Renew the cache:

   Enter ⇒ `mktexlsr`

   — or —

   Enter ⇒ `texhash`

   Or, for Windows MiKTEX, start the program called `MiKTeX Settings (Admin)` and click on the button called Refresh FNDB.

CTAN `.dtx` and `.ins` files: Another form of TEX package is the `.dtx` and `.ins` source files. These files are used to create the documentation and `.sty` files.

1. See http://ctan.org/pkg/lwarp for the lwarp package.
2. Download the zip archive `lwarp.zip` into your own `lwarp` directory.
3. Unpack `lwarp.zip`.
4. Locate the contents `lwarp.dtx` and `lwarp.ins`
5. Create the documentation:

   Enter ⇒ `pdflatex lwarp.dtx`

   (several times)
6. Create the `.sty` files:

   Enter ⇒ `pdflatex lwarp.ins`
7. Copy the `.sty` files somewhere such as the TEX Live local tree found in the previous CTAN TDS section, under the subdirectory:

   `<texlocal>/tex/latex/local/lwarp`
8. Copy the documentation `lwarp.pdf` to a `source` directory in the local tree, such as:

   `<texlocal>/doc/local/lwarp`
9. Renew the cache:

   Enter ⇒ `mktexlsr`

   — or —

   Enter ⇒ `texhash`

   Or, for Windows MiKTEX, start the program called `MiKTeX Settings (Admin)` and click on the button called Refresh FNDB.
10. See section 4.2.1 to generate your local copy of `lwarpmk`.
11. Once the local version of `lwarpmk.lua` is installed, it may be made available system-wide as per section 4.2.

Project-local ctan `.dtx` and `.ins` files: The `.dtx` and `.ins` files may be downloaded to a project directory, then compiled right there, alongside the document source files. The resultant `*.sty` and `lwarpmk.lua` files may be used as-is, so long as they are in the same directory as the document source. This approach is especially useful if you would like to temporarily test lwarp before deciding whether to permanently install it.

Just testing!

4.2 Installing the `lwarpmk` utility

(Note: If `lwarpmk` is not already installed, it is easiest to use a local copy instead of installing it system-wide. See section 4.2.1.)

After the lwarp package is installed, you may need to setup the `lwarpmk` utility:

1. At a command line, try executing `lwarpmk`. If the `lwarpmk` help message appears, then `lwarpmk` is already set up. If not, it is easiest to generate and use a local copy. See section 4.2.1.
2. For MiKTEX, try updating the `miktex-misc` package. This may install the `lwarpmk` executable for you.

   Otherwise, continue with the following:
3. Locate the file `lwarpmk.lua`, which should be in the `scripts` directory of the TDS tree. On a TEX Live or MiKTEX system you may use

   Enter ⇒ `kpsewhich lwarpmk.lua`

   (If the file is not found, you may also generate a local copy and use it instead. See section 4.2.1.)
4. Create `lwarpmk`:

   Unix: Create a symbolic link and make it executable:

   (a) Locate the TEX Live binaries:

   Enter ⇒ `kpsewhich -var-value TEXMFROOT`

   This will be something like:

   `/usr/local/texlive/<year>`

   The binaries are then located in the `bin/<arch>` directory under the root:

   `/usr/local/texlive/<year>/bin/<architecture>/`

   In this directory you will find programs such as `pdflatex` and `makeindex`.

   (b) In the binaries directory, create a new symbolic link from the binaries directory to `lwarpmk.lua`:

   Enter ⇒ `ln -s <pathtolwarpmk.lua> lwarpmk`

(c) Make the link executable:

Enter ⇒ `chmod 0755 lwarpmk`

Windows TEX Live: Create a new `lwarpmk.exe` file:

(a) Locate the TEX Live binaries as shown above for Unix.

(b) In the binaries directory, make a *copy* of `runscript.exe` and call it `lwarpmk.exe` This will call the copy of `lwarpmk.lua` which is in the `scripts` directory of the distribution.

Windows MiKTEX: Create a new `lwarpmk.bat` file:

(a) Locate the binaries. These will be in a directory such as:

```
C:\Program Files\MiKTeX 2.9\miktex\bin\x64
```

In this directory you will find programs such as `pdflatex.exe` and `makeindex.exe`.

(b) Create a new file named `lwarpmk.bat` containing:

```
texlua "C:\Program Files\MiKTeX 2.9\scripts\lwarp\lwarp.texlua" %*
```

This will call the copy of `lwarpmk.lua` which is in the `scripts` directory of the distribution.

4.2.1 Using a local copy of lwarpmk

It is also possible to use a local version of `lwarpmk`:

1. When compiling the tutorial in section 5, use the `lwarpmk` option for the lwarp package:

   ```
   \usepackage[lwarpmk]{lwarp}
   ```

2. When the tutorial is compiled with `pdflatex`, the file `lwarpmk.lua` will be generated along with the other configuration files.

3. `lwarpmk.lua` may be used for this project:

   Unix:

   (a) Make `lwarpmk.lua` executable:

   Enter ⇒ `chmod 0755 lwarpmk.lua`

   (b) Compile documents with

   Enter ⇒ `./lwarpmk.lua html`

   Enter ⇒ `./lwarpmk.lua print`

   etc.

   (c) It may be useful to rename or link to a version without the `.lua` suffix.

Windows:

Compile documents with either of the following, depending on which command shell is being used:

Enter ⇒ `texlua lwarpmk.lua html`

Enter ⇒ `texlua lwarpmk.lua print`

etc.

Or:

Enter ⇒ `lwarpmk html`

Enter ⇒ `lwarpmk print`

etc.

4.3 Installing additional utilities

To test for the existence of the additional utilities:

Enter the following in a command line. If each programs' version is displayed, then that utility is already installed. See table 3 on page 36.

Enter ⇒ `luatex -version`

Enter ⇒ `xindy -version`

Enter ⇒ `latexmk -version`

Enter ⇒ `perl -version`

Enter ⇒ `pdfcrop -version`

Enter ⇒ `pdftotext -v`

Enter ⇒ `pdfseparate -version`

Enter ⇒ `pdftocairo -v`

To install `xindy`, `latexmk`, and `pdfcrop`:

The TEX utilities `xindy`, `latexmk`, and `pdfcrop` may be provided by your operating system's package manager, and are also provided by the CTAN archive:

http://ctan.org/pkg/xindy
http://ctan.org/pkg/latexmk
http://ctan.org/pkg/pdfcrop

Prog `pdftotext`
Prog `pdfseparate`
Prog `pdftocairo`

To install the Poppler utilities to a Unix/Linux system:

The tools from the POPPLER project should be provided by your operating system's package manager.

To install the Poppler utilities to a MacOS machine:

1. Install Homebrew from https://brew.sh/:
   Enter ⇒
```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```
2. Install the Poppler utilities:
   Enter ⇒ `brew install poppler`

To install the Poppler utilities to a Windows machine:

1. See table 3 on page 36.
2. Download and extract the Poppler utilities `pdftotext`, `pdfseparate`, and `pdftocairo` to a directory, such as `Poppler`.
3. In the Start window, type "Path" to search for results related to Path. Or, open the control panel and search for "Path".
4. Choose "Edit the system environment variables" in the control panel.
5. Choose the "Environment Variables" button.
6. Choose the "Path" variable, then the "Edit" button.
7. Choose the "New" button to make an additional entry.
8. Enter the `bin` directory of the Poppler utilities, such as:
   `C:\Users\<myname>\Desktop\Poppler\poppler-0.5_x86\poppler-0.5\bin`

   Be sure to include `\bin`.
9. Click "Ok" when done.

Prog `perl`

To install Perl to a Windows machine:

1. Download and install a version of Perl, such as Straweberry Perl, to a directory without a space in its name, such as `C:\Strawberry`.
2. Edit the Path as seen above for the Poppler utilities.

3. Enter the `bin` directory of the Perl utility, such as:

   `C:\Strawberry\perl\bin`

   Be sure to include `\bin`.
4. Click "Ok" when done.

Any utilities installed by hand must be added to the `PATH`.

5 Tutorial

This section shows an example of how to create an lwarp document.

5.1 Starting a new project

1. Create a new project directory called `tutorial`.

File `tutorial.tex`

2. Inside the `tutorial` directory, create a new file called `tutorial.tex`. This may be done several ways:

   Copy from the documentation PDF:
   A listing is in fig. 1, which may be copied/pasted from the figure directly into your own editor, depending on the quality of the PDF viewer and editor, or:

   Copy from the lwarp documentation directory:

File `lwarp_tutorial.txt`

⚠ Note: `.txt` suffix!

   Another copy may be found by entering into a command line:

   Enter ⇒ `texdoc -l lwarp_tutorial.txt`

   This should be in the `doc/latex/lwarp/` directory along with this PDF documentation. Copy `lwarp_tutorial.txt` directly into your `tutorial` directory, renamed as `tutorial.tex`.

⚠ Bad formatting!

   When using Windows, use an editor other than Notepad, since Notepad does not accept the end-of-line from a Unix text file.

3. Compile the project:

   Enter ⇒ `pdflatex tutorial.tex`

   (several times)

   (`xelatex` or `lualatex` may be used as well.)

4. View the resulting `tutorial.pdf` with a PDF viewer.

A number of new files are created when `tutorial.tex` is compiled, as shown in table 4. These files are created by the `lwarp` package.

(Two of the new files are configuration files for the helper program `lwarpmk`. Whenever a print version of the document is created, the configuration files for `lwarpmk` are updated to record the operating system, LaTeX program (`pdflatex`, `xelatex`, or `lualatex`), the filenames of the source code and HTML output, and whether the additional helper program `latexmk` will be used to compile the document.)

Figure 1: tutorial.tex listing

Note: There are two pages!

```
% Save this as tutorial.tex for the lwarp package tutorial.

\documentclass{book}

\usepackage{iftex}

% --- LOAD FONT SELECTION AND ENCODING BEFORE LOADING LWARP ---

\ifPDFTeX
\usepackage{lmodern}            % pdflatex
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\else
\usepackage{fontspec}           % XeLaTeX or LuaLaTeX
\fi

% --- LWARP IS LOADED NEXT ---
\usepackage[
%   HomeHTMLFilename=index,     % Filename of the homepage.
%   HTMLFilename={node-},       % Filename prefix of other pages.
%   IndexLanguage=english,      % Language for xindy index, glossary.
%   latexmk,                    % Use latexmk to compile.
%   OSWindows,                  % Force Windows. (Usually automatic.)
%   mathjax,                    % Use MathJax to display math.
]{lwarp}
% \boolfalse{FileSectionNames}  % If false, numbers the files.

% --- OTHER PACKAGES ARE LOADED AFTER LWARP ---
\usepackage{makeidx} \makeindex
\usepackage{xcolor}             % (Demonstration purposes only.)
\usepackage{hyperref,cleveref}  % LOAD THESE LAST!

% --- LATEX AND HTML CUSTOMIZATION ---
\title{The Lwarp Tutorial}
\author{Some Author}
\setcounter{tocdepth}{2}        % Include subsections in the \TOC.
\setcounter{secnumdepth}{2}     % Number down to subsections.
\setcounter{FileDepth}{1}       % Split \HTML\ files at sections
\booltrue{CombineHigherDepths}  % Combine parts/chapters/sections
\setcounter{SideTOCDepth}{1}    % Include subsections in the side\TOC
\HTMLAuthor{Some Author}        % Sets the HTML meta author tag.
\HTMLLanguage{en-US}            % Sets the HTML meta language.
\HTMLDescription{A description.}% Sets the HTML meta description.
\HTMLFirstPageTop{Name and \fbox{HOMEPAGE LOGO}}
```

```
\HTMLPageTop{\fbox{LOGO}}
\HTMLPageBottom{Contact Information and Copyright}
\CSSFilename{lwarp_sagebrush.css}

\begin{document}

\maketitle                      % Or titlepage/titlingpage environment.

% An article abstract would go here.

\tableofcontents                % MUST BE BEFORE THE FIRST SECTION BREAK!
\listoffigures

\chapter{First chapter}

\section{A section}

This is some text which is indexed.\index{Some text.}

\subsection{A subsection}

See \cref{fig:withtext}.

\begin{figure}\begin{center}
\fbox{\textcolor{blue!50!green}{Text in a figure.}}
\caption{A figure with text\label{fig:withtext}}
\end{center}\end{figure}

\section{Some math}

Inline math: $r = r_0 + vt - \frac{1}{2}at^2$
followed by display math:
\begin{equation}
a^2 + b^2 = c^2
\end{equation}

\printindex

\end{document}
```

Table 4: Files created along with the print version

`tutorial.pdf`: The PDF output from LaTeX. The print version of the document.

`tutorial_html.tex`: A small `.tex` file used to create a parallel HTML version of the document, which co-exists with usual the PDF version, and which will have its own auxiliary files. In this way, both PDF and HTML documents may co-exist side-by-side.

Auxiliary files: The usual LaTeX files `.aux`, `.log`, `.out`, `.toc`, `.lof`, `.idx`. When an HTML version of the document is created, `_html` versions of the auxiliary files will also be generated.

`lwarpmk.conf`: A configuration file for `lwarpmk`, which is used to automate the compilation of PDF or HTML versions of the document.

`tutorial.lwarpmkconf`: Another configuration file used by `lwarpmk`, which is only useful if you wish to have several projects residing in the same directory.

`.css` files: `lwarp.css`, `lwarp_formal.css`, `lwarp_sagebrush.css` These files are standard for lwarp, and are not meant to be modified by the user.

`sample_project.css`: An example of a user-customized CSS file, which may be used for project-specific changes to the lwarp defaults.

`lwarp.xdy`: Used by lwarp while creating an index. This file should not be modified by the user. A custom file may be used instead, if necessary.

`lwarp_mathjax.txt`: Inserted into the HTML files when `MathJax` is used to display math. This file should not be modified by the user.

`comment.cut`: A temporary file used by lwarp to conditionally process blocks of text. This file may be ignored.

When the `lwarpmk` option is given to the lwarp package:

`lwarpmk.lua`: A local copy of the `lwarpmk` utility.

On Unix-related operating systems this file must be made executable:

```
chmod u+x lwarpmk.lua
```

This may be useful to have to archive with a project for future use.

5.2 Compiling the print version with `lwarpmk`

The `lwarpmk` utility program is used to compile either the printed or the HTML version of the document.

`lwarpmk print` is used to recompile a printed version of the document.

1. Re-compile the print version:

   Enter ⇒ `lwarpmk print`

   `lwarpmk` prints an introduction then checks to see if the document must be recompiled. If it seems that the files are up-to-date, then `lwarpmk` informs you of that fact and then exits.

2. Make a small change in the original document, such as adding a space character.

3. Recompile again.

   Enter ⇒ `lwarpmk print`

   The document is recompiled when a change is seen in the source. Several compilations may be necessary to resolve cross-references.

4. Force a recompile to occur.

   Enter ⇒ `lwarpmk again`

   Enter ⇒ `lwarpmk print`

   `lwarpmk again` updates the date code for the file, triggering a recompile the next time the document is made.[3]

5. Process the index.[4]

   Enter ⇒ `lwarpmk printindex`

6. Recompile again to include the index.

   Enter ⇒ `lwarpmk print`

Note that the HTML customization commands are ignored while making the print version.

[3] Although, when using the utility `latexmk` (introduced later), the changed date is ignored and an actual change in contents must occur to cause a recompile.

[4] A `lwarpmk printglossary` command is also available to process a glossary produced with the glossaries package. See section 6.12.24.

5.3 Compiling the HTML version with `lwarpmk`

`lwarpmk html` is used to recompile an HTML version of the document.

1. Compile the HTML version:

   Enter ⇒ `lwarpmk html`

   (a) `lwarpmk` uses LaTeX to process `tutorial_html.tex` to create `tutorial_html.pdf`.

   (b) `pdftotext` is then used to convert to the file `tutorial_html.html`. This file is a plain-text file containing HTML tags and content for the entire document.

   (c) `lwarpmk` manually splits `tutorial_html.html` into individual HTML files according to the HTML settings. For this tutorial, the result is `tutorial.html` (the home page), along with `First-chapter.html`[5], `Some-math.html`, and the document's index in `_Index.html`.[6]

2. View the homepage in a web browser.

   Open the file `tutorial.html` in a web browser.

   math Note that math is still displayed as its plain-text LaTeX source until the images of the math expressions have been generated. Math may be displayed as SVG images or by a MathJax script, as seen in sections 5.4 and 5.5.

3. Force a recompile:

   Enter ⇒ `lwarpmk again`

   Enter ⇒ `lwarpmk html`

   Enter ⇒ `lwarpmk print`

4. Process the HTML index and recompile:[7]

   Enter ⇒ `lwarpmk htmlindex`

   Enter ⇒ `lwarpmk html`

   `_Index.html`, is updated for the new LaTeX index.

5. Reload the web page to see the added index.

[5] `First-chapter.html` also contains the first section, even though the second section is its own HTML page. This behavior is controlled by the boolean `CombineHigherDepths`.

[6] `index.html` is commonly used as a homepage, so the document index is in `_Index.html`.

[7] A `lwarpmk htmlglossary` command is also available to process a glossary produced with the glossaries package. See section 6.12.24.

5.4 Generating the SVG images

math as SVG images

By default lwarp represents math as SVG images with the LaTeX source included in `alt` tags. In this way, the math displays as it was drawn by LaTeX, and the LaTeX source may be copied and pasted into some other document.

`picture` and Ti*k*z

lwarp uses the same mechanism for `picture` and Ti*k*z environments.

1. Create the SVG images:

   Enter ⇒ `lwarpmk limages`

   Enter ⇒ `lwarpmk html`

2. Move to the tutorial's math page and reload.
3. The math images are displayed using the same font and formatting as the printed version.
4. Copy/paste a math expression into a text editor to see the LaTeX source.

⚠ Adding/removing

When a math expression, `picture`, or Ti*k*z environment is added or removed, the SVG images must be re-created with `lwarpmk limages` to maintain the proper image file sequence numbers.

⚠ Lots of files!

Expressing math as SVG images has the advantage of representing the math exactly as LaTeX would, but has the disadvantage of requiring an individual file for each math expression. There is no attempt at reusing the same file each time the same expression occurs, so each time x is used, for example, yet another file is created. For a document with a large amount of math, see section 5.5 to use MathJax instead.

5.5 Using MathJax for math

math with MathJax

Math may also be represented using the MathJax Javascript project.

1. In the tutorial's source code, uncomment the `mathjax` package option for lwarp:

   ```
   mathjax, % Use MathJax to display math.
   ```

2. Recompile

   Enter ⇒ `lwarpmk html`

3. Reload the math page.

⚠ MathJax requirements

MathJax requires web access unless a local copy of MathJax is available, and it also requires that Javascript is enabled for the web page. The math is rendered by MathJax. Right-click on math to see several options for rendering, and for copying the LaTeX source.

While using MathJax has many advantages, it may not be able to represent complex expressions or spacing adjustments as well as LaTeX.

5.6 Changing the CSS style

\CSSFilename

`\CSSFilename` may be used to choose which `.css` file is used to display each section of the web page. Use `\CSSFilename` before `\begin{document}` to assign the style of the home page. If different parts of the website should have different styles, call `\CSSFilename` again before each section heading which creates a new file.

The styles provided by lwarp include:

`lwarp.css`: A default style if `\CSSFilename` is not used. This style is comparable to a plain LaTeX document. To set this style, you may use `\CSSFilename{lwarp.css}`, or no `\CSSFilename` call at all.

`lwarp_formal.css`: A formal style with a serif fonts and a traditional look.

`lwarp_sagebrush.css`: A style with muted colors, gradient backgrounds, additional borders, and rounded corners.

To see each style in use, change the `\CSSFilename` entry in the tutorial, `lwarpmk html` again, and then reload the webpage.

Custom CSS

A customized style may also be created. For each new project a file called `sample_project.css` is generated. This may be renamed to `<project>.css` then used by assigning `\CSSFilename{<project>.css}`.

⚠ Rename it!

Note that `sample_project.css` is overwritten whenever lwarp is loaded in print mode. It is therefore important to rename the file to something like `<project>.css` before using it, so that your own changes are not overwritten.

`<project>.css` has an entry which loads `lwarp.css`, and this entry may be changed to load `lwarp_formal.css` or `lwarp_sagebrush.css` if desired. Additional changes to the CSS may be made by making entries later in the `<project>.css` file.

5.7 Customizing the HTML output

⚠ Placement!

Several settings may be used to customize the HTML output. Watch for the correct placement of each!

⚠ Changes!

Note that if changes are made, it is best to first:

1. Clear all the HTML, PDF, and auxiliary files:

   Enter ⇒ `lwarpmk clearall`

2. Recompile the print version in order to recreate the configuration files for `lwarpmk`:

   Enter ⇒ `lwarpmk print`

3. Finally, recompile the HTML version with the new settings:

   Enter ⇒ `lwarpmk html`

Options for the lwarp package:

Use the following as options for `\usepackage[<options>]{lwarp}`:

Opt `HomeHTMLFilename`

`HomeHTMLFilename`: Filename of the homepage, without the "`.html`" suffix. Defaults to the `\BaseJobname`. A common setting is:

```
HomeHTMLFilename=index
```

filename underscores

causing the homepage to be the file `index.html`. Underscores are allowed in `HomeHTMLFilename` and `HTMLFilename` options, but may need to be escaped elsewhere, such as when appearing in a list:

```
\item [\href{file\_name.pdf}{text}] \
```

Opt `HTMLFilename`

`HTMLFilename`: A filename prefix for the rest of the HTML web pages. Useful for numbered web pages with a common prefix. May be empty.

Opt `latexmk`

`latexmk`: Controls whether lwarp uses `latexmk` to compile the document. This setting is written to `lwarpmk`'s configuration files. Defaults to false.

Opt `mathsvg`

`mathsvg`: Selects SVG display for math output. (The default.)

Opt `mathjax`

`mathjax`: Selects MathJax for math output.

Placed in the preamble before `\begin{document}`:

Ctr `tocdepth`

`tocdepth`: Sectioning depth of the table of contents. See section 12 for a list of LaTeX stack depths.

Ctr `SideTOCDepth`

`SideTOCDepth`: Sectioning depth of the sideTOC. Defaults to 1, causing the sideTOC to show sections but not subsections.

sideTOC

Each subpage of the website has its own small table of contents on the side (the "sideTOC"). Its depth is set by `SideTOCDepth`. This sideTOC is only shown if the web page is wide enough. When using a narrow web browser window, "responsive web design" is used to show the sideTOC at the top of the page and a link back to "Home" at the bottom.

It is recommended to set:

```
SideTOCDepth = FileDepth
```

or

```
SideTOCDepth = FileDepth+1
```

⚠ If `SideTOCDepth` < `FileDepth`, web pages will be inaccessible via the sideTOC.

Ctr `FileDepth`

`FileDepth`: Sectioning depth of file splits. Defaults to -5, causing the entire HTML website to be one single file.

- To place the entire file into one HTML page, use:
  `\setcounter{FileDepth}{-5}`
- To split the HTML file at `\section` depth, use:
  `\setcounter{FileDepth}{1}`
- ⚠ To ensure that the HTML pages/files are accessible:
  Place a `\tableofcontents` somewhere before the first section break (therefore in the "home page"), and set
  `tocdepth >= FileDepth`

Bool `CombineHigherDepths`

`CombineHigherDepths`: Combine a higher section with its first lower subsections, down to the `FileDepth`. Defaults to true. Set to false to simulate the concept of a chapter opening on its own page, for example.

The file splits are controlled by the counter `FileDepth` and the boolean `CombineHigherDepths`. Setting `FileDepth` to 0 splits the file at chapters, 1 at sections, etc. `CombineHigherDepths` controls whether to combine pages at levels higher than the chosen `FileDepth`, such as in this tutorial where the page which opens the chapter also contains the first section. Be careful to set `tocdepth` and `SideTOCDepth` to allow access to each page of the website. Set `tocdepth` and `SideTOCDepth` to be greater than or equal to `FileDepth`.

⚠ Inaccesible pages!

⚠ Lost in an old page!

When making changes to the file structure, it is possible to end up with the web browser pointing to an old file which is no longer in use. When this occurs, changes to the web site will not appear in the browser, even

if reloading the page, because that page is no longer in use. It is best to return to the home page, clean the files (`lwarpmk cleanall`), change `FileDepth` and/or `CombineHigherDepths`, then finally recompile and renavigate to the desired page using the new file structure.

Bool `FileSectionNames`

`FileSectionNames`: If true, web page filenames are derived from a sanitized version of the section names. If false, web pages are numbered. Either way, the `HTMLFilename` option is used as a prefix.

HTML filenames

Example HTML filenames:

Numbered html nodes:
Example: Homepage `index.html`, and `node-1`, `node-2`. (See `\SetHTMLFileNumber` to number grouped by chapter, for example.)

```
\usepackage[
    HomeHTMLFilename=index,
    HTMLFilename={node-}
]{lwarp}
\boolfalse{FileSectionNames}
```

Named html sections, no prefix:
Example: `index.html`, and `About.html`, `Products.html`

```
\usepackage[
    HomeHTMLFilename=index,
    HTMLFilename={}
]{lwarp}
\booltrue{FileSectionNames}
```

Named html sections, with prefix:
Example: Homepage `mywebsite.html`, and additional pages such as `mywebsite-About.html`, etc.

```
\usepackage[
    HomeHTMLFilename=mywebsite,
    HTMLFilename={mywebsite-}
]{lwarp}
\booltrue{FileSectionNames}
```

`\abstractname`

`\abstractname`: The name of the abstract. This may also be over-written by the babel package. Defaults to "Abstract".

Placed before `\begin{document}`, or before any sectioning command which causes a file break:

`\CSSFilename`

`\CSSFilename`: `{`⟨*filename.css*⟩`}` Sets the CSS file to use for the following

files. May be changed before each each sectioning command which would cause a file split.

The CSS styles of the web pages are set by the `\CSSFilename` command. If `\CSSFilename` is not used, a default plain style is used to mimic printed LaTeX output. `lwarp_sagebrush.css` is a semi-fancy colored style as shown in this tutorial. Change it to `lwarp_formal.css` for a more formal look, or comment out the `\CSSFilename` command to see the default. `\CSSFilename` may be used before each file break to set the CSS for individual pagess of the website.

`\HTMLLanguage` **`\HTMLLanguage:`** The HTML file's `html lang` tag. Defaults to `en-US`.

`\HTMLAuthor` **`\HTMLAuthor:`** The HTML header's meta author. Defaults to `\theauthor`.

`\HTMLDescription` **`\HTMLDescription:`** `{`⟨*description*⟩`}` Sets the HTML description tag for the following files. May be changed before each each sectioning command which would cause a file split.

`\HTMLFirstPageTop` **`\HTMLFirstPageTop:`** `{`⟨*contents*⟩`}` A user-definable custom action applied to the top of the home page. Useful for logos, etc. Defaults empty. Ignored in print output.

`\HTMLPageTop` **`\HTMLPageTop:`** `{`⟨*contents*⟩`}` A user-definable custom action applied to the top of pages other than the home page. Useful for logos, etc. Defaults empty. `\LinkHome` may be used to place a link back to the homepage. Ignored in print output.

`\HTMLPageBottom` **`\HTMLPageBottom:`** `{`⟨*contents*⟩`}` A user-definable custom action applied to the bottom of each web page. Useful for authors, copyright notices, contact information, etc. Defaults empty. `\LinkHome` may be used to place a link back to the homepage. Ignored in print output.

Placed in the home page before the first sectioning command which causes a file break:

`\tableofcontents` ⚠ TOC on the homepage!

`\tableofcontents:` Used to place a table of contents on the home page. This command must be used before the first file split, so that a way is available to navigate to other files from the homepage.

Links to each chapter/section are provided, as selected by `tocdepth`.

Placed in the document wherever necessary:

Env `warpprint` **`warpprint:`** An environment which is only used while generating print output. Place here anything which does not apply to HTML and which may cause problems with lwarp. If lwarp knows about and emulates or supports a package then its related macros, lengths, counters, etc. probably

won't have to be placed inside a `warpprint` environment, but unknown packages may cause problems which may be isolated from lwarp using this environment.

Env `warpHTML`

`warpHTML`: An environment which is only used while generating HTML output. This is useful for website logos and other items which have no purpose in printed output.

`\warpprintonly`

`\warpprintonly`: `{`⟨*contents*⟩`}` A macro version of the `warpprint` environment.

`\warpHTMLonly`

`\warpHTMLonly`: `{`⟨*contents*⟩`}` A macro version of the `warpHTML` environment.

5.8 Using `latexmk`

`latexmk` is a LaTeX utility used to monitor changes in source files and recompile as needed.

1. In the tutorial's source code uncomment the `latexmk` option for the lwarp package:

   ```
   latexmk, % Use latexmk to compile.
   ```

2. Recompile the printed version of the document.

   Enter ⇒ `lwarpmk print`

   lwarp updates its own configuration files (`lwarpmk.conf` and `tutorial.lwarpmkconf`) whenever the printed version of the document is compiled. These configuration files remember that `lwarpmk` should use `latexmk` to compile the document.

3. Recompile the document.

   Enter ⇒ `lwarpmk print`

   and/or

   Enter ⇒ `lwarpmk html`

Changes are detected by comparing checksums rather than modification times, so `lwarpmk again` will not trigger a recompile, but `latexmk` has a much better awareness of changes than the `lwarpmk` utility does and it is likely to correctly know when to recompile. A recompile may be forced by making a small change to the source.

5.9 Using XeLaTeX or LuaLaTeX

XeLaTeX or LuaLaTeX may be used instead of LaTeX.

1. Remove the auxiliary files for the project:

   Enter ⇒ `lwarpmk cleanall`

2. Use `xelatex` or `lualatex` to recompile the printed version.

   Enter ⇒ `xelatex tutorial.tex`

   -or-

   Enter ⇒ `lualatex tutorial.tex`

   When the recompile occurs, the configuration files for `lwarpmk` are modified to remember which TeX engine was used. XeLaTeX or LuaLaTeX will be used for future runs of `lwarpmk`.

3. To recompile the document:

   Enter ⇒ `lwarpmk print`

   -and-

   Enter ⇒ `lwarpmk html`

4. Also rememeber to update the indexes and recompile again.

5.10 Using a glossary

lwarp supports the glossaries package, although this tutorial does not supply an example.

Opt `IndexLanguage` To assign a language to be used while processing the index and glossary, use the `IndexLanguage` option:

```
\usepackage[IndexLanguage=english]{lwarp}
```

To process the glossary for the print version:

Enter ⇒ `lwarpmk printglossary`

To process the glossary for the HTML version:

Enter ⇒ `lwarpmk htmlglossary`

In each case, the document will have to be recompiled afterwards.

5.11 Cleaning auxiliary files

To remove the auxiliary files `.aux`, `.toc`, `.lof`, `.lot`, `.idx`, `.ind`, `.log`, and `.gl*`:

Enter ⇒ `lwarpmk clean`

5.12 Cleaning auxiliary and output files

To remove the auxiliary files, and also remove the `.pdf` and `.html` files:

Enter ⇒ `lwarpmk cleanall`

5.13 Processing multiple projects in the same directory

It is possible to have several projects in the same directory. `lwarpmk` has an optional parameter which is the document to compile.

To create each project:

Enter ⇒ `pdflatex project_a`

Enter ⇒ `pdflatex project_b`

Each project is given its own configuration file:

`project_a.lwarpmkconf, project_b.lwarpmkconf`

To compile each project with `lwarkmk`:

Enter ⇒ `lwarpmk print project_a`

Enter ⇒ `lwarpmk html project_b`

5.14 Using the `make` utility

`lwarpmk` has an action which may be useful for integration with the common `make` utility:

`lwarpmk pdftohtml [project]`

`make` may be used to compile the code to PDF with HTML tags (`project_html.pdf`), then `lwarpmk` may be used to convert each target to HTML files.

6 Additional details

6.1 Font and UTF-8 support

lwarp uses `pdftotext` to convert PDF output into UTF-8-encoded text. This process requires that UTF-8 information be embedded in the PDF file, which usually prevents the use of bit-mapped fonts.

vector fonts
Computer Modern

While using `pdflatex`, if no font-related package is specified, the default bit-mapped Computer Modern font is used, so simply add

⚠ `usepackage{lmodern}`

to the preamble to enable the related vector font instead, or use

`\usepackage{dejavu}`

or other other font packages, which may provide an increased coverage of Unicode mappings. Avoid bit-mapped fonts.

⚠ XeLaTeX and LuaLaTeX users must use the fontspec package. Do NOT use fontenc!

Place fontspec or fontenc and other font and UTF-8 related commands after the `\documentclass` command and before `\usepackage{lwarp}`:

1. `documentclass{article/book/report}` goes here, followed by any of:
2. Font and UTF-8 related commands:
   - For XeLaTeX or LuaLaTeX:
     - fontspec and font choices

       Pkg `fontspec`

       ligatures

       lwarp sets the following to turn off TeX ligatures during the generation of HTML tags, and turn off common ligatures in regular text, since older browsers may not display them correctly and newer browsers can automatically re-create them.

       ```
       \defaultfontfeatures[\rmfamily]{Ligatures={NoCommon,TeX}}
       \defaultfontfeatures[\sffamily]{Ligatures={NoCommon,TeX}}
       \defaultfontfeatures[\ttfamily]{Ligatures=NoCommon}
       ```
   - For `pdflatex`:
     - lmodern or other font-related packages

       Pkg `lmodern`
     - fontenc

       Pkg `fontenc`
     - inputenc

       Pkg `inputenc`
     - newunicodechar

       Pkg `newunicodechar`
     - `\input glyphtounicode.tex`

       File `glyphtounicode`
     - `\input glyphtounicode-cmr.tex% from the pdfx package`

- `\pdfgentounicode=1`
- cmap
- textcomp
- microtype is automatically used by lwarp to turn off f,q,t,T,Q ligatures for the same browser-related reasons shown above. Also, the monospaced font is used during HTML tag generation to turn off TEX ligatures.

Pkg `cmap`
Pkg `textcomp`
Pkg `microtype`
ligatures

3. `\usepackage{lwarp}` (section 6.2) goes after any of the above, followed by:
4. ... the rest of the preamble and the main document.

6.1.1 Indexes and UTF-8

lwarp uses the `xindy` program to processes indexes.

While using `xelatex` or `lualatex`, `xindy` is used for the index. Everything is handled in UTF-8 encoding, and should work as expected.

While using `pdflatex`, the `texindy` program is used with the `-C utf8` option, which is newly supported in recent distributions of LaTeX. This option correctly sorts index entries into headings while using Latin languages, but will not work well with others. XeLaTeX or LuaLaTeX are recommended for non-Latin languages.

For an older distribution of LaTeX, it may be necessary to generate a local version of `lwarpmk.lua` and modify it to remove the `-C utf8` option from the texindy call. See section 9.3.

6.2 lwarp package loading and options

lwarp supports `book`, `report`, and `article` classes.

Pkg `lwarp` Load the lwarp package immediately after the font and UTF-8 setup commands.

Opt `warpprint`
Opt `warpHTML`

Select the `warpprint` option to generate print output (default), or the `warpHTML` option to generate HTML5 output. The default is print output, so the print version may be compiled with the usual `pdflatex`, etc. When lwarp is loaded in print mode, it creates `<project>_html.tex`, which sets the `warpHTML` option before calling the user's source code `<project>.tex`. In this way, `<project>.tex` can `\usepackage{lwarp}` without any options to create a printed version, while `<project>_html.tex` will create an HTML version.

Opt `mathsvg`
Opt `mathjax`

For math display, select `mathsvg` (default), or `mathjax`. For more information about the math options, see section 6.12.5.

See table 5 for the full list of options.

Table 5: Package options

Option	Description
warpprint	Generate print output, and also generate configuration files.
warpHTML	Generate HTML output.
mathsvg	Show math using SVG images.
mathjax	Show math using MathJax.
OSWindows	Force compatibility with MS-Windows.
BaseJobname	The \jobname to use. Set to the \jobname of the printed version even while generating HTML.
HomeHTMLFilename	The filename of the home page.
HTMLFilename	A prefix for the filenames of the remaining web pages.
IndexLanguage	The xindy language option used for index and glossary generation.
latexmk	Boolean for lwarpmk to use latexmk for compiling documents. Otherwise, lwarpmk attempts to recompile several times by itself.
lwarpmk	Generate a local copy of lwarpmk.lua.
xdyFilename	Tells lwarpmk to use a custom filename for xindy, instead of lwarp.xdy.

6.3 Selecting the operating system

Prog `Unix`
Prog `Mac OS`
Prog `Linux`

lwarp tries to detect which operating system is being used. UNIX / MAC OS / LINUX is the default (collectively referred to as "UNIX" in the configuration files), and MS-WINDOWS is supported as well.

Prog `MS-Windows`

If WINDOWS is not correctly detected, use the lwarp option `OSWindows`.

Prog `Windows`
Opt `OSWindows`

When detected or specified, the operating-system path separator used by lwarp is modified, the boolean `usingOSWindows` is set true. This boolean may be tested by the user for later use.

6.4 Selecting actions for print or HTML output

The following environments and macros are used to select actions which only apply to either traditional LaTeX print-formatted PDF generation, or to HTML generation.

For most of built-in LaTeX and many additional packages there is user-level source code support or emulation, so no special handling will be required. For those cases which lwarp does not handle by itself, the following environments and macros may be used to isolate sections of code for print-only or HTML-only.

These environments are also useful for creating a special version of the titlepage for print and another for HTML.

Env `warpHTML`

Anything which is to be done only for HTML5 output is surrounded by a `warpHTML` environment:

```
\begin{warpHTML}
 ... something to be done only during HTML generation
\end{warpHTML}
```

Env `warpprint`

Anything which is to be done only for print output is surrounded by a `warpprint` environment:

```
\begin{warpprint}
 ... something to be done only during traditional PDF generation
\end{warpprint}
```

Env `warpall`

Anything which is to be done for any output may be surrounded by a `warpall` environment. Doing so is optional.

```
\begin{warpall}
 ... something to be done during print PDF or HTML output
\end{warpall}
```

Macros are also provided for print-only or HTML-only code:

`\warpprintonly` {⟨*actions*⟩}

Performs the given actions only when print output is being generated.

`\warpHTMLonly` {⟨*actions*⟩}

Performs the given actions only when HTML output is being generated.

6.5 Commands to be placed into the `warpprint` environment

Certain print-related commands should always be placed inside a `warpprint` environment, or may need other special handling. These are unrelated to HTML output, but are hard to isolate automatically. For example:

- Paragraph formatting: `\parindent` `\parskip`
- Variable spaces such as `\vspace`. `\hfill` is turned into a `\quad`. Fixed spaces such as `\quad` are emulated correctly.
- Manual page positions such as the `textpos` package, which is emulated but only in a limited way.

Some packages require additional setup commands. Where these packages are emulated for HTML, setup commands may work for the emulated HTML output as well as for print output. See the details for each package in this document for more information.

Also see section 10: Troubleshooting.

6.6 Commands for a successful HTML conversion

Some commonly-used LaTeX expressions should be modified to allow for a smooth conversion to both HTML and print-formatted outputs:

Page references: The printed page does not translate to the HTML page, so references to page numbers are converted to parentheses containing

`\pagerefPageFor`, which defaults to "see ", followed by a hyperlink to the appropriate object. Ex: "Sec. 1.23 on page (see sec. 1.23)". `\pagerefPageFor` may be redefined to "page for ", empty, etc.

`\bfseries`, etc: Use `\textbf` instead.

`\centering`, `\raggedright`, `\raggedleft`:

Use the environments `center`, `flushright`, `flushleft` instead.

Superscripts and other non-math uses of math mode:

Use `\textsuperscript{x}` instead of `$^{x}$`

Empty `\item` followed by a new line of text or a nested list:

Use a trailing backslash: `\item[label] \`

Filenames in lists:

filename underscore

Escape underscores in the filenames:

```
\item[\href{file\_name.pdf}{text}]
```

Side-by-side minipages:

Place side-by-side minipages inside a `center` environment, with horizontal space between them, such as `\quad`, `\qquad`, `\hspace`, or `\hfill`. The result is similar in print and HTML. Do not use space commands at the start or end of the line.

`\fbox` around a minipage:

`\fbox` can only be used around inline items during HTML output.

For an `\fbox` around a minipage, you may:

- Place the `\fbox` command and its closing brace inside `warpprint` environments.
- Use `\mdframed` instead.
- Use a custom environment to create a sidebar, containing a `BlockClass` environment with custom CSS formatting, and `\warpprintonly{\hrule}` command:

```
\begin{BlockClass}{frameminipage}% ignored in print output
  % use CSS to format div class ``framedminipage''
\warpprintonly{\hrule} % only appears in print output
Contents
\warpprintonly{\hrule} % only appears in print output
\end{BlockClass}
```

Also see section 10: Troubleshooting.

6.7 Title page

In the preamble, place an additional block of code to set the following:

```
\title{Document Title} % One line only
\subtitle{Optional Document Subtitle \\ with optional multiple lines}
\author{Author One\affiliation{Affiliation One} \and
    Author Two\affiliation{Affiliation Two} }
\date{Optional date}
\published{Optional Journal Name \\ Optional multiple lines}
```

The title is used in the meta tags in the HTML files, and the rest are used in `\maketitle`.

`\maketitle` Use `\maketitle` just after the `\begin{document}`, as this will establish the title of the homepage. Optionally, use a `titlepage` environment instead.

Env `titlepage` The `titlepage` environment may be used to hold a custom title page. The titlepage will be set in a `<div>` class `titlepage`, and `\printtitle`, etc. may be used inside this environment.

Env `titlingpage` Another form of custom title page, where `\maketitle` is allowed, and additional information may be included as well.

`\title` {⟨*title*⟩}

⚠ Avoid newlines in the `\title`; these will interfere with the file break and CSS detection. Use the `\subtitle` command instead. In HTML, the title will appear in a heading `h1`.

`\author` {⟨*author*⟩}

⚠ In `\author`, use `\protect` before formatting commands such as `\textsc`. In HTML, the author will appear in a `<div>` class `author`. `\affiliation` is a new addition to lwarp.

`\date` {⟨*date*⟩}

`\date` works as expected. In HTML, this will appear in a `<div>` class `titledate`.

`\subtitle` {⟨*subtitle*⟩}

A new command which sets a subtitle. Newlines are allowed. The default is empty. In HTML, this will appear in a `<div>` class `subtitle`.

`\published` {⟨*published*⟩}

A new command which sets a publisher. The default is empty. In HTML, this will appear in a `<div>` class `published`.

`\thanks` {⟨*text*⟩}

`\thanks` are allowed in the titlepage fields, and will be rendered as HTML notes at the bottom of the title page.

6.8 HTML page meta descriptions

`\HTMLDescription` {⟨*A description of the web page.*⟩} The default is no description.

limitations Each page of HTML output should have its own HTML meta description, which usually shows up in web search results, is limited to around 150 characters in length, and should not include the ASCII double quote character (").

placement Use `\HTMLDescription` just before `\begin{document}` to set the description of the home page, and also just before each sectioning command such as `\chapter` or `\section` where a new file will be generated, depending on `FileDepth`. For example, if `FileDepth` is 1, use `\HTMLDescription` just before each `\section` command, and that description will be placed inside the HTML page for that `\section`. The same descrition will be used for all following HTML files as well, until reset by a new `\HTMLDescription`. It is best to use a unique description for each HTML file.

disabling To disable the generation of HTML description meta tags, use:
`\HTMLDescription{}`

6.9 HTML page meta author

`\HTMLAuthor` {⟨*author*⟩} Sets the contents of the web page `<meta name="author">` tag. Defaults to `\HTMLAuthor{\theauthor}`. May be set empty to cancel the meta author tag.

6.10 CSS

File `lwarp.css`
File `project.css`
File `sample_project.css`

It is best to make a local project-specific CSS file such as `project.css`, containing only things which are different from `lwarp.css`. `project.css` should refer to `lwarp.css` as follows:

```
/* ( --- Start of project.css --- ) */
/* A sample project-specific CSS file for lwarp --- ) */

/* Load default lwarp settings: */
@import url("lwarp.css") ;
/* or lwarp_formal.css, lwarp_sagebrush.css */
```

```
/* Project-specific CSS setting follow here. */
/* . . . */

/* ( --- End of project.css --- ) */
```

An example file called `sample_project.css` is provided, and may be renamed `project.css`.

`\CSSFilename` For each section at which HTML files are split, `\CSSFilename` may be used before the sectioning command to select a CSS file for that and all following sections. This may be changed numerous times throughout the file, resulting in different HTML pages having different CSS files assigned:

```
...
\newCSS{myCSS.css}
\chapter{Another Chapter}
...
```

6.11 Modifying `xindy` index processing

Prog `xindy`
File `lwarp.xdy`

lwarpmk uses the file `lwarp.xdy` to process the index. This file is over-written by lwarp whenever a print version of the document is processed.

To customize index processing:

1. Copy `lwarp.xdy` to a new filename such as `projectname.xdy`
2. Make changes to `projectname.xdy`. Keep the line which says

   ```
   (markup-locref :open "\hyperindexref{" :close "}")
   ```

   This line creates the hyperlinks for the HTML index. During print output `\hyperindexref` becomes a null function.

Opt `xdyFilename`

3. In the document source use the `xdyFilename` option for lwarp:

   ```
   \usepackage[
       ... other options ...
       xdyFilename=projectname.xdy,
   ]{lwarp}
   ```

4. Recompile the print version, which causes lwarp to rewrite the `lwarpmk.conf` configuration file. This tells `lwarpmk` to use the custom `projectname.xdy` file instead of `lwarp.xdy`.

6.12 Special cases and limitations

6.12.1 Text formatting

`\textbf`, etc. are supported, but `\bfseries`, etc. are not yet supported.

6.12.2 Cross-references

`\nameref` refers to the most recently-used section where the `\label` was defined. If no section has been defined before the `\label`, the link will be empty. Index entries also use `\nameref` and have the same limitation.

6.12.3 cleveref and varioref packages

cleveref and varioref are supported, but printed page numbers do not map to HTML, so a section name or a text phrase are used instead. See section 6.6 to redefine the message which is printed for page number references.

6.12.4 Footnotes and page notes

lwarp uses native LaTeX footnote code, although with its own `\box` to avoid the LaTeX output routine. The usual functions work as-is.

6.12.5 Math

Math may be rendered as SVG graphics or using the MathJax JavaScript display engine.

SVG math option

For SVG math, math is rendered as usual by LaTeX into the initial PDF file using the current font[8], then is captured from the PDF and converted to SVG graphics via a number of utility programs. The SVG format is a scalable-vector web format, so math may be typeset by LaTeX with its fine control and precision, then displayed or printed at any size, depending on (sometimes broken) browser support. An HTML `ALT` tag carries the LaTeX code which generated the math, allowing copy/paste of the LaTeX math expression into other documents.

SVG image font size

The size of the math and text used in the SVG image may be adjusted by setting `\LateximageFontSizeName` to a font size name — *without the backslash*, for ex:

```
\renewcommand{\LateximageFontSizeName}{large}
```

[8]See section 165 regarding fonts and fractions.

SVG files

As currently implemented, each instance of math creates a new SVG file. In text with many references to math variables, this can result in a large number of files with duplicate content. In the future, some method of content-based naming and checksumming may be used to remove the need for duplicate files.

SVG inline

Another approach would be to in-line the SVG files directly into the HTML. This avoids having a large number of files and potentially speeds loading the images, but dis-allows the possibility of sharing one file among many instances without user intervention.

PNG files

Others have used PNG files, sometimes pre-scaled for print resolution but displayed on-screen at a scaled down size. This allows high-quality print output at the expense of larger files, but SVG files are also larger as well.

MathML

Conversion to MathML might be a better approach, among other things allowing a more compact representation of math than SVG drawings. Problems with MathML include limited browser support and some issues with the fine control of the appearance of the result. Also see section 7 regarding EPUB output with MathJax.

MathJax math option

Prog `MathJax`

The popular MathJax alternative (`mathjax.org`) may be used to display math.

When MathJax is enabled, math is rendered twice:

1. As regular LaTeX PDF output placed inside an HTML comment, allowing equation numbering and cross referencing to be almost entirely under the control of LaTeX, and
2. As detokenized printed LaTeX commands placed directly into the HTML output for interpretation by the MathJax display scripts. An additional script is used to pre-set the equation number format and value according to the current LaTeX values, and the MathJax cross-referencing system is ignored in favor of the LaTeX internal system, seamlessly integrating with the rest of the LaTeX code.

MathJax limitations

Prog `MathJax`

Limitations when using MathJax include:

chapter numbers

- In document classes which have chapters, `\tag`ged equations have the chapter number prepended in HTML output, unlike LaTeX. `\tag*` equations (correctly) do not. This may be improved with future versions of the MathJax support script.

  `https://groups.google.com/forum/#!topic/mathjax-users/jUtewUcE2bY`

subequations

- MathJax itself does not support subequations. This may be improved by parsing the LaTeX math expression to manually insert tags, but this has not yet been done.

footnotes in math

- Footnotes inside equations are not yet supported while using MathJax.

lateximage

- Math appearing inside a lateximage, and therefore also inside a Ti*k*z or `picture` environment, is rendered as SVG math even if MathJax is used in the rest of the document.

siunitx

⚠ siunitx inside an equation

- Usage of siunitx inside a math equation is supported via a third-party MathJax extension. While inside a math expression, do not use `\SI` or `\si` inside `\text`, where it will be rendered as normal text.

  https://github.com/mathjax/MathJax-third-party-extensions/tree/master/siunitx

LaTeX macros

- MathJax does not automatically support custom LaTeX macros, but they may be set up by the user.

custom MathJax macros

For an example of using custom LaTeX macros with MathJax, see page 289.

6.12.6 ntheorem package

⚠ Font control

This conversion is not total. Font control is via CSS, and the custom LaTeX font settings are ignored.

⚠ Equation numbering

ntheorem has a bug with equation numbering in AMS environments when the option `thref` is used. lwarp does not share this bug, so equations with `\split`, etc, are numbered correctly with lwarp's HTML output, but not with the print output. It is recommended to use cleveref instead of ntheorem's `thref` option.

6.12.7 Graphics

⚠ graphics vs. graphicx

⚠ viewports

If using the older graphics syntax, use both optional arguments for `\includegraphics`. A single optional parameter is interpreted as the newer graphicx syntax. Note that viewports are not supported by warp; the entire image will be shown.

⚠ `\graphicspath`

`\graphicspath` only works for a single directory; all graphics must be in this directory.

units

For `\includegraphics`, avoid `px` and % units for width and height, or enclose them inside `warpHTML` environments. For font-proportional image sizes, use `ex` or `em`. For fixed-sized images, use `cm`, `mm`, `in`, `pt`, or `pc`. Using the keys `width=.5\linewidth`, or similar for `\textwidth` or `\textheight` to give fixed-sized images proportional to a 6 by 9 inch text area.

options

`\includegraphics` accepts `width` and `height`, `origin`, `rotate` and `scale`, plus a new `class` key.

HTML class

With HTML output, `\includegraphics` accepts an optional `class=xyz` keyval combination, and if this is given then the HTML output will include that class for the image. The class is ignored for print output.

⚠ image file types

For `\includegraphics` the user should provide both `.pdf` and `.svg` images, but always refer to `.pdf` images in the document source. All `\includegraphics` references to `.pdf` will automatically be changed to `.svg` for HTML output, and will be left as `.pdf` for print output. Images may also be `.jpg` and `.png`, and will be used as-is for either output.

`\rotatebox`

`\rotatebox` accepts the optional `origin` key.

⚠ browser support

`\rotatebox`, `\scalebox`, and `\reflectbox` depend on modern browser support. The CSS3 standard declares that when an object is transformed the whitespace which they occupied is preserved, unlike LaTeX, so expect some ugly results for scaling and rotating.

6.12.8 xcolor package

support

Color definitions, models, and mixing are fully supported without any changes required.

tables

Colored tables are ignored so far. Use CSS to style tables.

colored text and boxes

`\textcolor`, `\colorbox`, and `\fcolorbox` are supported.

`\color` and `\pagecolor`

`\color` and `\pagecolor` are ignored. Use CSS or `\textcolor` where possible.

6.12.9 Tabular

column types

- Vertical rules are not yet supported.
- `*` in a column specification is not used (so far). Repeat the column type the correct number of times.
- Only one each of `@`, `!`, `>`, and `<` may be used at each column, and they are used in that order.
- `\newcolumntype` is ignored; unknown column types are set to `l`.
- tabularx ignores the width, but `X` columns do produce paragraph columns or multicolumns.

⚠ `\multirow` & `\multicolumn`

- Multirow and multicolumn cannot be used at the same time. (No rectangular holes wider than one column or taller than one row.)

⚠ `\multirow`

- For multirow, insert `\mrowcell` into any empty multi-row cells. This will be a null function for the print output, and is a placeholder for parsing the

table for HTML output.

\multirow with rules

- If a multirow reaches to the bottom of a table, and \bottomrule does not go over to that edge, try adding a line of empty cells below the \bottomrule. This may be a browser bug.

rule at last row

- If a \midrule is desired after the last row, an additional row of blank cells must be used.

⚠ paragraphs

- Multiple paragraphs in one cell of a p, b, m column must have \newline between paragraphs.

\cmidrule width, trim

- \cmidrule does not support width or trim options due to CSS limitations.

longtable headings

- For longtable, place headings and footings which do not apply to HTML inside \warpprintonly{}.

⚠ \warpprintonly

- For \toprule and \bottomrule, when combined with a warpprint or warpHTML environment, if a "misplaced \noalign" error occurs, change

  ```
  This & That \endhead
  ```

  to

  ```
  \warpprintonly{This & That \endhead}
  ```

  and likewise with the other \end headings. Keep the \endfirsthead row unchanged, as it is still relevent to HTML output.

⚠ S columns

- For S columns (from the siunitx package), while producing print output, anything non-numeric must be placed inside { } braces, including commands such as \multirow. While producing HTML output, though, anything placed inside braces is not seen by lwarp's tabular handling algorithm. To resolve this problem, make a copy of the row, with one version for print output, containing the extra braces, and another version for HTML output, without the extra braces, such as:

  ```
  \warpprintonly{1 & 2 & {\multirow{2}{2cm}{Text}} & 3 \\}
  \warpHTMLonly{1 & 2 & \multirow{2}{2cm}{Text} & 3 \\}
  ```

6.12.10 longtable package

⚠ Longtable \endhead, \endfoot, and \endlastfoot rows are not used for HTML, and these rows should be disabled. Use

```
\warpprintonly{row contents}
```

instead of

```
\begin{warpprint} ... \end{warpprint}
```

Doing so helps avoid "Misplaced \noalign." when using \begin{warpprint}.

Keep the \endfirsthead row, which is still relevent to HTML output.

⚠ \kill is ignored, place a \kill line inside

```
\begin{warpprint} ... \end{warpprint}
```

or place it inside `\warpingprintonly`.

6.12.11 Save Boxes

LaTeX boxes are placed inline and do not allow line breaks, so boxes with long contents may overflow the line during HTML conversion. This is mostly a problem when the boxes contain objects which themselves hold large HTML tags, such as rotation commands with long contents. When this object overflows the line, some HTML code will be lost and the page will be corrupted.

6.12.12 Minipages

placement

Minipages and parboxes will be placed side-by-side in HTMLunless you place a `\newline` between them.

⚠ inline

A line of text with an inline minipage or parbox will have the minipage or parbox placed onto its own line, because a paragraph is a block element and cannot be made `inline-block`.

side-by-side

Side-by-side minipages may be separated by `\quad`, `\qquad`, `\enskip`, `\hspace`, `\hfill`, or a `\rule`. When inside a `center` environment, the result is similar in print and HTML. Paragraph tags are surpressed between side-by-side minipages and these spacing commands, but not at the start or end of the paragraph.

in a span

There is limited support for minipages inside an HTML `<span>`. An HTML `<div>` cannot appear inside a `<span>`. While in a `<span>`, minipages and parboxes are ignored. Use `\newline` or `\par` for an HTML break.

size

When using `\linewidth`, `\textwidth`, and `\textheight`, widths and heights are scaled proportionally to a 6×9 inch text area.

no-width minipages

A minipage of width exactly `\linewidth` is automatically given no HTML width.

full-width minipages

A new macro `\minipagefullwidth` requests that the next minipage be generated without an HTML `width` tag, allowing it to be the full width of the display rather than the fixed width given.

⚠ text alignment

Nested minipages adopt their parent's text alignment in HTML, whereas in regular LaTeX PDF output they do not. Use a `flushleft` or similar environment in the child minipage to force a text alignment.

6.12.13 mdframed package

support Most basic functionality is supported, including frame background colors and single-border colors and thickness, title and subtitle background colors and borders and thickness, border radius, and shadow. CSS classes are created for mdframed environments and frame titles.

⚠ loading When used, lwarp loads mdframed in HTML with `framemethod=none`.

font For title font, use

```
frametitlefont=\textbf,
```

instead of

```
frametitlefont=\bfseries,
```

where `\textbf` must appear just before the comma and will receive the following text as its argument (since the text happens to be between braces in the mdframed source). Since lwarp does not support `\bfseries` and friends, only one font selection may be made at a time.

theoremtitlefont `theoremtitlefont` is not supported, since the following text is not in braces in the mdframed source.

footnotes Footnotes are currently placed at the bottom of the HTML page.

ignored options `userdefinedwidth` and `align` are currently ignored.

6.12.14 float, trivfloat, and/or algorithmicx together

⚠ package conflicts If using `\newfloat`, trivfloat, and/or algorithmicx together, see section 159.1.

6.12.15 caption and subcaption packages

To ensure proper float numbering, set caption positions such as:

```
\captionsetup[table]{position=top}
\captionsetup[figure]{position=bottom}
```

Similarly for `subtable`, `subfigure`, and `longtable`.

6.12.16 subfig package

⚠ `lof/lotdepth` At present, the package options for `lofdepth` and `lotdepth` are not working. These counters must be set separately after the package has been loaded.

horizontal spacing

In the document source, use `\hfill` and `\hspace*` between subfigures to spread them apart horizontally. The use of other forms of whitespace may cause paragraph tags to be generated, resulting in subfigures appearing on the following lines instead of all on a single line.

6.12.17 floatrow package

⚠ subfig package

When combined with the subfig package, while inside a `subfloatrow` `\ffigbox` and `\ttabbox` must have the caption in the first of the two of the mandatory arguments.

⚠ `\FBwidth`, `\FBheight`

The emulation of floatrow does not support `\FBwidth` or `\FBheight`. These values are pre-set to `.3\linewidth` and `2in`. Possible solutions include:

- Use fixed lengths. lwarp will scale the HTML lengths appropriately.
- Use `warpprint` and `warpHTML` environments to select appropriate values for each case.
- Inside a `warpHTML` environment, manually change `\FBwidth` or `\FBheight` before the `\ffigbox` or `\ttabbox`. Use `\FBwidth` or `\FBheight` normally afterwards; it will be used as expected in print output, and will use your custom-selected value in HTML output. This custom value will be used repeatedly, until it is manually changed to a new value.

6.12.18 abstract package

abstract is supported. If using the `number` option with file splits, be sure to place the table of contents before the abstract. The number option causes a section break which may cause a file split, which would put a table of contents out of the home page if it is after the abstract.

6.12.19 verse and memoir

`\attrib`

The documentation for the verse and memoir packages suggest defining an `\attrib` command, which may already exist in current documents, but it will only work for print output. lwarp provides `\attribution`, which works for both print and HTML output. To combine the two so that `\attrib` is used for print and `\attribution` is used for HTML:

```
\begin{warpHTML}
\let\attrib\attribution
\end{warpHTML}
```

Len \leftskip
Len \leftmargini
Len \TMLvleftskip
Len \TMLleftmargini

These lengths are used by verse and memoir to control the left margin, and they may already be set by the user for print output. New lengths \HTMLvleftskip and \HTMLleftmargini are provided to control the margins in HTML output. These new lengths may be set by the user before any verse environment, and persist until they are manually changed again. One reason to change \HTMLleftmargini is if there is a wide \flagverse in use, such as the word "Chorus", in which case the value of \HTMLleftmargini should be set to a wide enough length to contain "Chorus". The default is wide enough for a stanza number.

Horizontal spacing relies on pdftotext's ability to discern the layout (-layout option) of the text in the HTML-tagged PDF output. For some settings of \HTMLleftmargini or \HTMLleftskip the horizontal alignment may not work out exactly, in which case a label may be shifted by one space.

6.12.20 siunitx package

Pkg siunitx
⚠ per-mode

Do not use per-mode=fraction, which cannot be seen by the final pdftotext conversion.

6.12.21 newclude package

Pkg newclude
⚠ loading

newclude modifies \label in a non-adaptive way, so newclude must be loaded before lwarp is loaded.

Ex:

```
\documentclass{article}
...<font setup>
\usepackage{newclude}
\usepackage[warpHTML]{lwarp}
...
```

6.12.22 newtxmath package

Pkg newtxmath
⚠ loading sequence

The proper load order is:

1. ...
2. `\usepackage{lwarp}`
3. ...
4. `\usepackage{amsthm}`
5. `\usepackage{newtxmath}`
6. ...

6.12.23 babel package

Pkg `babel`

⚠ French

If using babel with French, use

```
\frenchbsetup{StandardLists=true}
```

to preserve the special HTML and enumitem list handling.

`\CaptionSeparator`

Also, when French is used, the caption separator is changed to a dash. The following may be used to restore it to a colon:

```
\renewcommand*{\CaptionSeparator}{:~}
```

6.12.24 glossaries package

Pkg `glossaries`

`xindy` is required for glossaries.

The default `style=item` option for glossaries conflicts with lwarp, so the style is forced to `index` instead.

The page number list in the printed form would become `\namerefs` in HTML, which could become a very long string if many items are referenced. For now, the number list is simply turned off.

`lwarpmk` has the commands `printglossary` and `htmlglossary` to process the glossaries created by glossaries using `xindy`.

Opt `IndexLanguage`

The package lwarp takes an option `IndexLanguage=english` to set the language used by `xindy`. This is passed to `xindy` using its `-L` option, and is used for both index and glossary generation.

6.12.25 enumitem package

Pkg `enumitem`

enumitem

enumitem is pre-loaded during HTML output. Many of the spacing options are rendered irrelevant by `pdftotext` and HTML. Numbering, labels, and `\newlist` function correctly.

6.12.26 enumerate package

Pkg `enumerate` enumerate conflicts with enumitem if both are loaded at the same time, but lwarp does not actually load enumerate. While generating HTML, lwarp only loads enumitem, and enumerate is simulated by enumitem using the functionality of the `shortlabels` option.

A problem may occur during print output if enumitem is loaded, either manually or by some other package such as siunitx. If these are used, enumerate will conflict with enumitem during print output.

7 EPUB conversion

lwarp does not produce EPUB documents, but it may be told to modify its HTML output to greatly assist in the conversion. An external program may then be used to finish the conversion to EPUB.

<meta> author

To assign the author's name for regular lwarp HTML files, and also for the EPUB, use `\HTMLAuthor {⟨name⟩}`. This assigns the name to the `<meta>` author tag. It may be set empty, and it defaults to `\theauthor`.

A special boolean is provided to simplify the process of converting lwarp HTML output to EPUB:

Bool `FormatEPUB`

Default false. `FormatEPUB` changes HTML output for easy EPUB conversion via an external program. Removes per-file headers, footers, and nav. Adds footnotes per chapter/section.

To help convert lwarp HTML output to EPUB, add

```
\booltrue{FormatEPUB}
```

to the project's source preamble after `\usepackage{lwarp}`. The EPUB version of the document cannot co-exist with the regular HTML version, so

Enter ⇒ `lwarpmk cleanall`

then

Enter ⇒ `lwarpmk html`

to recompile with the `FormatEPUB` boolean turned on. Several changes are then made to the HTML output:

- Headers, footers, and navigation are removed at file splits.
- Any accumulated footnotes are printed at the bottom of each file split.

Calibre

The resulting files will be ready to be loaded into an EPUB conversion program, such as the open-source program Calibre (https://calibre-ebook.com/).

⚠ search order

The EPUB conversion program must know what order the files are included. For lwarp projects, set the EPUB conversion software to do a breadth-first search of the files. For Calibre, this option is found in

Preferences → Plugins → File type plugins → HTML to Zip

Check the box Add linked files in breadth first order.

⚠ section breaks

The EPUB-conversion program must also know where the section breaks are located. For a list of lwarp's section headings, see table 6. For example, an `article` class document would break at `\section`, which is mapped to HTML heading level `<h4>`,

whereas a `book` class document would break at `\chapter`, which is HTML heading level `<h3>`. For Calibre, this option is found in

Preferences → Conversion (Common Options) → Structure Detection → Detect chapters at (XPath expression)

Select the "magic wand" to the right of this entry box, and set the first entry

Math HTML tags with tag name:

to `<h4>`. (Or `<h3>` for document classes with `\chapter`s.) The Detect chapters at field should then show

//h:h4 — or — //h:h3

This option is also available on the main tool bar at the Convert books button.

Once these settings have been made, the lwarp-generated HTML files may be loaded by Calibre, and then converted to an EPUB.

MathJax support

MathJax may be used in EPUB documents. Some e-readers include MathJax, but any given reader may or may not have a recent version, and may or may not include extensions such as support for siunitx.

lwarp adds some modifications to MathML to support equations numbered by chapter. These modifications may not be compatible with the e-reader's version of MathJax, so lwarp requests that a known version be loaded instead. In some cases chapter numbering of equations still doesn't work.

Until math support in EPUB documents is improved, it is recommended to use SVG images instead of MathJax, especially for equations numbered by chapter, or where siunitx support is important.

8 Word-processor conversion

lwarp may be told to modify its HTML output to make it easier to import the HTML document into a word processor. At the time of this writing, it seems that LibreOffice works best at preserving table layout, but it still has some limitations, such as an inability to automatically assign figure and table frames and captions according to user-selected HTML classes. lwarp provides some assistance in locating these frame boundaries, as shown below.

A special boolean is provided to simplify the process of converting lwarp HTML output to EPUB:

Bool `FormatWordProcessor`

Default false. Changes HTML output for easier conversion by a word processor. Removes headers and nav, prints footnotes per section, and also forces single-file output and turns off HTML debug comments.

To help modify lwarp HTML output for easier import to a word processor, add

```
\booltrue{FormatWordProcessor}
```

to the project's source preamble after lwarp is loaded. Several changes are then made to the HTML output:

- Headers, footers, and navigation are removed at file splits.
- Any accumulated footnotes are printed at the bottom of each file split. These will have to be manually moved to their proper place in the document. lwarp does not know where the page breaks will be in the word processor's document, so the footnotes are simply moved to the end of each sectional break.
- Forces single-file output.
- Turns off HTML debugging comments. These are comments appearing inside the HTML code, marking the opening/closing of sections and `<div>`s, but they are no longer useful when the document has been imported into a word processor.

An additional boolean may be set to help mark float boundaries:

Bool `HTMLMarkFloats`

Default true. Adds `=== table begin` or `=== figure begin`, and `=== end` around floats while formatting for word processors. This helps identify boundaries of floats to be manually converted to word-processor frames.

When enabled, markers are placed around each float, helping the user to identify float boundaries for further conversion to word processor frames and captions.

9 Modifying lwarp

Purely text-based packages probably will work as-is when generating HTML.

Look to existing code for ideas on how to expand into new code.

An environment may be converted to a `lateximage` then displayed with an image of the resulting LaTeX output. See section 66 for an example of the `picture` environment.

To create a custom HTML block or inline CSS class, see section 35.7.

9.1 Creating an lwarp version of a package

When creating HTML, lwarp redefines the `\usepackage` and `\RequirePackage` macros such that it first looks to see if a `lwarp-<packagename>.sty` version exists. If so, the lwarp version is used instead. This modular system allows users to create their own versions of packages for lwarp to use for HTML, simply by creating a new package with a `lwarp-` prefix. If placed in the local directory along with the source code, it will be seen by that project alone. If placed alongside the other `lwarp-` packages where TeX can see it, then the user's new package will be seen by any documents using lwarp. (Remember `mktexlsr` or `texhash`.)

An `lwarp-<packagename>.sty` package is only used during HTML generation. Its purpose is to pretend to be the original package, while modify anything necessary to create a successful HTML conversion. For many packages it is sufficient to simply provide nullified macros, lengths, counters, etc. for anything which the original package does, while passing the raw text on to be typeset. See the pre-existing `lwarp-` packages for examples.

Anything the user might expect of the original package must be replaced or emulated by the new `lwarp-` package, including package options, user-adjustable counters, lengths, and booleans, and conditional behaviors. In many of these packages, most of the new definitions have a "local" prefix according to the package name, and `@` characters inside the name, which hides these names from the user. In most cases these macros will not need to be emulated for HTML output. Only the "user-facing" macros need to be nullified or emulated.

Each `lwarp-` package should first call either `\LWR@ProvidesPackageDrop` or `\LWR@ProvidesPackagePass`. If `Drop`ped, the original print-version package is ignored, and only the `lwarp-` version is used. Use this where the original print version is useless for HTML. If `Pass`ed, the original package is loaded first, with the user-supplied options, then the `lwarp-` version continues loading as well. See section 135 (Ntheorem) for an example of selectively disabling user options for a package. Use this when HTML output only requires some modifications of the

original package. For a case where the original package is usable without changes, there is no need to create a `lwarp-` version.

9.2 Testing lwarp

When changes have been made, test the print output before testing the HTML. The print output compiles faster, and any errors in the printed version will be easier to figure out than the HTML version.

Remember that the configuration files are only rewritten when compiling the printed version of the document.

Sometimes it is worth checking the `<project>_html.pdf` file, which is the PDF containing HTML tags. Also, `<project>_html.html` has the text conversion of these tags, before the file is split into individual HTML files.

It is also worth checking the browser's tools for verifying the correctness of HTML and CSS code.

9.3 Modifying `lwarpmk`

Prog `lwarpmk`
File `lwarpmk.lua`

In most installations, `lwarpmk.lua` is an executable file located somewhere the operating system knows about, and it is called by typing "`lwarpmk`" into a terminal.

A project-local copy of `lwarpmk.lua` may be generated, modified, and then used to compile documents:

1. Add the `lwarpmk` option to the lwarp package.
2. Recompile the printed version of the document. The `lwarpmk` option causes lwarp to create a local copy of `lwarpmk.lua`
3. The `lwarpmk` option may now be removed from the lwarp package.
4. Copy and rename `lwarpmk.lua` to a new file such as `mymake.lua`.
5. Modify `mymake.lua` as desired.
6. If necessary, make `mymake.lua` executable.
7. Use `mymake.lua` instead of `lwarpmk.lua`.

To adjust the command-line arguments for compiling the document, look in `mymake.lua` for "`latexname`".

To adjust the command-line arguments for processing the index, look for "`xindy`".

10 Troubleshooting

10.1 Using the `lwarp.sty` package

Also see:
Section 6.5: Commands to be placed into the `warpprint` environment
Section 6.6: Commands for a successful HTML conversion
Section 6.12: Special cases and limitations

Text is not converting:

- Font-related UTF-8 information must be embedded in the PDF file. See section 6.1 regarding vector fonts.

Undefined html settings:

See the warning regarding the placement of the HTML settings at section 5.7.

Obscure error messages:

- Be sure that a print version of the document compiles and that your document's LaTeX code is correct, before attempting to generate an HTML version.

Missing sections:

- See section 5.7 regarding the `FileDepth` and `SideTOCDepth` counters, and the use of `\tableofcontents` in the home page.

Missing html files:

See the warning regarding changes to the HTML settings at section 5.7.

Missing / incorrect cross-references:

- Use `lwarpmk again` followed by `lwarpmk html` or `lwarpmk print` to compile the document one more time.
- `\nameref` refers to the most recently-used section where the `\label` was defined. If no section has been defined before the `\label`, the link will be empty. Index entries also use `\nameref` and have the same limitation.
- cleveref and varioref are supported, but printed page numbers do not map to HTML, so a section name or a text phrase are used instead. See section 6.6 to redefine the message which is printed for page number references.

Em-dashes or En-dashes in listing captions and titles:

Use XeLaTeX or LuaLaTeX.

Floats out of sequence:

Mixed "Here" and floating: Floats [H]ere and regular floats may become out of order. `\clearpage` if necessary.

Caption setup: With `\captionsetup` set the positions for the captions `above` or `below` to match their use in the source code.

Print document contains html tags:

- Be sure that the document selects `\usepackage[warpprint]{lwarp}` instead of `[warpHTML]`.

HTML document contains a single unformatted print document:

- Be sure that the document selects `\usepackage[warpHTML]{lwarp}` instead of `[warpprint]`.

Images are appearing in strange places:

- `lwarpmk limages` to refresh the `lateximage` images.

"Leaders not followed by proper glue": This can be caused by a missing `l@<floattype>` or `l@<sectiontype>` definition. See lwarp's definitions for examples.

Plain-looking document:

- The document's CSS stylesheet may not be available, or may be linked incorrectly. Verify any `\CSSFilename` statements point to a valid CSS file.

Broken fragments of HTML:

- Check the PDF file used to create HTML to see if the tags overflowed the margin. (This is why such large page size and margins are used.)

Changes do not seem to be taking effect:

- Be sure to `lwarpmk clean`, recompile, then start by reloading the home page. You may have been looking at an older version of the document. If you changed a section name, you may have been looking at the file for the old name.
- See the warning regarding changes to the HTML settings at section 5.7.
- Verify that the proper CSS is actually being used.
- The browser may compensate for some subtle changes, such as automatically generating ligatures, reflowing text, etc.

Un-matched conditional compiles:

- Verify the proper `begin/end` of `warpprint`, `warpHTML`, and `warpall` environments.

10.1.1 Debug tracing output

\tracinglwarp When \tracinglwarp is used, lwarp will add extra tracing messages to the .log file. The last several messages may help track down errors.

Place \tracinglwarp just after \usepackage{lwarp} to activate tracing.

10.2 Compiling the lwarp.dtx file

lwarp_tutorial.tex: Copy or link lwarp_tutorial.txt from the TDS doc directory to the source directory, or wherever you wish to compile the documentation. This file is included verbatim into the documentation, but is in the doc directory so that it may be found by texdoc and copied by the user.

Illogical error messages caused by an out-of-sync lwarp.sty file:

1. Delete the lwarp.sty file.
2. pdflatex lwarp.ins to generate a new lwarp.sty file.
3. pdflatex lwarp.dtx to recompile the lwarp.pdf documentation.

Un-nested environments:

Be sure to properly nest:

- \begin{macrocode} and \end{macrocode}
- \begin{macro} and \end{macro}
- \begin{environment} and \end{environment}

11 Implementation

This package is perhaps best described as a large collection of smaller individual technical challenges, in many cases solved through a number of ~~crude hacks~~ clever tricks. Reference sources are given for many of the solutions, and a quick internet search will provide additional possibilities.

Judgement calls were made, and are often commented. Improvements are possible. The author is open to ideas and suggestions.

Packages were patched for re-use where they provided significant functionality. Examples include xcolor with its color models and conversion to HTML color output, and siunitx which provides many number and unit-formatting options, almost all of which are available in pure-text form, and thus easily used by pdftotext.

Packages were emulated where their primary purpose was visual formatting which is not relevent to HTML output. For example, packages related to sectioning are already patched by numerous other packages, creating a difficult number of combinations to try to support, and yet in HTML output all of the formatting is thrown away, so these packages are merely emulated.

Packages with graphical output are allowed as-is, but must be nested inside a `lateximage` environment to preserve the graphics.

There is still room to improve the factoring of the code, and doing so will become important if support for other output formats is added. Rather than wait until the code is pristine, the author felt it best to publish early and accept input before pushing on towards a perhaps less-than-ideal solution.

Testing has primarily been done with the Iceweasel/Firefox browser.

12 Stack depths

Stacks are created to track depth inside the LaTeX document structure. This depth is translated to HTML headings as shown in table 6. "Depth" here is not depth in the traditional computer-science stack-usage sense, but rather a representation of the nesting depth inside the LaTeX document structure.

When starting a new section, the program first must close out any existing sections and lists of a deeper level to keep the HTML tags nested correctly.

Support for the memoir package will require the addition of a `book` level, which may push the HTML headings down a step, and also cause `subsubsection` to become a `<div>` due to a limit of six HTML headings.

It is possible to use HTML5 `section` and `H1` for all levels, but this may not be well-recognized by older browsers.

Fixed levels for parts and chapters allow the CSS to remain fixed as well.

Table 6: Section depths and HTML headings

Section	LaTeX depth	HTML headings
title of the entire website		h1
none	-5	new for this package
book	-2	**not yet used**
part	-1	h2
chapter	0	h3
section	1	h4
subsection	2	h5
subsubsection	3	h6
paragraph	4	span class = "paragraph"
subparagraph	5	span class = "subparagraph"
listitem	7	new for this package, used for list items

13 Source Code

This is where the documented source code for lwarp begins, continuing through the following sections all the way to the change log and index at the end of this document.

The following sections document the actual implementation of the lwarp package.

line numbers — The small numbers at the left end of a line refer to line numbers in the `lwarp.sty` file.

subjects — Blue-colored tags in the left margin aid in quickly identifying the subject of each paragraph.

objects — Black-colored tags in the left marign are used to identify programming objects such as files, packages, environments, booleans, and counters. Items without a tag

index entries — are command macros. Each of these also appears in the index as individual entries, and are also listed together under "files", "packages", "environments", "booleans", and "counters".

⚠ warnings — Special warnings are marked with a warning icon.

for HTML output:
for PRINT output:
for HTML & PRINT:

Green-colored tags in the left margin show which sections of source code apply to the generation of HTML, print, or both forms of output.

lwarp source code begins on the following page.

14 Detecting the TEX Engine — pdflatex, lualatex, xelatex

```
\RequirePackage{iftex}

\ifLuaTeX
\RequirePackage{luatex85}% until the geometry package is updated
\fi
```

15 Unicode Input Characters

for HTML & PRINT:

If using `pdflatex`, convert a minimal set of Unicode characters. Additional characters may be defined by the user, as needed.

A commonly-used multiply symbol is declared to be `\texttimes`.

The first arguments of `\newunicodechar` below are text ligatures in the source code, even though they are not printed in the following listing.

```

\RequirePackage{newunicodechar}

\newunicodechar{×}{\texttimes}

\ifPDFTeX
\newunicodechar{ff}{ff}% the first arguments are ligatures
\newunicodechar{fi}{fi}
\newunicodechar{fl}{fl}
\newunicodechar{ffi}{ffi}
\newunicodechar{ffl}{ffl}
\newunicodechar{--}{---}
\newunicodechar{-}{--}
```

In PDFTEX, preserve upright quotes in verbatim text:

```
\RequirePackage{upquote}
\else
\fi
```

16 Early package requirements

Pkg `etoolbox` Provides `\ifbool` and other functions.

```
\RequirePackage{etoolbox}[2011/01/03]
% requires v2.6 for \BeforeBeginEnvironment, etc.
```

Pkg `ifplatform` Provides `\ifwindows` to try to automatically detect Windows OS.

```
\RequirePackage{ifplatform}% sense op-system platform
```

Pkg `comment` Provides conditional code blocks.

```
\RequirePackage{comment}
\excludecomment{testing}
```

17 Operating-System portability

Prog `Unix`
Prog `Mac OS`
Prog `Linux`
Prog `MS-Windows`
Prog `Windows`
Opt `OSWindows`

lwarp tries to detect which operating system is being used. UNIX / MAC OS / LINUX is the default (collectively referred to as "UNIX" in the configuration files), and MS-WINDOWS is supported as well.

If WINDOWS is not correctly detected, use the lwarp option `OSWindows`.

When detected or specified, the operating-system path separator used by lwarp is modified, the boolean `usingOSWindows` is set true. This boolean may be tested by the user for later use.

17.1 Common portability code

Bool `usingOSWindows`

Set if the `OSWindows` option is used.

```
\newbool{usingOSWindows}
\boolfalse{usingOSWindows}
```

17.2 Unix, Linux, and Mac OS

`\OSPathSymbol`

Symbol used to separate directories in a path.

```
\newcommand*{\OSPathSymbol}{/}
```

17.3 MS-Windows

For MS-Windows:

`\LWR@setOSWindows`

Set defaults for the MS-Windows operating system. lwarp attempts to auto-detect the operatings system, and the `OSWindows` option may also be used to force MS-Windows compatibility.

```
\newcommand*{\LWR@setOSWindows}
{
\booltrue{usingOSWindows}
\renewcommand*{\OSPathSymbol}{\@backslashchar}
}
```

Test for windows during compile. The user may also specify `OSWindows` package option in case this test fails.

```
\ifwindows
\LWR@setOSWindows
\fi
```

18 Package options

Pkg **kvoptions** Allows key/value package options.

```
\RequirePackage{kvoptions}
\SetupKeyvalOptions{family=LWR,prefix=LWR@}
```

Bool **warpingprint**
Bool **warpingHTML**
Bool **mathjax**

Set to true/false depending on the package option selections for print/HTML/EPUB output and mathsvg/mathjax:

```
\newbool{warpingprint}
\newbool{warpingHTML}
\newbool{mathjax}
```

\warpprintonly {⟨*contents*⟩}

Only process the contents if producing printed output.

```
\newcommand{\warpprintonly}[1]{\ifbool{warpingprint}{#1}{}}
```

\warpHTMLonly {⟨*contents*⟩}

Only process the contents if producing HTML output.

```
\newcommand{\warpHTMLonly}[1]{\ifbool{warpingHTML}{#1}{}}
```

Env **warpall** Anything in the `warpall` environment will be generated for print or HTML outputs.

```
\includecomment{warpall}
```

Env **warpprint** Anything in the `warpprint` environment will be generated for print output only.

Opt **warpprint** If the `warpprint` option is given, boolean `warpingprint` is true and boolean `warpingHTML` is false, and may be used for `\ifbool` tests.

```
\DeclareVoidOption{warpprint}{%
\PackageInfo{lwarp}{Using option 'warpprint'}
\includecomment{warpprint}%
```

```
\excludecomment{warpHTML}%
\booltrue{warpingprint}%
\boolfalse{warpingHTML}%
}
```

Env `warpHTML` Anything in the `warpHTML` environment will be generated for HTML output only.

Opt `warpHTML` If the `warpHTML` option is given, boolean `warpingHTML` is true and boolean `warpingprint` is false, and may be used for `\ifbool` tests.

```
\DeclareVoidOption{warpHTML}{%
\PackageInfo{lwarp}{Using option 'warpHTML'}
\excludecomment{warpprint}%
\includecomment{warpHTML}%
\booltrue{warpingHTML}%
\boolfalse{warpingprint}%
}
```

Opt `mathsvg` Option `mathsvg` selects SVG math display: If the `mathsvg` option is given, boolean `mathjax` is false, and may be used for `\ifbool` tests.

```
\DeclareVoidOption{mathsvg}{%
\PackageInfo{lwarp}{Using option 'mathsvg'}
\boolfalse{mathjax}%
}
```

Opt `mathjax` Option `mathjax` selects MathJax math display: If the `mathjax` option is given, boolean `mathjax` is true, may be used for `\ifbool` tests.

```
\DeclareVoidOption{mathjax}{%
\PackageInfo{lwarp}{Using option 'mathjax'}
\booltrue{mathjax}%
}
```

Opt `BaseJobname` Option `BaseJobname` sets the `\BaseJobname` for this document.

This is the `\jobname` of the printed version, even if currently compiling the HTML version. I.e. this is the `\jobname` without `_html` appended. This is used to set `\HomeHTMLFilename` if the user did not provide one.

```
\DeclareStringOption[\jobname]{BaseJobname}
```

Opt `IndexLanguage` Sets the language to be assigned in `lwarpmk`'s configuration files. This is then used by `lwarpmk` while processing the index and glossary.

```
\DeclareStringOption[english]{IndexLanguage}
```

Opt `xdyFilename` Selects a custom `.xdy` file. The default is `lwarp.xdy`. A customized file should be based on `lwarp.xdy`, and must retain the line

```
arkup-locref :open "\hyperindexref{" :close "}")
```

```
\DeclareStringOption[lwarp.xdy]{xdyFilename}
```

Opt `lwarpmk` Tells lwarp to generate a local copy of `lwarpmk` called `lwarpmk.lua`. Useful for archiving for future use. This file may be made executable and acts just like `lwarpmk`.

If `lwarpmk` option, creates a local copy of `lwarpmk.lua`:

```
\DeclareVoidOption{lwarpmk}{
\PackageInfo{lwarp}{Using option 'lwarpmk'}
\includecomment{LWR@createlwarpmk}
}
```

Opt `OSWindows` Tells lwarp to use MS-Windows compatibility. Auto-detection of the operating system is attempted, and this option is only necessary if the auto-detection fails. See the automatically-generated `lwarpmk.conf` file to find out whether the operating system was detected correctly.

```
\DeclareVoidOption{OSWindows}{
\PackageInfo{lwarp}{Using option 'OSWindows'}
\LWR@setOSWindows
}
```

Opt `HomeHTMLFilename` The filename of the homepage. The default is the jobname. This option is stored into `\LWR@HomeHTMLFilename`, and later transferred into `\HomeHTMLFilename` for internal use.

```
\DeclareStringOption[]{HomeHTMLFilename}
```

Opt `HTMLFilename` The filename prefix of web pages after the homepage. The default is empty, no prefix. This option is stored into `\LWR@HTMLFilename`, and later transferred into `\HTMLFilename` for internal use.

```
\DeclareStringOption[]{HTMLFilename}
```

Opt `latexmk` Option `latexmk` tells `lwarpmk` to use `latexmk` when compiling documents.

```
\DeclareBoolOption[false]{latexmk}
```

defaults The default is print output, and SVG math if the user chose HTML output.

```
\includecomment{warpprint}%
\excludecomment{warpHTML}%
\booltrue{warpingprint}%
\boolfalse{warpingHTML}%
\boolfalse{mathjax}%
```

Optionally generate a local copy of `lwarpmk`. Default to no:

```
\excludecomment{LWR@createlwarpmk}
```

Execute options

Execute the package options, with the defaults which have been set just above:

```
\ProcessKeyvalOptions*\relax
```

Assign the `\BaseJobname` if the user hasn't provided one:

```
\providecommand*{\BaseJobname}{\LWR@BaseJobname}
```

Defaults unless already over-ridden by the user:

```
\ifcsempty{LWR@HomeHTMLFilename}{
\newcommand*{\HomeHTMLFilename}{\BaseJobname}
}{
\csedef{HomeHTMLFilename}{\LWR@HomeHTMLFilename}
}

\csedef{HTTMLFilename}{\LWR@HTMLFilename}
```

19 Misplaced packages

Several packages should only be loaded before lwarp, and others should only be loaded after.

Packages which should only be loaded before lwarp have their own

```
lwarp-<packagename>.sty
```

which will trigger an error if they are loaded after lwarp. Examples include fontspec, inputenc, fontenc, and newunicodechar.

`\LWR@loadafter` {⟨*packagename*⟩} Error if this package was loaded before lwarp.

```
\newcommand*{\LWR@loadafter}[1]{%
\@ifpackageloaded{#1}
{
\PackageError{lwarp}
{Package #1, or one which uses #1, must be loaded after lwarp}
```

```
{Move \detokenize{\usepackage}{#1} after \detokenize{\usepackage}{lwarp}.
Package #1 may also be loaded by something else, which must also be moved
after lwarp.}
}
{}
}
```

`\LWR@loadbefore` {⟨*packagename*⟩} Error if this package is after lwarp.

```
\newcommand*{\LWR@loadbefore}[1]{%
\PackageError{lwarp}
{Package #1 must be loaded before lwarp}
{Move \detokenize{\usepackage}{#1} before \detokenize{\usepackage}{lwarp}.}
}
```

`\LWR@loadnever` {⟨*badpackagename*⟩} {⟨*replacementpkgname*⟩}

The first packages is not supported, so tell the user to use the second instead.

```
\newcommand*{\LWR@loadnever}[2]{%
\PackageError{lwarp}
{Package #1 does not work with lwarp's HTML conversion.
Please use the #2 package instead}
{Package #1 conflicts with lwarp in some way, but package #2 probably will work instead.}
}
```

Packages which should only be loaded after lwarp are tested here to trip an error of they have already been loaded.

The following packages must be loaded after lwarp:

```
\LWR@loadafter{abstract}
\LWR@loadafter{afterpage}
\LWR@loadafter{algorithmicx}
\LWR@loadafter{alltt}
\LWR@loadafter{amsthm}
\LWR@loadafter{bookmark}
\LWR@loadafter{booktabs}
\LWR@loadafter{ccaption}
\LWR@loadafter{changepage}
\LWR@loadafter{cutwin}
\LWR@loadafter{dcolumn}
\LWR@loadafter{draftwatermark}
\LWR@loadafter{ellipsis}
\LWR@loadafter{emptypage}
\LWR@loadafter{enumerate}
\LWR@loadafter{epigraph}
\LWR@loadafter{eso-pic}
```

```
\LWR@loadafter{everypage}
\LWR@loadafter{extramarks}
\LWR@loadafter{fancyhdr}
\LWR@loadafter{floatrow}
\LWR@loadafter{float}
\LWR@loadafter{floatflt}
\LWR@loadafter{ftnright}
\LWR@loadafter{geometry}
\LWR@loadafter{glossaries}
% \LWR@loadafter{graphics}% pre-loaded by xunicode
% \LWR@loadafter{graphicx}% pre-loaded by xunicode
\LWR@loadafter{hyperref}
\LWR@loadafter{indentfirst}
\LWR@loadafter{keyfloat}
\LWR@loadafter{layout}
\LWR@loadafter{letterspace}
\LWR@loadafter{lettrine}
\LWR@loadafter{lips}
\LWR@loadafter{listings}
\LWR@loadafter{longtable}
\LWR@loadafter{lscape}
\LWR@loadafter{ltcaption}
\LWR@loadafter{marginfix}
\LWR@loadafter{marginnote}
\LWR@loadafter{mcaption}
\LWR@loadafter{mdframed}
\LWR@loadafter{microtype}
\LWR@loadafter{mparhack}
%\LWR@loadafter{multicol}% loaded by ltxdoc
\LWR@loadafter{multirow}
\LWR@loadafter{nameref}
\LWR@loadafter{needspace}
\LWR@loadafter{newtxmath}
\LWR@loadafter{nextpage}
\LWR@loadafter{nowidow}
\LWR@loadafter{ntheorem}
\LWR@loadafter{pagenote}
\LWR@loadafter{parskip}
\LWR@loadafter{placeins}
\LWR@loadafter{ragged2e}
\LWR@loadafter{rotating}
\LWR@loadafter{setspace}
\LWR@loadafter{showidx}
\LWR@loadafter{showkeys}
\LWR@loadafter{sidecap}
\LWR@loadafter{sidenotes}
\LWR@loadafter{soul}
\LWR@loadafter{subfig}
\LWR@loadafter{tabularx}
\LWR@loadafter{tabulary}
```

```
\LWR@loadafter{textpos}
\LWR@loadafter{theorem}
\LWR@loadafter{threeparttable}
\LWR@loadafter{tikz}
\LWR@loadafter{titleps}
\LWR@loadafter{titlesec}
\LWR@loadafter{titletoc}
\LWR@loadafter{tocloft}
\LWR@loadafter{trivfloat}
\LWR@loadafter{ulem}
\LWR@loadafter{varioref}
\LWR@loadafter{verse}
\LWR@loadafter{wallpaper}
\LWR@loadafter{wrapfig}
\LWR@loadafter{xcolor}
\LWR@loadafter{xfrac}
```

20 Required packages

These packages are automatically loaded by lwarp when generating HTML output. Some of them are also automatically loaded when generating print output, but some are not.

In the document preamble, create a `warpprint` environment, and place inside it any of the following packages which are required and which are labeled as "Print: OK to Load in a `warpprint` environment". Those packages which are labeled as "Print: Pre-Loaded" need not be placed into the document preamble.

for HTML & PRINT:

```
\begin{warpall}
```

See: http://tex.stackexchange.com/a/47579.

Detects XeTeX and LuaLaTeX:

```
\RequirePackage{iftex}
\newif\ifxetexorluatex
\ifXeTeX
    \xetexorluatextrue
\else
    \ifLuaTeX
        \xetexorluatextrue
    \else
        \xetexorluatexfalse
    \fi
\fi

\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

```
\ifxetexorluatex
% ^^A     \usepackage[no-math]{fontspec}
```

The monospaced font is used for HTML tags, so turn off its TeX ligatures and common ligatures:

```
\defaultfontfeatures[\rmfamily]{Ligatures={NoCommon,TeX}}
\defaultfontfeatures[\sffamily]{Ligatures={NoCommon,TeX}}
\defaultfontfeatures[\ttfamily]{Ligatures=NoCommon}
\else
```

`pdflatex` only: Only pre-loaded if `pdflatex` is being used.

Pkg `microtype`

ligatures Older browsers don't display ligatures. Turn off letter ligatures, keeping LaTeX dash and quote ligatures, which may fail on older browers but at least won't corrupt written words.

```
\RequirePackage {microtype}

\microtypesetup{
protrusion=false,
expansion=false,
tracking=false,
kerning=false,
spacing=false}

\DisableLigatures[f,q,t,T,Q]{encoding = *,family = *}
```

```
\fi
```

```
\end{warpHTML}
```

Pkg `geometry` Tactics to avoid unwanted page breaks and margin overflow:

- Uses a very long and wide page to minimize page breaks and margin overflow.
- Uses a scriptsize font.
- Uses extra space at the margin to avoid HTML tag overflow off the page.
- Forces a new PDF page before some environments.
- Forces line break between major pieces of long tags.

for HTML output:

```
\begin{warpHTML}
\RequirePackage[paperheight=190in,paperwidth=20in,%
left=2in,right=12in,%
top=1in,bottom=1in,%
]{geometry}
\@twosidefalse
\@mparswitchfalse
\end{warpHTML}
```

for HTML & PRINT:

```
\begin{warpall}
```

Pkg `xparse`

LaTeX3 command argument parsing

```
\RequirePackage{xparse}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

Pkg `expl3`

LaTeX3 programming

```
\RequirePackage{expl3}
```

Pkg `gettitlestring`

Used to emulate `\nameref`.

```
\RequirePackage{gettitlestring}
```

Pkg `everyhook`

everyhook is used to patch paragraph handling.

```
\RequirePackage{everyhook}
\end{warpHTML}
```

for HTML & PRINT:

```
\begin{warpall}
```

Pkg `fancyvrb`

Used for Verbatim, verse.

```
\RequirePackage{fancyvrb}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

Pkg `xifthen`

```
\RequirePackage{xifthen}
```

Pkg `xstring`

```
\RequirePackage{xstring}
```

Pkg `makeidx`

```
\RequirePackage{makeidx}
\makeindex
```

Pkg `calc`

```
\RequirePackage{calc}
```

Pkg `refcount`

```
\RequirePackage{refcount}
```

Pkg `newfloat`

```
\RequirePackage{newfloat}
```

Pkg `caption`

```
\RequirePackage{caption}
```

Pkg `enumitem`

enumitem is patched to support `\newlist` with HTML.

```
\RequirePackage{enumitem}
\setlist[itemize]{leftmargin=0em}
\setlist[enumerate]{leftmargin=0em}
\setlist[description]{leftmargin=0em}
```

```
\end{warpHTML}
```

for HTML & PRINT:

```
\begin{warpall}
```

Pkg `titling`

Used for `\maketitle` and the title page. See section 48.

```
\RequirePackage{titling}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

Pkg **zref**

Used for cross-references.

```
\RequirePackage{zref}
```

Pkg **amsmath**

Equation numbers are placed to the left for HTML.

newtxmath automatically loads amsmath, so the options `leqno` and `fleqn` are passed beforehand to be picked up both here and by newtxmath if it is used.

```
\PassOptionsToPackage{leqno,fleqn}{amsmath}
\RequirePackage{amsmath}
```

Pkg **environ**

Used to encapsulate math environments for re-use in HTML ALT text.

```
\RequirePackage{environ}
```

Pkg **titleps**

Used to place an HTML comment into the footer of a page below the footnotes. This comment is used for `lateximage` environments, including math.

The `nopatches` option prevents titleps from trying to patch sectioning commands.

`\pagestyle` and `\thispagestyle` are nullified for HTML output.

```
\RequirePackage[nopatches]{titleps}
```

`\pagestyle` {⟨*style*⟩}

```
\let\LWR@origpagestyle\pagestyle
\renewcommand*{\pagestyle}[1]{}
```

`\thispagestyle` {⟨*style*⟩}

```
\let\LWR@origthispagestyle\thispagestyle
\renewcommand*{\thispagestyle}[1]{}
```

\pagenumbering {⟨*commands*⟩}

```
\let\LWR@origpagenumbering\pagenumbering
\renewcommand*{\pagenumbering}[1]{}
```

Pkg xfrac

Patched for HTML use. See section 165.

```
\RequirePackage{xfrac}
```

Used to convert lengths for image width/height options.

```
\RequirePackage{printlen}
```

```
\end{warpHTML}
```

21 Loading packages

for HTML output:

```
\begin{warpHTML}
```

Remember the original \RequirePackage:

```
\let\LWR@origRequirePackage\RequirePackage
```

\LWR@requirepackagenames Stores the list of required package names.

```
\newcommand*{\LWR@requirepackagenames}{}
```

\LWR@findword [⟨*1: separator*⟩] {⟨*2: list*⟩} {⟨*3: index*⟩} [⟨*4: destination*⟩]

Note that argument 4 is passed directly to \StrBetween.

```
\newcommand*\LWR@findword[3][,]{%
    \StrBetween[#3,\numexpr#3+1]{#1#2#1}{#1}{#1}%
}
```

\LWR@lookforpackagename {⟨*index*⟩} If this is a package name, re-direct it to the lwarp version by renaming it lwarp- followed by the original name.

```
\newcommand*{\LWR@lookforpackagename}[1]{
```

Find the n'th package name from the list:

```
\LWR@findword{\LWR@requirepackagenames}{#1}[\LWR@strresult]
```

See if the package name was found:

```
\IfStrEq{\LWR@strresult}{}
{}% no filename
{% yes filename
```

If found, and if an lwarp-equivalent name exists, use `lwarp-*` instead.

```
\IfFileExists{lwarp-\LWR@strresult.sty}
{% latex_html_ file found
\StrSubstitute
{\LWR@requirepackagenames}
{\LWR@strresult}
{lwarp-\LWR@strresult}[\LWR@requirepackagenames]
}
{}% no latex_html_* file
}% yes filename
}
```

`\RequirePackage` [⟨*1: options*⟩] {⟨*2: package names*⟩} [⟨*3: version*⟩]

For each of many package names in a comma-separated list, if an lwarp version of a package exists, select it instead of the LaTeX version.

```
\RenewDocumentCommand{\RequirePackage}{o m o}{%
```

Redirect up to nine names:

```
\renewcommand*{\LWR@requirepackagenames}{#2}
\LWR@lookforpackagename{1}
\LWR@lookforpackagename{2}
\LWR@lookforpackagename{3}
\LWR@lookforpackagename{4}
\LWR@lookforpackagename{5}
\LWR@lookforpackagename{6}
\LWR@lookforpackagename{7}
\LWR@lookforpackagename{8}
\LWR@lookforpackagename{9}
```

`\RequirePackage` depending on the options and version:

```
\IfValueTF{#1}
{% options given
\IfValueTF{#3}% version given?
```

```
{\LWR@origRequirePackage[#1]{\LWR@requirepackagenames}[#3]}
{\LWR@origRequirePackage[#1]{\LWR@requirepackagenames}}
}
{% no options given
\IfValueTF{#3}% version given?
{\LWR@origRequirePackage{\LWR@requirepackagenames}[#3]}
{\LWR@origRequirePackage{\LWR@requirepackagenames}}
}
}
\let\usepackage\RequirePackage
```

`\LWR@ProvidesPackagePass` {⟨*pkgname*⟩} [⟨*version*⟩]

Uses the original package, including options.

```
\NewDocumentCommand{\LWR@ProvidesPackagePass}{m o}{
\PackageInfo{lwarp}{Using package ‘#1’ and adding lwarp modifications, including options,}%
\IfValueTF{#2}
{\ProvidesPackage{lwarp-#1}[#2]}
{\ProvidesPackage{lwarp-#1}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{#1}}
\ProcessOptions\relax

\IfValueTF{#2}
{\LWR@origRequirePackage{#1}[#2]}
{\LWR@origRequirePackage{#1}}
}
```

`\LWR@ProvidesPackageDrop` {⟨*pkgname*⟩} [⟨*version*⟩]

Ignores the original package and uses lwarp's version instead. Drops/discards all options.

```
\NewDocumentCommand{\LWR@ProvidesPackageDrop}{m o}{
\PackageInfo{lwarp}{Replacing package ‘#1’ with the lwarp version, discarding options,}%
\IfValueTF{#2}
{\ProvidesPackage{lwarp-#1}[#2]}
{\ProvidesPackage{lwarp-#1}}
\DeclareOption*{}
\ProcessOptions\relax
}
```

```
\end{warpHTML}
```

22 Copying a file

for HTML output:

```
\begin{warpHTML}
```

`\LWR@copyfile` {⟨*source filename*⟩} {⟨*destination filename*⟩}

Used to copy the `.toc` file to `.sidetoc` to re-print the TOC in the sideTOC navigation pane.

```
\newcommand*{\LWR@copyfile}[2]{%
\newwrite\copyfile  % open the file to write to
\immediate\openout\copyfile=#2
\newread\file      % open the file to read from
\openin\file=#1
\begingroup\endlinechar=-1
\makeatletter
\loop\unless\ifeof\file
  \read\file to\fileline  % Read one line and store it into \fileline
%   \fileline\par                  % print the content into the pdf
% print the content:
  \immediate\write\copyfile{\unexpanded\expandafter{\fileline}}%
\repeat
\closeout\copyfile
\endgroup
}
```

```
\end{warpHTML}
```

23 Debugging messages

```
\begin{warpall}
```

Bool `LWR@tracinglwarp` True if tracing is turned on.

```
\newbool{LWR@tracinglwarp}
```

`\tracinglwarp` Turns on the debug tracing messages.

```
\newcommand{\tracinglwarp}{\booltrue{LWR@tracinglwarp}}
```

`\LWR@traceinfo` {⟨*text*⟩} If tracing is turned on, writes the text to the `.log` file.

```
\newcommand{\LWR@traceinfo}[1]{%
\ifbool{LWR@tracinglwarp}%
{%
```

```
\typeout{*** lwarp: #1}%
% \PackageInfo{lwarp}{#1 : }%
}%
{}%
}
```

Bool `HTMLDebugComments` Default false. Add comments in HTML about closing `<div>`s, sections, etc.

```
\newbool{HTMLDebugComments}
\boolfalse{HTMLDebugComments}
```

24 HTML-conversion output modifications

These booleans modify the HTML output in various ways to improve conversion to EPUB or word processor imports.

Bool `FormatEPUB` Default false. Changes HTML output for easy EPUB conversion via an external program. Removes per-file headers, footers, and nav. Adds footnotes per chapter/section.

```
\newbool{FormatEPUB}
\boolfalse{FormatEPUB}
```

Bool `FormatWordProcessor` Default false. Changes HTML output for easier conversion by a word processor. Removes headers and nav, prints footnotes per section, and also forces single-file output and turns off HTML debug comments.

```
\newbool{FormatWordProcessor}
\boolfalse{FormatWordProcessor}
```

Bool `HTMLMarkFloats` Default true. Adds `=== table begin` or `=== figure begin`, and `=== end` around floats while formatting for word processors. This helps identify boundaries of floats to be manually converted to word-processor frames. (Perhaps some day word processors will have HTML import options for identifying `<div>` classes for figures and tables.)

```
\newbool{HTMLMarkFloats}
\booltrue{HTMLMarkFloats}
```

```
\end{warpall}
```

25 Remembering original formatting macros

for HTML output:

```
\begin{warpHTML}
```

Remember original definitions of formatting commands. Will be changed to HTML commands for most uses. Will be temporarily restored to original meaning inside any lateximage environment. Also nullify unused commands.

```
\let\LWR@origtextit\textit
\let\LWR@origtextbf\textbf
\let\LWR@origtexttt\texttt
\let\LWR@origtextsc\textsc
\let\LWR@origtextsf\textsf
\let\LWR@origtextrm\textrm
\let\LWR@origbfseries\bfseries
\let\LWR@origrmfamily\rmfamily
\let\LWR@origttfamily\ttfamily
\let\LWR@orignormalfont\normalfont

\let\LWR@origraggedright\raggedright
\let\LWR@origonecolumn\onecolumn

\let\LWR@origtextsuperscript\textsuperscript
\let\LWR@origtextsubscript\textsubscript

\let\LWR@origscriptsize\scriptsize

\let\LWR@orignewpage\newpage

\let\LWR@origminipage\minipage
\let\LWR@origendminipage\endminipage

\let\LWR@orignewline\newline

\let\LWR@origitem\item

\let\LWR@origpar\par


\let\LWR@origfootnote\footnote
\let\LWR@orig@mpfootnotetext\@mpfootnotetext

\let\LWR@origclearpage\clearpage
\let\clearpage\relax
\let\cleardoublepage\relax

\end{warpHTML}
```

26 Configuration Files

```
\begin{warpprint}
\typeout{lwarp: generating configuration files}
\end{warpprint}
```

26.1 project_html.tex

File `project_html.tex` Used to allow an HTML version of the document to exist alongside the print version.

Only write `\jobname_html.tex` if generating the print version.

```
\begin{warpprint}
\ifcsdef{LWR@file}{}{\newwrite{\LWR@file}}
\immediate\openout\LWR@file=\jobname_html.tex
\immediate\write\LWR@file{%
\detokenize{\PassOptionsToPackage}%
{warpHTML,BaseJobname=\jobname}{lwarp}%
}
\immediate\write\LWR@file{%
\detokenize{\input}\string{\jobname.tex\string }%
}
\immediate\closeout\LWR@file
\end{warpprint}
```

26.2 lwarpmk.conf

File `lwarpmk.conf` `lwarpmk.conf` is automatically (re-)created by the lwarp package when executing
`pdflatex <project.tex>`,
or similar for `xelatex` or `lualatex`, in print-document generation mode, which is the default unless the `warpHTML` option is given. `lwarpmk.conf` is then used by the utility `lwarpmk`.

An example `lwarpmk.conf`:

```
opsystem = "Unix"   -- or "Windows"
latexname = "pdflatex"   -- or "lualatex" or "xelatex"
sourcename = "projectname"   -- your .tex source
homehtmlfilename = "index"   -- or "projectname"
htmlfilename = ""   -- or "projectname" if numbered HTML files
```

for PRINT output:

```
\begin{warpprint}
\ifcsdef{LWR@file}{}{\newwrite{\LWR@file}}
\immediate\openout\LWR@file=lwarpmk.conf
\ifbool{usingOSWindows}{
```

```
\immediate\write\LWR@file{opsystem = "Windows"}
}{
\immediate\write\LWR@file{opsystem = "Unix"}
}
\ifPDFTeX
\immediate\write\LWR@file{latexname = "pdflatex"}
\fi
\ifXeTeX
\immediate\write\LWR@file{latexname = "xelatex"}
\fi
\ifLuaTeX
\immediate\write\LWR@file{latexname = "lualatex"}
\fi
\immediate\write\LWR@file{sourcename = "\jobname"}
\immediate\write\LWR@file{%
homehtmlfilename = "\HomeHTMLFilename"%
}
\immediate\write\LWR@file{htmlfilename = "\HTMLFilename"}
\immediate\write\LWR@file{latexmk = "\ifbool{LWR@latexmk}{true}{false}"}
\immediate\write\LWR@file{language = "\LWR@IndexLanguage"}
\immediate\write\LWR@file{xdyfile = "\LWR@xdyFilename"}
\immediate\closeout\LWR@file
\end{warpprint}
```

26.3 project.lwarpmkconf

File `project.lwarpmkconf` A project-specific configuration file for `lwarpmk`.

```
\begin{warpprint}
\ifcsdef{LWR@file}{}{\newwrite{\LWR@file}}
\immediate\openout\LWR@file=\jobname.lwarpmkconf
\ifbool{usingOSWindows}{
\immediate\write\LWR@file{opsystem = "Windows"}
}{
\immediate\write\LWR@file{opsystem = "Unix"}
}
\ifPDFTeX
\immediate\write\LWR@file{latexname = "pdflatex"}
\fi
\ifXeTeX
\immediate\write\LWR@file{latexname = "xelatex"}
\fi
\ifLuaTeX
\immediate\write\LWR@file{latexname = "lualatex"}
\fi
\immediate\write\LWR@file{sourcename = "\jobname"}
\immediate\write\LWR@file{%
homehtmlfilename = "\HomeHTMLFilename"%
```

```
}
\immediate\write\LWR@file{htmlfilename = "\HTMLFilename"}
\immediate\write\LWR@file{latexmk = "\ifbool{LWR@latexmk}{true}{false}"}
\immediate\write\LWR@file{language = "\LWR@IndexLanguage"}
\immediate\write\LWR@file{xdyfile = "\LWR@xdyFilename"}
\immediate\closeout\LWR@file
\end{warpprint}
```

26.4 lwarp.css

File `lwarp.css` This is the base CSS layer used by lwarp.

This must be present both when compiling the project and also when distributing the HTML files.

```
\begin{warpprint}
\begin{VerbatimOut}{lwarp.css}
/*
  CSS stylesheet for the LaTeX lwarp package
  Copyright 2016-2017 Brian Dunn -- BD Tech Concepts LLC
*/


/* a fix for older browsers: */
header, section, footer, aside, nav, main,
    article, figure { display: block; }


A:link {color:#000080 ; text-decoration: none ; }
A:visited {color:#800000 ; }
A:hover {color:#000080 ; text-decoration: underline ;}
A:active {color:#800000 ; }

a.tocpart {display: inline-block ; margin-left: 0em ;
    font-weight: bold ;}
a.tocchapter {display: inline-block ; margin-left: 0em ;
    font-weight: bold ;}
a.tocsection {display: inline-block ; margin-left: 1em ;
    text-indent: -.5em ; font-weight: bold ; }
a.tocsubsection {display: inline-block ; margin-left: 2em ;
    text-indent: -.5em ; }
a.tocsubsubsection {display: inline-block ; margin-left: 3em ;
    text-indent: -.5em ; }
a.tocparagraph {display: inline-block ; margin-left: 4em ;
    text-indent: -.5em ; }
a.tocsubparagraph {display: inline-block ; margin-left: 5em ;
    text-indent: -.5em ; }
```

```
a.tocfigure {margin-left: 0em}
a.tocsubfigure {margin-left: 2em}
a.toctable {margin-left: 0em}
a.tocsubtable {margin-left: 2em}
a.toctheorem {margin-left: 0em}
a.toclstlisting {margin-left: 0em}


body {
    font-family: "DejaVu Serif", "Bitstream Vera Serif",
        "Lucida Bright", Georgia, serif;
    background: #FAF7F4 ;
    color: black ;
    margin:0em ;
    padding:0em ;
    font-size: 100% ;
    line-height: 1.2 ;
}

p {margin: 1.5ex 0em 1.5ex 0em ;}

/* Holds a section number to add space between it and the name */
span.sectionnumber { margin-right: .6em }

/* Inserted in front of index lines */
span.indexitem {margin-left: 0em}
span.indexsubitem {margin-left: 2em}
span.indexsubsubitem {margin-left: 4em}

div.hidden { display: none ; }

kbd {
    font-family: "DejaVu Mono", "Bitstream Vera Mono", "Lucida Console",
        "Nimbus Mono L", "Liberation Mono", "FreeMono", "Andale Mono",
        "Courier New", monospace;
    font-size: 100% ;
}

span.strong { font-weight: bold; }

span.textmd { font-weight: normal; }

span.textsc { font-variant: small-caps; }

span.textup { font-variant: normal; }

span.textrm {
    font-family: "DejaVu Serif", "Bitstream Vera Serif",
    "Lucida Bright", Georgia, serif;
}
```

```

span.textsf {
     font-family: "DejaVu Sans", "Bitstream Vera Sans",
        Geneva, Verdana, sans-serif ;
}

span.attribution {
  margin-left: 1em ; font-size: 80% ; font-variant: small-caps;
}

span.citetitle {
  margin-left: 1em ; font-size: 80% ; font-style: oblique;
}

span.poemtitle {
  font-size: 120% ; font-weight: bold;
}

blockquote {
margin-left: 0px ;
margin-right: 0px ;
}

blockquote p {
  line-height: 1.5;
    text-align: left ;
    font-size: .85em ;
    margin-left: 3em ;
margin-right: 3em ;
}

blockquotation {
margin-left: 0px ;
margin-right: 0px ;
}

blockquotation p {
  line-height: 1.5;
    text-align: left ;
    font-size: .85em ;
    margin-left: 3em ;
margin-right: 3em ;
}

div.epigraph {
  line-height: 1.2;
    text-align: left ;
    padding: 3ex 1em 0ex 1em ;
/*     margin: 3ex auto 3ex auto ; */ /* Epigraph centered */
    margin: 3ex 1em 3ex auto ; /* Epigraph to the right */
```

```
/*    margin: 3ex 1em 3ex 1em ; */ /* Epigraph to the left */
    font-size: .85em ;
    max-width: 27em ;
}



div.epigraphsource{
    text-align:right ;
    margin-left:auto ;
/*     max-width: 50% ; */
    border-top: 1px solid #A0A0A0 ;
    padding-bottom: 3ex ;
    line-height: 1.2;
}

div.epigraph p { padding: .5ex ; margin: 0ex ;}
div.epigraphsource p { padding: .5ex 0ex 0ex 0ex ; margin: 0ex ;}


/* lettrine package: */
span.lettrine { font-size: 3ex ; float: left ; }
span.lettrinetext { font-variant: small-caps ; }

/* ulem and soul packages: */
span.uline {
    text-decoration: underline ;
    text-decoration-skip ;
}

span.uuline {
    text-decoration: underline ;
    text-decoration-skip ;
    text-decoration-style: double ;
}

span.uwave {
    text-decoration: underline ;
    text-decoration-skip ;
    text-decoration-style: wavy ;
}

span.sout {
    text-decoration: line-through ;
}

span.xout {
    text-decoration: line-through ;
}

```

```
span.dashuline {
    text-decoration: underline ;
    text-decoration-skip ;
    text-decoration-style: dashed ;
}

span.dotuline {
    text-decoration: underline ;
    text-decoration-skip ;
    text-decoration-style: dotted ;
}

span.letterspacing { letter-spacing: .2ex ; }

span.capsspacing {
    font-variant: small-caps ;
    letter-spacing: .1ex ;
}

span.highlight { background: #F8E800 ; }



html body {
  margin: 0 ;
  line-height: 1.2;
}


body div {
  margin: 0ex;
}


h1, h2, h3, h4, h5, h6, span.paragraph, span.subparagraph
{
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    font-style: normal ;
    font-weight: bold ;
    text-align: left ;
}

h1 {    /* title of the entire website, used on each page */
    text-align: center ;
    font-size: 2.5em ;
    padding: .4ex 0em 0ex 0em ;
```

```
}
h2 { font-size: 2.25em }
h3 { font-size: 2em }
h4 { font-size: 1.75em }
h5 { font-size: 1.5em }
h6 { font-size: 1.25em }
span.paragraph {font-size: 1em ; font-variant: normal ;
    margin-right: 1em ; }
span.subparagraph {font-size: 1em ; font-variant: normal ;
    margin-right: 1em ;  }



/* Title of the file */
h1 {
  margin: 0ex 0em 0ex 0em ;
  line-height: 1.3;
  text-align: center ;
}

/* Part */
h2 {
  margin: 1ex 0em 1ex 0em ;
  line-height: 1.3;
  text-align: center ;
}

/* Chapter */
h3 {
  margin: 3ex 0em 1ex 0em ;
  line-height: 1.3;
}

/* Section */
h4 {
  margin: 3ex 0em 1ex 0em ;
  line-height: 1.3;
}

/* Sub-Section */
h5 {
  margin: 3ex 0em 1ex 0em ;
  line-height: 1.3;
}

/* Sub-Sub-Section */
h6 {
  margin: 3ex 0em 1ex 0em ;
  line-height: 1.3;
}
```

```


div.titlepage {
  text-align: center ;
}

.footnotes {
    font-size: .85em ;
    margin: 3ex 1em 0ex 1em ;
    padding-bottom: 1ex ;
border-top: 1px solid silver ;
}

.marginpar {
    max-width:50%;
    float:right;
    text-align:left;
    margin: 1ex 0.5em 1ex 1em ;
    padding: 1ex 0.5em 1ex 0.5em ;
    font-size: 85% ;
    border-top: 1px solid silver ;
    border-bottom: 1px solid silver ;
    overflow-x: auto;
}

.marginpar br { margin-bottom: 2ex ; }

div.marginblock {
    max-width:50%;
    float:right;
    text-align:left;
    margin: 1ex 0.5em 1ex 1em ;
    padding: 1ex 0.5em 1ex 0.5em ;
    overflow-x: auto;
}

div.marginblock div.minipage {
    display: block ;
    margin: 0pt auto 0pt auto ;
}

div.marginblock div.minipage p { font-size: 85%}

div.marginblock br { margin-bottom: 2ex ; }


section.textbody div.footnotes{
    margin: 3ex 0em 0ex 0em ;
    border-bottom: 2px solid silver ;
}
```

```

.footnoteheader {
    border-top: 2px solid silver ;
    margin-top: 3ex ;
    padding-top: 1ex ;
    font-weight: bold ;
}

.mpfootnotes {
    text-align: left ;
    font-size: .85em ;
    margin-left: 1em ;
    border-top: 1px solid silver ;
}

/* Remove footnote top border in the title page. */
div.titlepage div.mpfootnotes {
    border-top: none ;
}



ol {
  margin: 1ex 1em 1ex 0em;
  line-height: 1.2;
}

ul, body dir, body menu {
  margin: 1ex 1em 1ex 0em;
  line-height: 1.2;
}

li { margin: 0ex 0em 1ex 0em; }

html {
  margin: 0;
  padding: 0;
}

.programlisting {
  font-family: "DejaVu Mono", "Bitstream Vera Mono", "Lucida Console",
        "Nimbus Mono L", "Liberation Mono", "FreeMono", "Andale Mono",
        "Courier New", monospace;
  margin: 1ex 0ex 1ex 0ex ;
  padding: .5ex 0pt .5ex 0pt ;
  overflow-x: auto;
}

section.textbody>pre.programlisting {
border-top: 1px solid silver ;
```

```
border-bottom: 1px solid silver ;
}


.inlineprogramlisting {
  font-family: "DejaVu Mono", "Bitstream Vera Mono", "Lucida Console",
        "Nimbus Mono L", "Liberation Mono", "FreeMono", "Andale Mono",
        "Courier New", monospace;
  overflow-x: auto;
}


div.abstract {
  margin: 2em 5% 2em 5% ;
  padding: 1ex 1em 1ex 1em ;
/*  font-weight: bold ; */
  font-size: 90% ;
}

div.abstract dl {line-height:1.5;}
div.abstract dt {color:#304070;}

div.abstracttitle{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
    font-size:1.25em;
    text-align: center ;
}

span.abstractrunintitle{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
}


.verbatim {
    overflow-x: auto ;
}

.alltt {
    overflow-x: auto ;
}


.bverbatim {
    margin: 1ex 0pt 1ex 0pt ;
```

```
    padding: .5ex 0pt .5ex 0pt ;
    overflow-x: auto ;
}

.lverbatim {
    margin: 1ex 0pt 1ex 0pt ;
    padding: .5ex 0pt .5ex 0pt ;
    overflow-x: auto ;
}

.fancyvrb {
    font-size:.85em ;
    margin: 3ex 0pt 3ex 0pt
}

.fancyvrblabel {
    font-weight:bold;
    text-align: center ;
}


.verse {
    font-family: "Linux Libertine Mono O", "Lucida Console",
        "Droid Sans Mono", "DejaVu Mono", "Bitstream Vera Mono",
        "Liberation Mono", "FreeMono", "Andale Mono",
        "Nimbus Mono L", "Courier New", monospace;
    margin-left: 1em ;
}


div.singlespace { line-height: 1.2 ; }
div.onehalfspace { line-height: 1.5 ; }
div.doublespace { line-height: 2 ; }




/* Minipage environments, vertically aligned to top, center, bottom: */
.minipage {
    /* display: inline-block ; */
        /* Mini pages which follow each other will be tiled. */
    margin: .25em .25em .25em .25em;
    padding: .25em .25em .25em .25em;
    display: inline-flex;
    flex-direction: column ;
    overflow: auto;
}

/* Paragraphs in the flexbox did not collapse their margins. */
```

```
/* Have not yet researched this. */
.minipage p {margin: .75ex 0em .75ex 0em ;}



.framebox {
    margin: 0ex ;
    padding: 0ex ;
   border: 1px solid black;
     border-radius: 0px ;
    padding: .3ex .2em 0ex .2em ;
    margin: .1ex ;
  display: inline-block ;
}


.mdframed {
/*     padding: 0ex ; */
/*    border: 1px solid blafck; */
/*      border-radius: 0px ; */
    padding: 0ex ;
    margin: 3ex 5% 3ex 5% ;
/*   display: inline-block ; */
}

.mdframed p { padding: 0ex .5em 0ex .5em ; }

.mdframed dl { padding: 0ex .5em 0ex .5em ; }

.mdframedtitle {
    padding: .5em ;
    display: block ;
    font-size: 130%
}

.mdframedsubtitle {
    padding: 0ex .5em 0ex .5em ;
    display: block ;
    font-size: 115% ;
}

.mdframedsubsubtitle {
    padding: 0ex .5em 0ex .5em ;
    display: block ;
}

.mdtheorem {
    padding: 0ex .5em 0ex .5em ;
    margin: 3ex 5% 3ex 5% ;
/*   display: inline-block ; */
```

```
}


/* framed package */
.framed {
    margin: 3ex 0em 3ex 0em ;
   border: 1px solid black;
     border-radius: 0px ;
    padding: .3ex 1em 0ex 1em ;
  display: block ;
}

.snugframed {
    margin: 3ex 0em 3ex 0em ;
   border: 1px solid black;
     border-radius: 0px ;
  display: block ;
}

.framedleftbar {
    margin: 3ex 0em 3ex 0em ;
   border-left: 3pt solid black;
     border-radius: 0px ;
    padding: .3ex .2em .3ex 1em ;
  display: block ;
}

.framedtitle {
margin: 0em ;
padding: 0em ;
    font-size: 130%
}

.framedtitle p { padding: .3em }



dl {
  margin: 1ex 2em 1ex 0em;
  line-height: 1.3;
}

dl dt {
  margin-top: 1ex;
    font-weight: bold;
}

dl dd p { margin-top: 0em; }


```

```
nav.toc, nav.lof, nav.lot, nav.lol, nav.lothm {
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "DejaVu Sans", "Bitstream Vera Sans",
        Geneva, Verdana, sans-serif ;
    margin-bottom: 4ex ;
}

nav.toc p, nav.lof p, nav.lot p, nav.lol p, nav.lothm p {
    line-height: 1.2 ;
    margin-top:.5ex ;
    margin-bottom:.5ex;
    font-size: .9em ;
}



img, img.hyperimage, img.borderimage {
    max-width: 600px;
    border: 1px solid silver;
    box-shadow: 3px 3px 3px #808080 ;
    padding: .5% ;
    margin: .5% ;
    background: none ;
}

img.inlineimage{
    padding: 0px ;
    box-shadow: none ;
    border: none ;
    background: none ;
    margin: 0px ;
    display: inline-block ;
    border-radius: 0px ;
}

img.logoimage{
    max-width: 300px ;
    box-shadow: 3px 3px 3px #808080 ;
    border: 1px solid black ;
    background:none ;
    padding:0 ;
    margin:.5ex ;
    border-radius: 10px ;
}


.section {
/*
    To have each section float relative to each other:
*/
```

```
/*
    display: block ;
    float: left ;
    position: relative ;
    background: white ;
    border: 1px solid silver ;
    padding: .5em ;
*/
    margin: 0ex .5em 0ex .5em ;
    padding: 0 ;
}


figure {
    margin: 3ex auto 3ex auto ;
    padding: 1ex 1em 1ex 1em ;
    overflow-x: auto ;
}


/* To automatically center images in figures: */
/*
figure img.inlineimage {
    margin: 0ex auto 0ex auto ;
    display: block ;
}
*/

/* To automatically center minipages in figures: */
/*
figure div.minipage, figure div.minipage div.minipage {
    margin: 1ex auto 1ex auto ;
    display: block ;
}
*/

figure div.minipage p { font-size: 85% ; }

figure.subfigure, figure.subtable  {
    display: inline-block ; margin: 3ex 1em 3ex 1em ;
}

figcaption .minipage { margin:0 ; padding: 0 }

div.floatrow { text-align: center; }

div.floatrow figure { display: inline-block ; margin: 1ex 2% ; }

div.floatfoot { font-size: .85em ;
    border-top: 1px solid silver ; line-height: 1.2 ; }
```

```
figcaption , .lstlistingtitle {
    font-size: .85em ;
    text-align: center ;
    font-weight: bold ;
margin-top: 1ex ;
margin-bottom: 1ex ;
}

figure.subfigure figcaption, figure.subtable figcaption  {
    border-bottom: none ; background: none ;
}

div.nonfloatcaption {
    margin: 1ex auto 1ex auto ;
    font-size: .85em ;
    text-align: center ;
    font-weight: bold ;
}

/* For a \RawCaption inside a minipage inside a figure's floatrow: */
figure div.floatrow div.minipage figcaption {
border: none ;
background: none ;
}


table {
    margin: 1ex auto 1ex auto ;
    border-collapse: collapse ;
    border-spacing: 0px ;
    line-height: 1.3 ;
    }

tr.hline {border-top: 1px solid silver ; margin-top: 0ex ;
    margin-bottom: 0ex ; } /* for \hline */

tr.tbrule {border-top: 1px solid black ; margin-top: 0ex ;
    margin-bottom: 0ex ; } /* for \toprule, \bottomrule */

td {padding: 1ex .5em 1ex .5em ;}

table td.tdl { text-align: left ; vertical-align: middle ; }
table td.tdc { text-align: center ; vertical-align: middle ; }
table td.tdr { text-align: right ; vertical-align: middle ; }
table td.tdp { text-align: left ; vertical-align: bottom ; }
table td.tdm { text-align: left ; vertical-align: middle ; }
table td.tdb { text-align: left ; vertical-align: top ; }
table td.tdP { text-align: center ; vertical-align: bottom ; }
table td.tdM { text-align: center ; vertical-align: middle ; }
```

```
table td.tdB { text-align: center ; vertical-align: top ; }
table td.tdlrule { text-align: left ; border-top: 1px solid silver ;
    vertical-align: middle ; } /* for cmidrule */
table td.tdcrule { text-align: center ; border-top: 1px solid silver ;
    vertical-align: middle ; }
table td.tdrrule { text-align: right ; border-top: 1px solid silver ;
    vertical-align: middle ; }
table td.tdprule { text-align: left ; border-top: 1px solid silver ;
    vertical-align: bottom ; }
table td.tdmrule { text-align: left ; border-top: 1px solid silver ;
    vertical-align: middle ; }
table td.tdbrule { text-align: left ; border-top: 1px solid silver ;
    vertical-align: top ; }
table td.tdPrule { text-align: center ; border-top: 1px solid silver ;
    vertical-align: bottom ; }
table td.tdMrule { text-align: center ; border-top: 1px solid silver ;
    vertical-align: middle ; }
table td.tdBrule { text-align: center ; border-top: 1px solid silver ;
    vertical-align: top ; }

/* Margins of paragraphs inside table cells: */
td.tdp p , td.tdprule p , td.tdP p , td.tdPrule p { padding-top: 1ex ;
    padding-bottom: 1ex ; margin: 0ex ; }
td.tdm p , td.tmbrule p , td.tdM p , td.tdMrule p { padding-top: 1ex ;
    padding-bottom: 1ex ; margin: 0ex ; }
td.tdb p , td.tdbrule p , td.tdB p , td.tdBrule p { padding-top: 1ex ;
    padding-bottom: 1ex ; margin: 0ex ; }

td.tdp , td.tdprule , td.tdP , td.tdPrule
    { padding: 0ex .5em 0ex .5em ; }
td.tdm , td.tdmrule , td.tdM , td.tdMrule
    { padding: 0ex .5em 0ex .5em ; }
td.tdb , td.tdbrule , td.tdB , td.tdBrule
    { padding: 0ex .5em 0ex .5em ; }


/* table notes: */
.tnotes {
    margin: 0ex 5% 1ex 5% ;
    padding: 0.5ex 1em 0.5ex 1em;
    font-size:.85em;
    text-align: left ;
}

.tnotes dl dt p {margin-bottom:0px;}

.tnoteitemheader {margin-right: 1em;}
```

```
/* center, flushleft, flushright environments */
div.center{text-align:center;}
div.center table {margin-left:auto;margin-right:auto;}
div.flushleft{text-align:left;}
div.flushleft table {margin-left:0em ; margin-right:auto;}
div.flushright{text-align:right;}
div.flushright table {margin-left:auto ; margin-right: 0em ;}



/* program listing callouts: */
span.callout {
     font-family: "DejaVu Sans", "Bitstream Vera Sans",
        Geneva, Verdana, sans-serif ;
    border-radius: .5em;
    background-color:black;
    color:white;
    padding:0px .25em 0px .25em;
margin: 0 ;
    font-weight: bold;
    font-size:.72em ;
}

div.programlisting pre.verbatim span.callout{
font-size: .85em ;
}



div.published
{
    text-align: center ;
    font-variant: normal ;
    font-style: italic ;
    font-size: 1em ;
    margin: 3ex 0em 3ex 0em ;
}

div.subtitle
{
    text-align: center ;
    font-variant: normal ;
    font-style: italic ;
    font-size: 1.25em ;
    margin: 3ex 0em 3ex 0em ;
}

```

```
div.subtitle p { margin: 1ex ; }

div.author
{
    font-variant: normal ;
    font-style: normal ;
    font-size: 1em ;
    margin: 3ex 0em 3ex 0em ;
}

div.author table {
    margin: 3ex auto 0ex auto ;
    background: none ;
}

div.author table tbody tr td { padding: .25ex ; }

span.affiliation {font-size: .85em ; font-variant: small-caps; }

div.titledate {
    text-align: center ;
    font-size: .85em ;
    font-style: italic;
    margin: 6ex 0em 6ex 0em ;
}


nav.topnavigation{
    text-align: left ;
    padding: 0.5ex 1em 0.5ex 1em ;
/*     margin: 2ex 0em 3ex 0em ; */
    margin: 0 ;
    border-bottom: 1px solid silver ;
    border-top: 1px solid silver ;
    clear:right ;
}

nav.botnavigation{
    text-align: left ;
    padding: 0.5ex 1em 0.5ex 1em ;
/*     margin: 3ex 0em 2ex 0em ; */
    margin: 0 ;
    border-top: 1px solid silver ;
    border-bottom: 1px solid silver ;
    clear:right ;
}


header{
    line-height: 1.2 ;
```

```
    font-size: 1em ;
/*     border-bottom: 2px solid silver ; */
    margin: 0px ;
    padding: 0ex 1em 0ex 1em ;
    text-align:center ;
}

header p {margin:0ex;padding:4ex 0em 2ex 0em ;text-align:center;}


footer{
    font-size: .85em ;
    line-height: 1.2 ;
    margin-top: 1ex ;
    border-top: 2px solid silver ;
    padding: 2ex 1em 2ex 1em ;
    clear:right ;
    text-align:left ;
}


a.linkhome { font-weight:bold ; font-size: 1em ;}


div.lateximagesource { padding: 0px ; margin: 0px ; display: none; }

img.lateximage{
    padding: 0px 0px 0px 0px ;
    box-shadow: none ;
    border: none ;
    background: none ;
    margin: 0px 0px -.15ex 0px ;
        /* pdfcrop leaves a slight margin, adjust to baseline */
    max-width: 100% ;
    border-radius: 0ex ;
    border: none ;
}



nav.sidetoc {
    font-family: "DejaVu Serif", "Bitstream Vera Serif",
        "Lucida Bright", Georgia, serif;
    float:right ;
    width: 20%;
    border-left: 1px solid silver;
    border-top: 1px solid silver;
    border-bottom: 1px solid silver;
/*     border-top: 2px solid #808080 ; */
    background: #FAF7F4 ;
```

```
    padding: 2ex 0em 2ex 1em ;
    margin: 0ex 0em 2ex 1em ;
    font-size:.9em ;
    border-radius: 20px 0px 0px 20px ;
    }

div.sidetoccontents {
/*     border-top: 1px solid silver ; */
    overflow-y: auto ;
    width: 100% ;
    text-align: left ;
}

nav.sidetoc p {line-height:1.2 ; margin: 1ex .5em 1ex .5em ;
    text-indent: 0 ; }
nav.sidetoc p a {color:black ; font-size: .7em ;}
div.sidetoctitle {font-size: 1.2em; font-weight:bold; text-align:center;
    border-bottom: 1px solid silver ;    }
nav.sidetoc a:hover {text-decoration: underline ; }



section.textbody { margin: 0ex 1em 0ex 1em ;}


div.multicolsheading { -webkit-column-span: all;
    -moz-column-span: all; column-span: all; }
div.multicols { -webkit-columns: 3 380px ;
    -moz-columns: 3 380px ; columns: 3 380px ; }
div.multicols p {margin-top: 0ex}



/* Used to support algorithmicx: */
span.floatright { float: right ; }



/* Native LaTeX theorems: */

.theoremcontents { font-style: italic; margin-top: 3ex ; margin-bottom: 3ex ; }
.theoremlabel { font-style: normal; font-weight: bold ; margin-right: .5em ; }


/* theorem, amsthm, and ntheorem packages */

span.theoremheader,
span.theoremheaderplain,
span.theoremheaderdefinition,
```

```
span.theoremheaderbreak,
span.theoremheadermarginbreak,
span.theoremheaderchangebreak,
span.theoremheaderchange,
span.theoremheadermargin
{
font-style:normal ; font-weight: bold ; margin-right: 1em ;
}

span.amsthmnameplain,
span.amsthmnamedefinition,
span.amsthmnumberplain,
span.amsthmnumberdefinition
{
font-style:normal ; font-weight: bold ;
}


span.amsthmnameremark,
span.amsthmnumberremark
{font-style:italic ; font-weight: normal ; }


span.amsthmnoteplain,
span.amsthmnotedefinition
{font-style:normal ;}


span.theoremheaderremark,
span.theoremheaderproof,
span.amsthmproofname
{font-style:italic ; font-weight: normal ; margin-right: 1em ; }

span.theoremheadersc
{
font-style:normal ;
font-variant: small-caps ;
font-weight: normal ;
margin-right: 1em ;
}

.theoremendmark {float:right}

div.amsthmbodyplain, div.theorembodyplain, div.theorembodynonumberplain,
div.theorembodybreak, div.theorembodynonumberbreak,
div.theorembodymarginbreak,
div.theorembodychangebreak,
div.theorembodychange,
div.theorembodymargin
{
```

```
font-style:italic;
margin-top: 3ex ; margin-bottom: 3ex ;
}

div.theorembodydefinition, div.theorembodyremark, div.theorembodyproof,
div.theorembodyplainupright, nonumberplainuprightsc,
div.amsthmbodydefinition, div.amsthmbodyremark,
div.amsthmproof
{
font-style: normal ;
margin-top: 3ex ; margin-bottom: 3ex ;
}

span.amsthmnoteremark {}



/*
For CSS LaTeX and related logos:
Based on:
http://edward.oconnor.cx/2007/08/tex-poshlet
http://nitens.org/taraborelli/texlogo
*/

.latexlogofont {
    font-family: "Linux Libertine O", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-variant: normal ;
}

.latexlogo {
    font-family: "Linux Libertine O", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    letter-spacing: .03em ;
    font-size: 1.1em;
}

.latexlogo sup {
  text-transform: uppercase;
  letter-spacing: .03em ;
  font-size: 0.85em;
  vertical-align: 0.15em;
  margin-left: -0.36em;
  margin-right: -0.15em;
}

.latexlogo sub {
  text-transform: uppercase;
  vertical-align: -0.5ex;
  margin-left: -0.1667em;
```

```
  margin-right: -0.125em;
  font-size: 1em;
}

.xetexlogo {
    font-family: "Linux Libertine O", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    letter-spacing: .03em ;
    font-size: 1.1em;
}

/* A smaller gap between Xe and Tex v.s. LaTeX: */
.xetexlogo sub {
  text-transform: uppercase;
  vertical-align: -0.5ex;
  margin-left: -0.0667em;
  margin-right: -0.2em;
  font-size: 1em;
  letter-spacing: .03em ;
}

/* A large gap between Xe and LaTeX v.s. TeX: */
.xelatexlogo sub {
  text-transform: uppercase;
  vertical-align: -0.5ex;
  margin-left: -0.0667em;
  margin-right: -.05em;
  font-size: 1em;
  letter-spacing: .03em ;
}

.amslogo {
    font-family: "TeXGyreChorus","URW Chancery L",
        "Apple Chancery","ITC Zapf Chancery","Monotype Corsiva",
        "Linux Libertine O", "Nimbus Roman No 9 L",  "FreeSerif",
        "Hoefler Text", Times, "Times New Roman", serif;
  font-style: italic;
}

.lyxlogo {
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "DejaVu Sans", "Bitstream Vera Sans", Geneva,
Verdana, sans-serif ;
}



/* Only display top and bottom navigation if a small screen: */
/* Hide the sidetoc if a small screen: */
```

```
nav.topnavigation { display:none; }
nav.botnavigation { display:none; }

@media screen and (max-width: 45em) {
/*     nav.sidetoc {display:none;} */
    nav.sidetoc {
        float: none ;
        width: 100% ;
        margin: 5ex 0px 5ex 0px ;
        padding: 0 ;
        border-radius: 0 ;
        border-bottom: 1px solid black ;
        border-top: 1px solid black ;
        box-shadow: none ;
    }
/*     nav.topnavigation { display:block } */
    nav.botnavigation { display:block }
    .marginpar {
        max-width: 100%;
        float: none;
        display:block ;
        margin: 1ex 1em 1ex 1em ;
    }
}

@media print {
    body {
        font-family: "Linux Libertine O",
        "DejaVu Serif", "Bitstream Vera Serif",
        "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    }
    nav.sidetoc { display:none; }
    nav.topnavigation { display: none; }
    nav.botnavigation { display: none; }
}

@media handheld {
    nav.sidetoc { display:none; }
    nav.topnavigation { display:block }
    nav.botnavigation { display:block }
}

@media projection {
    nav.sidetoc { display:none; }
    nav.topnavigation { display:block }
    nav.botnavigation { display:block }
}
\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting
```

```
\end{warpprint}
```

26.5 lwarp_sagebrush.css

File `lwarp_sagebrush.css` An optional CSS which may be used for a semi-modern appearance.

If used, this must be present both when compiling the project and also when distributing the HTML files.

```
\begin{warpprint}
\begin{VerbatimOut}{lwarp_sagebrush.css}
@import url("lwarp.css") ;


A:link {color:#105030 ; text-decoration: none ; }
A:visited {color:#705030 ; text-shadow:1px 1px 2px #a0a0a0;}
A:hover {color:#006000 ; text-decoration: underline ; text-shadow:0px 0px 2px #a0a0a0;}
A:active {color:#00C000 ; text-shadow:1px 1px 2px #a0a0a0;}



h1, h2, h3, h4, h5, h6, span.paragraph, span.subparagraph
{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif",
        "Nimbus Roman No 9 L",  "FreeSerif",
        "Hoefler Text", Times, "Times New Roman", serif;
    font-variant: small-caps ;
font-weight: normal ;
    color: #304070 ;
    text-shadow: 2px 2px 3px #808080;
}

h1 {    /* title of the entire website, used on each page */
    font-variant: small-caps ;
    color: #304070 ;
    text-shadow: 2px 2px 3px #808080;
    background-color: #F7F7F0 ;
    background-image: linear-gradient(to bottom, #F7F7F0, #C0C0C4);
}

h1 {
  border-bottom: 1px solid #304070;
  border-top: 2px solid #304070;
}

h2 {
```

```
  border-bottom: 1px solid #304070;
  border-top: 2px solid #304070;
    background-color: #F7F7F0 ;
    background-image: linear-gradient(to bottom, #F7F7F0, #DAD0C0);
}



div.abstract {
    background: #f5f5eb ;
    background-image: linear-gradient(to bottom, #f5f5eb, #C8C8B8);

  border: 1px solid silver;
    border-radius: 1em ;
}

div.abstract dl {line-height:1.5;}
div.abstract dt {color:#304070;}

div.abstracttitle{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
    font-variant: small-caps ;
    font-size:1.5em;
    border-bottom: 1px solid silver ;
    color: #304070 ;
    text-align: center ;
    text-shadow: 1px 1px 2px #808080;
}

span.abstractrunintitle{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
}


div.epigraph {
    background: #f5f5eb ;
    background-image: linear-gradient(to bottom, #f5f5eb, #C8C8B8);

    border: 1px solid silver ;
    border-radius: 1ex ;
    box-shadow: 3px 3px 3px #808080 ;
}
```

```
.example {
    background-color: #f5f5eb ;
    background-image: linear-gradient(to bottom, #f5f5eb, #C8C8B8);

}

div.exampletitle{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
    font-variant: small-caps ;
    border-bottom: 1px solid silver ;
    color: #304070 ;
    text-align: center ;
    text-shadow: 1px 1px 2px #808080;
}


.sidebar {
    background-color: #f5f5eb ;
    background-image: linear-gradient(to bottom, #f5f5eb, #C8C8B8);

}

div.sidebartitle{
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
    font-variant: small-caps ;
    border-bottom: 1px solid silver ;
    color: #304070 ;
    text-align: center ;
    text-shadow: 1px 1px 2px #808080;
}


.fancyvrblabel {
    font-family: "URW Classico", Optima, "Linux Biolinum O",
        "Linux Libertine O", "Liberation Serif", "Nimbus Roman No 9 L",
        "FreeSerif", "Hoefler Text", Times, "Times New Roman", serif;
    font-weight:bold;
    font-variant: small-caps ;
font-size: 1.5em ;
    color: #304070 ;
    text-align: center ;
    text-shadow: 1px 1px 2px #808080;
}

```

```
.minipage {
    background-color: #eeeee7 ;
    border: 1px solid silver ;
border-radius: 1ex ;
}

.framed .minipage , .framedleftbar .minipage {
border: none ;
background: none ;
padding: 0ex ;
margin: 0ex ;
}

figure.figure .minipage, figcaption .minipage { border: none; }

div.marginblock div.minipage { border: none; }

figure , div.marginblock {
    background-color: #eeeee7 ;
    border: 1px solid silver ;
    border-radius: 1ex ;
    box-shadow: 3px 3px 3px #808080 ;
}

figure figure {
    border: 1px solid silver ;
    margin: 0em ;
box-shadow: none ;
}

/*
figcaption {
    border-top: 1px solid silver ;
    border-bottom: 1px solid silver ;
    background-color: #e8e8e8 ;
}
*/


div.table {
    box-shadow: 3px 3px 3px #808080 ;
}

/*
.tnotes {
    background: #e8e8e8;
    border: 1px solid silver;
}
```

```
*/


nav.topnavigation{
    background-color: #b0b8b0 ;
    background-image: linear-gradient(to bottom,#e0e0e0,#b0b8b0) ;
}

nav.botnavigation{
    background-color: #b0b8b0 ;
    background-image: linear-gradient(to top,#e0e0e0,#b0b8b0) ;
}



header{
    background-color: #F7F7F0 ;
    background-image: linear-gradient(to top, #F7F7F0, #b0b8b0);
}

footer{
    background-color: #F7F7F0 ;
    background-image: linear-gradient(to bottom, #F7F7F0, #b0b8b0);
}



nav.sidetoc {
    background-color: #F7F7F0 ;
    background-image: linear-gradient(to bottom, #F7F7F0, #C0C0C0);
    box-shadow: 3px 3px 3px #808080 ;
    border-radius: 0px 0px 0px 20px ;
    }

div.sidetoctitle {color: #304070 ; }

nav.sidetoc a:hover {
    color:#006000 ;
    text-decoration: none ;
    text-shadow:0px 0px 2px #a0a0a0;
}


@media screen and (max-width: 45em) {
    nav.sidetoc { border-radius: 0 ; }
}


\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting
```

```
\end{warpprint}
```

26.6 lwarp_formal.css

File `lwarp_formal.css` An optional CSS which may be used for a more formal appearance.

If used, this must be present both when compiling the project and also when distributing the HTML files.

```
\begin{warpprint}
\begin{VerbatimOut}{lwarp_formal.css}
@import url("lwarp.css") ;



A:link {color:#802020 ; text-decoration:none; }
A:visited {color:#802020 ; text-shadow:none ;}
A:hover {color:#400000 ; text-shadow:none ;}
A:active {color:#C00000 ; text-shadow:none ;}


body {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    background: #fffcf5;
}

span.textrm {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
}

span.textsf {
     font-family: "DejaVu Sans", "Bitstream Vera Sans",
        Geneva, Verdana, sans-serif ;
}



h1, h2, h3, h4, h5, h6, span.paragraph, span.subparagraph
{
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
```

```
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    color: #800000 ;
    text-shadow: none ;
}

h1, h2 {
    background-color: #fffcf5 ;
    background-image: none ;
    border-bottom: 1px solid #808080;
    border-top: 2px solid #808080;
}

div.abstracttitle {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    color: black ;
    text-shadow: none ;
}

span.abstractrunintitle {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    color: black ;
    text-shadow: none ;
}

div.abstract { font-size: 100% }

.sidebar {
    background: #fffcf5;
    background-image: none ;
  margin: 2em 5% 2em 5%;
  padding: 0.5em 1em;
  border: none ;
  border-top : 1px solid silver;
  border-bottom : 1px solid silver;
  font-size: 90% ;
}

div.sidebartitle{
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    color: #800000 ;
```

```
    text-shadow: none ;
    border: none ;
}

.example {
    background: #fffcf5;
    background-image: none ;
  margin: 2em 5% 2em 5%;
  padding: 0.5em 1em;
  border: none ;
  border-top : 1px solid silver;
  border-bottom : 1px solid silver;
}

div.exampletitle{
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    color: #800000 ;
    text-shadow: none ;
    border: none ;
}

div.fancyvrblabel{
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    color: #800000 ;
    text-shadow: none ;
    border: none ;
}



.verse {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
}


figure {
    margin: 3ex 5% 3ex 5% ;
    padding: 1ex 1em 1ex 1em ;
    background-color: #fffcf5 ;
    overflow-x: auto ;
    border: none ;
```

```
/*     border-top: 1px solid silver; */
/*     border-bottom: 1px solid silver; */
}


figcaption , .lstlisting {
    border: none ;
/*     border-top: 1px solid silver ; */
/*     border-bottom: 1px solid silver ; */
    background-color: #fffcf5 ;
}

.tnotes {
    background: #fffcf5 ;
}

.theorem {
        background: none ;
}

.minipage {
    background-color: #fffcf5 ;
    border: none ;
}

div.floatrow figure { border: none ; }

figure figure { border: none ; }


nav.toc, nav.lof, nav.lot, nav.lol {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
}

nav.sidetoc {
    font-family: "Linux Libertine O", "Hoefler Text", "Garamond",
        "Bembo", "Janson", "TeX Gyre Pagella", "Palatino",
        "Liberation Serif", "Nimbus Roman No 9 L",  "FreeSerif", Times,
        "Times New Roman", serif;
    background-image: linear-gradient(to bottom, #fffcf5, #C0C0C0);
    border-radius: 0px 0px 0px 20px ;
}

div.sidetoctitle{
    color: #800000 ;
}

```

```
header{
    background-color: #e0e0e0 ;
    background-image: linear-gradient(to top, #fffcf5, #b0b0b0);
    text-align:center ;
}

footer{
    background-color: #e0e0e0 ;
    background-image: linear-gradient(to bottom, #fffcf5, #b0b0b0);
    padding: 2ex 1em 2ex 1em ;
    clear:right ;
    text-align:left ;
}

nav.botnavigation {
    background: #dedcd5 ;
    border-top: 1px solid black ;
}
\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting
\end{warpprint}
```

26.7 sample_project.css

File `sample_project.css` The project-specific CSS file. Use with `\CSSFilename`.

If used, this must be present both when compiling the project and also when distributing the HTML files.

```
\begin{warpprint}
\begin{VerbatimOut}{sample_project.css}
/* ( --- Start of project.css --- ) */
/* A sample project-specific CSS file for lwarp --- ) */

/* Load default lwarp settings: */
@import url("lwarp.css") ;
/* or lwarp_formal.css, lwarp_sagebrush.css */

/* Project-specific CSS setting follow here. */
/* . . . */

/* ( --- End of project.css --- ) */
\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting
\end{warpprint}
```

26.8 lwarp.xdy

File `lwarp.xdy` Used to modify the index for lwarp.

This must be present when compiling the project, but does not need to be present when distributing the resulting HTML files.

```
\begin{warpprint}
\begin{VerbatimOut}{lwarp.xdy}
(require "tex/inputenc/latin.xdy")
(merge-rule "\\PS *" "Postscript")
(require "texindy.xdy")
(require "page-ranges.xdy")
(require "book-order.xdy")
(markup-locref :open "\hyperindexref{" :close "}")
\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting
\end{warpprint}
```

26.9 lwarp_mathjax.txt

File `lwarp_mathjax.txt` Used by lwarp when using MathJax.

This must be present when compiling the project, but does not need to be present when distributing the resulting HTML files.

```
\begin{warpprint}
\begin{VerbatimOut}{lwarp_mathjax.txt}
<!-- https://groups.google.com/forum/#!topic/
                              mathjax-users/jUtewUcE2bY -->
<script type="text/x-mathjax-config">
MathJax.Hub.Register.StartupHook("TeX AMSmath Ready",function () {
    var seteqsectionDefault = {name: "", num: 0};
    var seteqsections = {}, seteqsection = seteqsectionDefault;
    var TEX = MathJax.InputJax.TeX, PARSE = TEX.Parse;
    var AMS = MathJax.Extension["TeX/AMSmath"];
    TEX.Definitions.Add({
    macros: {
        seteqsection: "mySection",
        seteqnumber: "mySetEqNumber"
    }
    });

    PARSE.Augment({
    mySection: function (name) {
        seteqsection.num = AMS.number;
        var n = this.GetArgument(name);
```

```
        if (n === "") {
        seteqsection = seteqsectionDefault;
        } else {
        if (!seteqsections["_"+n])
            seteqsections["_"+n] = {name:n, num:0};
        seteqsection = seteqsections["_"+n];
        }
        AMS.number = seteqsection.num;
    },
    mySetEqNumber: function (name) {
        var n = this.GetArgument(name);
        if (!n || !n.match(/^ *[0-9]+ *$/))
            n = ""; else n = parseInt(n)-1;
        <!-- $ syntax highlighting -->
        if (n === "" || n < 1)
            TEX.Error
            ("Argument to "+name+" should be a positive integer");
        AMS.number = n;
    }
    });
    MathJax.Hub.Config({
    TeX: {
        equationNumbers: {
        formatTag: function (n)
            {return "("+(seteqsection.name+"."+n).replace(/^\./,"")+")"},
        formatID: function (n) {
            n = (seteqsection.name+'.'+n).replace
                (/[:"'<>&]/g,"").replace(/^\./,"");
            return 'mjx-eqn-' + n;
        }
        }
    }
    });
});
</script>

<!-- http://docs.mathjax.org/en/latest/options/ThirdParty.html -->
<script type="text/x-mathjax-config">
  MathJax.Ajax.config.path["Contrib"] =
    "https://cdn.mathjax.org/mathjax/contrib";
</script>

<!-- https://github.com/mathjax/MathJax-third-party-extensions/
                                          tree/master/siunitx -->
<script type="text/x-mathjax-config">
 MathJax.Hub.Config({
   extensions: ["tex2jax.js","[Contrib]/siunitx/siunitx.js"],
   jax: ["input/TeX","output/HTML-CSS"],
   tex2jax: {inlineMath: [["$","$"],["\\(","\\)"]]},
   TeX: {extensions: ["AMSmath.js","AMSsymbols.js", "sinuitx.js"]}
```

```
 });
 </script>

<script type="text/x-mathjax-config">
MathJax.Hub.Config({
    TeX: {
    equationNumbers: {
        autoNumber: "AMS"
    }
    }
});
</script>

<!-- Alternative CDN provider: -->
<script type="text/javascript" async
src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.0/MathJax.js?config=TeX-AMS_HTML-full">
</script>

<!-- No longer supported after April 30, 2017: -->
<!--
<script
  src="https://cdn.mathjax.org/mathjax/latest/MathJax.js?config=TeX-AMS_HTML-full">
</script>
-->

\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting
\end{warpprint}
```

26.10 lwarpmk option

The following is only generated if the `lwarpmk` option was given to lwarp.

```
\begin{LWR@createlwarpmk}
```

Prog `lwarpmk` Creates a local copy of `lwarpmk`:

```
\begin{VerbatimOut}{lwarpmk.lua}
#!/usr/bin/env texlua

-- Copyright 2016-2017 Brian Dunn

-- Print the usage of the lwarpmk command:

printversion = "v0.31"

function printhelp ()
```

```
print ("lwarpmk: Use lwarpmk -h or lwarpmk --help for help.") ;
end

function printusage ()
print ( [[

lwarpmk print [project]: Compile a print version.
lwarpmk printindex [project]: Process the index for the print version.
lwarpmk printglossary [project]: Process the glossary for the print version.
lwarpmk html [project]: Compile an HTML version.
lwarpmk htmlindex [project]: Process the index for the html version.
lwarpmk htmlglossary [project]: Process the glossary for the html version.
lwarpmk again [project]: Touch the source code to trigger recompiles.
lwarpmk limages [project]: Process the "lateximages" created by lwarp.sty.
lwarpmk pdftohtml [project]:
    For use with latexmk or a Makefile:
    Convert project_html.pdf to project_html.html and
    individual HTML files.
lwarpmk clean [project]: Remove project.aux, .toc, .lof/t, .idx, .ind, .log, .gl*
lwarpmk cleanall [project]: Remove auxiliary files and also project.pdf, *.html
lwarpmk -h: Print this help message.
lwarpmk --help: Print this help message.

]] )
printconf ()
end

-- Print the format of the configuration file lwarpmk.conf:

function printconf ()
print ( [[
An example lwarpmk.conf or <project>.lwarpmkconf project file:
--
opsystem = "Unix"   (or "Windows")
latexname = "pdflatex"  (or "lualatex", or "xelatex")
sourcename = "projectname"  (the source-code filename w/o .tex)
homehtmlfilename = "index"  (or perhaps the project name)
htmlfilename = ""  (or "projectname" - filename prefix)
latexmk = "false"  (or "true" to use latexmk to build PDFs)
languge = "english"  (use a language supported by xindy)
xdyfile = "lwarp.xdy" (or a custom file based on lwarp.xdy)
--
Filenames must contain only letters, numbers, underscore, or dash.
Values must be in "quotes".

]] ) ;
end


-- Split one large sourcefile into a number of files,
```

```
-- starting with destfile.
-- The file is split at each occurance of <!--|Start file|newfilename|*

function splitfile (destfile,sourcefile)
print ("lwarpmk: Splitting " .. sourcefile .. " into " .. destfile) ;
io.input(sourcefile)
io.output(destfile)
for line in io.lines() do
i,j,copen,cstart,newfilename = string.find (line,"(.*)|(.*)|(.*)|") ;
if ( (i~= nil) and (copen == "<!--") and (cstart == "Start file")) then -- split the file
io.output(newfilename) ;
else -- not a splitpoint
io.write (line .. "\n") ;
end
end -- do
end -- function

-- Incorrect value, so print an error and exit.

function cvalueerror ( line, linenum , cvalue )
    print ( linenum .. " : " .. line ) ;
    print ("lwarpmk: incorrect variable value \"" .. cvalue .. "\" in lwarpmk.conf.\n" ) ;
    printconf () ;
    os.exit(1) ;
end

-- Load settings from the project's "lwarpmk.conf" file:

function loadconf ()
-- Default configuration filename:
local conffile = "lwarpmk.conf"
-- Optional configuration filename:
if arg[2] ~= nil then conffile = arg[2]..".lwarpmkconf" end
-- Default language:
language = "english"
-- Default xdyfile:
xdyfile = "lwarp.xdy"
-- Verify the file exists:
if (lfs.attributes(conffile,"mode")==nil) then -- file not exists
print("lwarpmk: " .. conffile .." does not exist.")
print("lwarpmk: " .. arg[2] .. " does not appear to be a project name.\n")
printhelp () ;
os.exit(1) -- exit the entire lwarpmk script
else -- file exists
-- Read the file:
print ("lwarpmk: Reading " .. conffile ..".")
io.input(conffile) ;
-- Scan each line:
local linenum = 0
for line in io.lines() do -- scan lines
```

```
linenum = linenum + 1
i,j,cvarname,cvalue = string.find (line,"([%w-_]*)%s*=%s*\"([%w%-_%.]*)\"") ;
-- Error if incorrect enclosing characters:
if ( i == nil ) then
print ( linenum .. " : " .. line ) ;
print ( "lwarpmk: Incorrect entry in " .. conffile ..".\n" ) ;
printconf () ;
os.exit(1) ;
end
if ( cvarname == "opsystem" ) then
    -- Verify choice of opsystem:
    if ( (cvalue == "Unix") or (cvalue == "Windows") ) then
        opsystem = cvalue
    else
        cvalueerror ( line, linenum , cvalue )
    end
elseif ( cvarname == "latexname" ) then
    -- Verify choice of LaTeX compiler:
    if (
        (cvalue == "pdflatex") or
        (cvalue == "xelatex") or
        (cvalue == "lualatex")
    ) then
        latexname = cvalue
    else
        cvalueerror ( line, linenum , cvalue )
    end
elseif ( cvarname == "sourcename" ) then sourcename = cvalue
elseif ( cvarname == "homehtmlfilename" ) then homehtmlfilename = cvalue
elseif ( cvarname == "htmlfilename" ) then htmlfilename = cvalue
elseif ( cvarname == "latexmk" ) then latexmk = cvalue
elseif ( cvarname == "language" ) then language = cvalue
elseif ( cvarname == "xdyfile" ) then xdyfile = cvalue
else
print ( linenum .. " : " .. line ) ;
print ("lwarpmk: Incorrect variable name \"" .. cvarname .. "\" in " .. conffile ..".\n" ) ;
printconf () ;
os.exit(1) ;
end
end -- do scan lines
end -- file exists
-- Select some operating-system commands:
if opsystem=="Unix" then  -- For Unix / Linux / Mac OS:
rmname = "rm"
mvname = "mv"
touchnamepre = "touch"
touchnamepost = ""
dirslash = "/"
opquote= "\'"
elseif opsystem=="Windows" then -- For Windows
```

```
rmname = "DEL"
mvname = "MOVE"
touchnamepre = "COPY /b"
touchnamepost = "+,,"
dirslash = "\\"
opquote= "\""
else print ( "lwarpmk: Select Unix or Windows for opsystem" )
end --- for Windows

-- set xindycmd according to pdflatex vs xelatex/lualatex:
if ( latexname == "pdflatex" ) then
xindycmd = "texindy  -C utf8"
glossarycmd = "xindy -C utf8"
else
xindycmd = "xindy  -M texindy  -C utf8"
glossarycmd = "xindy -C utf8"
end

end -- loadconf


function refreshdate ()
os.execute(touchnamepre .. " " .. sourcename .. ".tex " .. touchnamepost)
end


-- Scan the LaTeX log file for the phrase "Rerun to get",
-- indicating that the file should be compiled again.
-- Return true if found.

function reruntoget (filesource)
io.input(filesource)
for line in io.lines() do
if ( string.find(line,"Rerun to get") ~= nil ) then return true end
end
return false
end


-- Compile one time, return true if should compile again.
-- fsuffix is "" for print, "_html" for HTML output.

function onetime (fsuffix)
print("lwarpmk: Compiling with " .. latexname .. " " .. sourcename..fsuffix)
err = os.execute(
--    "echo " ..
    latexname .. " " .. sourcename..fsuffix )
if ( err ~= 0 ) then print ( "lwarpmk: Compile error.") ; os.exit(1) ; end
return (reruntoget(sourcename .. fsuffix .. ".log") ) ;
end
```

```


-- Compile up to five times.
-- fsuffix is "" for print, "_html" for HTML output

function manytimes (fsuffix)
if onetime(fsuffix) == true then
if onetime(fsuffix) == true then
if onetime(fsuffix) == true then
if onetime(fsuffix) == true then
if onetime(fsuffix) == true then
end end end end end
end

-- Exit if the given file does not exist.

function verifyfileexists (filename)
if (lfs.attributes ( filename , "modification" ) == nil ) then
print ( "lwarpmk: " .. filename .. " not found." ) ;
os.exit (1) ;
end
end


-- Convert <project>_html.pdf into HTML files:

function pdftohtml ()
    -- Convert to text:
    print ("lwarpmk: Converting " .. sourcename
        .."_html.pdf to " .. sourcename .. "_html.html")
    os.execute("pdftotext  -enc UTF-8  -nopgbrk  -layout "
        .. sourcename .. "_html.pdf " .. sourcename .. "_html.html")
    -- Split the result into individual HTML files:
    splitfile (homehtmlfilename .. ".html" , sourcename .. "_html.html")
end


-- Remove auxiliary files:

function removeaux ()
    os.execute ( rmname .. " " ..
        sourcename ..".aux " .. sourcename .. "_html.aux " ..
        sourcename ..".toc " .. sourcename .. "_html.toc " ..
        sourcename ..".lof " .. sourcename .. "_html.lof " ..
        sourcename ..".lot " .. sourcename .. "_html.lot " ..
        sourcename ..".idx " .. sourcename .. "_html.idx " ..
        sourcename ..".ind " .. sourcename .. "_html.ind " ..
        sourcename ..".log " .. sourcename .. "_html.log " ..
        sourcename ..".gl* " .. sourcename .. "_html.gl* "
        )
```

```
end



-- Create lateximages based on lateximages.txt:
function createlateximages ()
print ("lwarpmk: Creating lateximages.")
io.input("lateximages.txt")
-- Create the lateximages directory, ignore error if alreadt exists
err = os.execute("mkdir lateximages")
-- Scan lateximages.txt
for line in io.lines() do
-- lwimgpage is the page number in the PDF which has the image
-- lwimgnum is the sequential lateximage number to assign for the image
i,j,lwimgpage,lwimgnum = string.find (line,"|(.*)|(.*)|")
-- For each entry:
if ( (i~=nil) ) then
-- Separate out the image into its own single-page pdf:
err = os.execute(
"pdfseparate -f " .. lwimgpage .. " -l " ..
 lwimgpage .. " " .. sourcename .."_html.pdf lateximagetemp-%d.pdf")
-- Crop the image:
err = os.execute(
"pdfcrop --hires lateximagetemp-" .. lwimgpage ..".pdf lateximage-" .. lwimgnum ..".pdf")
if ( err ~= 0 ) then print ( "lwarpmk: File error.") ; os.exit(1) ; end
-- Convert the image to svg:
err = os.execute(
"pdftocairo -svg lateximage-" .. lwimgnum ..".pdf lateximage-" .. lwimgnum ..".svg")
if ( err ~= 0 ) then print ( "lwarpmk: File error.") ; os.exit(1) ; end
-- Move the result into lateximages/:
err = os.execute(
mvname .. " lateximage-" .. lwimgnum ..".svg lateximages" .. dirslash )
if ( err ~= 0 ) then print ( "lwarpmk: File error.") ; os.exit(1) ; end
-- Remove the temporary files:
err = os.execute(
rmname .. " lateximage-" .. lwimgnum ..".pdf lateximagetemp-" .. lwimgpage ..".pdf")
if ( err ~= 0 ) then print ( "lwarpmk: File error.") ; os.exit(1) ; end
end
end -- do
end -- function


-- Use latexmk to compile source and index:
-- fsuffix is "" for print, or "_html" for HTML
function compilelatexmk ( fsuffix )
    -- The recorder option is required to detect changes in <project>.tex
    -- while we are loading <project>_html.tex.
    err=os.execute ( "latexmk -pdf -dvi- -ps- -recorder "
        .. "-e "
        .. opquote
```

```
        .. "$makeindex = q/"
        .. xindycmd
        .. "  -M " .. xdyfile
        .. "  -L " .. language .. " /"
        .. opquote
        .. " -pdflatex=\"" .. latexname .." %O %S\" "
        .. sourcename..fsuffix ..".tex" ) ;
    if ( err ~= 0 ) then print ( "lwarpmk: Compile error.") ; os.exit(1) ; end
end



-- lwarpmk --version :

if (arg[1] == "--version") then
print ( "lwarpmk: " .. printversion )

else -- not -- version

-- print intro:

print ("lwarpmk: " .. printversion .. "  Automated make for the LaTeX lwarp package.")

-- lwarpmk print:

if arg[1] == "print" then
loadconf ()
if ( latexmk == "true" ) then
    compilelatexmk ("")
    print ("lwarpmk: Done.")
else -- not latexmk
    verifyfileexists (sourcename .. ".tex") ;
    -- See if up to date:
    if (
        ( lfs.attributes ( sourcename .. ".pdf" , "modification" ) == nil ) or
        (
            lfs.attributes ( sourcename .. ".tex" , "modification" ) >
            lfs.attributes ( sourcename .. ".pdf" , "modification" )
        )
    ) then
        -- Recompile if not yet up to date:
        manytimes("")
        print ("lwarpmk: Done.") ;
    else
        print ("lwarpmk: " .. sourcename .. ".pdf is up to date.") ;
    end
end -- not latexmk

-- lwarp printindex:
-- Compile the index then touch the source
```

```
-- to trigger a recompile of the document:

elseif arg[1] == "printindex" then
loadconf ()
print ("lwarpmk: Processing the index.")
os.execute(
    xindycmd
    .. "  -M " .. xdyfile
    .. "  -L " .. language
    .. " " .. sourcename .. ".idx")
print ("lwarpmk: Forcing an update of " .. sourcename ..".tex.")
refreshdate ()
print ("lwarpmk: " .. sourcename ..".tex is ready to be recompiled.")
print ("lwarpmk: Done.")

-- lwarp printglossary:
-- Compile the glossary then touch the source
-- to trigger a recompile of the document:

elseif arg[1] == "printglossary" then
loadconf ()
print ("lwarpmk: Processing the glossary.")

os.execute(glossarycmd .. "  -L " .. language .. "  -I xindy -M " .. sourcename ..
    " -t " .. sourcename .. ".glg -o " .. sourcename .. ".gls "
    .. sourcename .. ".glo")
print ("lwarpmk: Forcing an update of " .. sourcename ..".tex.")
refreshdate ()
print ("lwarpmk: " .. sourcename ..".tex is ready to be recompiled.")
print ("lwarpmk: Done.")

-- lwarpmk html:

elseif arg[1] == "html" then
loadconf ()
if ( latexmk == "true" ) then
    compilelatexmk ("_html")
    pdftohtml ()
    print ("lwarpmk: Done.")
else -- not latexmk
    verifyfileexists ( sourcename .. ".tex" ) ;
    -- See if exists and is up to date:
    if (
        ( lfs.attributes ( homehtmlfilename .. ".html" , "modification" ) == nil ) or
        (
            lfs.attributes ( sourcename .. ".tex" , "modification" ) >
            lfs.attributes ( homehtmlfilename .. ".html" , "modification" )
        )
    ) then
        -- Recompile if not yet up to date:
```

```
        manytimes("_html")
        pdftohtml ()
        print ("lwarpmk: Done.")
    else
        print ("lwarpmk: " .. homehtmlfilename .. ".html is up to date.")
    end
end -- not latexmk

elseif arg[1] == "pdftohtml" then
    loadconf ()
    pdftohtml ()

-- lwarpmk htmlindex:
-- Compile the index then touch the source
-- to trigger a recompile of the document:

elseif arg[1] == "htmlindex" then
loadconf ()
print ("lwarpmk: Processing the index.")
os.execute(
    xindycmd
    .. "  -M " .. xdyfile
    .. "  -L " .. language
    .. " " .. sourcename .. "_html.idx"
)
print ("lwarpmk: Forcing an update of " .. sourcename ..".tex.")
refreshdate ()
print ("lwarpmk: " .. sourcename ..".tex is ready to be recompiled.")
print ("lwarpmk: Done.")

-- lwarpmk htmlglossary:
-- Compile the glossary then touch the source
-- to trigger a recompile of the document:

elseif arg[1] == "htmlglossary" then
loadconf ()
print ("lwarpmk: Processing the glossary.")

os.execute(glossarycmd .. "  -L " .. language .. "  -I xindy -M " ..sourcename ..
    "_html -t " .. sourcename .. "_html.glg -o " ..sourcename ..
    "_html.gls " ..sourcename .. "_html.glo")

print ("lwarpmk: Forcing an update of " .. sourcename ..".tex.")
refreshdate ()
print ("lwarpmk: " .. sourcename ..".tex is ready to be recompiled.")
print ("lwarpmk: Done.")

-- lwarpmk limages:
-- Scan the lateximages.txt file to create lateximages,
-- then touch the source to trigger a recompile.
```

```

elseif arg[1] == "limages" then
loadconf ()
print ("lwarpmk: Processing images.")
createlateximages ()
print ("lwarpmk: Forcing an update of " .. sourcename ..".tex.")
refreshdate ()
print ("lwarpmk: " .. sourcename ..".tex is ready to be recompiled.")
print ("lwarpmk: Done.")

-- lwarpmk again:
-- Touch the source to trigger a recompile.

elseif arg[1] == "again" then
loadconf ()
print ("lwarpmk: Forcing an update of " .. sourcename ..".tex.")
refreshdate ()
print ("lwarpmk: " .. sourcename ..".tex is ready to be recompiled.")
print ("lwarpmk: Done.")

-- lwarpmk clean:
-- Remove project.aux, .toc, .lof, .lot, .idx, .ind, .log, .gl*

elseif arg[1] == "clean" then
loadconf ()
removeaux ()
print ("lwarpmk: Done.")

-- lwarpmk cleanall
-- Remove project.aux, .toc, .lof, .lot, .idx, .ind, .log, .gl*
--    and also project.pdf, *.html

elseif arg[1] == "cleanall" then
loadconf ()
removeaux ()
os.execute ( rmname .. " " ..
    sourcename .. ".pdf " .. sourcename .. "_html.pdf " ..
    "*.html"
    )
print ("lwarpmk: Done.")

-- lwarpmk with no argument :

elseif (arg[1] == nil) then
printhelp ()

-- lwarpmk -h or lwarpmk --help :

elseif (arg[1] == "-h" ) or (arg[1] == "--help") then
printusage ()
```

```

else
print ("lwarpmk: Unknown command \""..arg[1].."\".\n")
printhelp ()
end

end -- not --version
\end{VerbatimOut}
% \end{Verbatim}% for syntax highlighting

\end{LWR@createlwarpmk}
```

27 Stacks

for HTML output:

```
\begin{warpHTML}
```

Stacks are used to remember how to close sections and list items. Before a new section is started, previously nested sections and items must be closed out (unnested) in proper order. Note that starting a new section may close several levels of previously nested items at the same time. For example, starting a new `\section` would close any currently open subsection, subsubsection, and paragraph. General environments are not nested on the stack since they have their own close mechanism. List environments are nested, and items inside those environments are nested one level deeper still. List environments may be nested inside other list environments, and list items are nested inside list environments as well. Thus, the stack may have items which are not necessarily in order, since a description may contain an enumerate, for example. Depths to be recorded in `\LWR@closedepthone`, etc.

27.1 Assigning depths

initial depths for empty stack entries:

```
\newcommand*{\LWR@depthnone}{-5}
```

all sectioning depths are deeper than `LWR@depthfinished`:

```
\newcommand*{\LWR@depthfinished}{-4}
\newcommand*{\LWR@depthpart}{-1}
\newcommand*{\LWR@depthchapter}{0}
\newcommand*{\LWR@depthsection}{1}
\newcommand*{\LWR@depthsubsection}{2}
\newcommand*{\LWR@depthsubsubsection}{3}
\newcommand*{\LWR@depthparagraph}{4}
\newcommand*{\LWR@depthsubparagraph}{5}
```

used by \itemize, \enumerate, \description:

```
\newcommand*{\LWR@depthlist}{6}
```

used by \item:

```
\newcommand*{\LWR@depthlistitem}{7}
```

27.2 Closing actions

A stack to record the action to take to close each nesting level: Add more levels of stack if necessary for a very deeply nested document, adding to \pushclose and \popclose as well.

```
\newcommand*{\LWR@closeone}{}% top of the stack
\newcommand*{\LWR@closetwo}{}
\newcommand*{\LWR@closethree}{}
\newcommand*{\LWR@closefour}{}
\newcommand*{\LWR@closefive}{}
\newcommand*{\LWR@closesix}{}
\newcommand*{\LWR@closeseven}{}
\newcommand*{\LWR@closeeight}{}
\newcommand*{\LWR@closenine}{}
\newcommand*{\LWR@closeten}{}
\newcommand*{\LWR@closeeleven}{}
\newcommand*{\LWR@closetwelve}{}
```

27.3 Closing depths

A stack to record the depth of each level:

⚠ Note that nested LaTeX structures may push depths which are non-sequential.

Ex:

```
\begin{itemize}
   \item{A}
   \begin{description}
      \item{B}
   \end{description}
\end{itemize}
```

```
\newcommand*{\LWR@closedepthone}{\LWR@depthnone}% top of the stack
\newcommand*{\LWR@closedepthtwo}{\LWR@depthnone}
\newcommand*{\LWR@closedepththree}{\LWR@depthnone}
\newcommand*{\LWR@closedepthfour}{\LWR@depthnone}
\newcommand*{\LWR@closedepthfive}{\LWR@depthnone}
\newcommand*{\LWR@closedepthsix}{\LWR@depthnone}
\newcommand*{\LWR@closedepthseven}{\LWR@depthnone}
\newcommand*{\LWR@closedeptheight}{\LWR@depthnone}
\newcommand*{\LWR@closedepthnine}{\LWR@depthnone}
\newcommand*{\LWR@closedepthten}{\LWR@depthnone}
\newcommand*{\LWR@closedeptheleven}{\LWR@depthnone}
\newcommand*{\LWR@closedepthtwelve}{\LWR@depthnone}
```

27.4 Pushing and popping the stack

`\pushclose` {⟨*action*⟩} {⟨*depth*⟩}

Pushes one return action and its LaTeX depth onto the stacks.

```
\NewDocumentCommand{\pushclose}{m m}
{
\let\LWR@closetwelve\LWR@closeeleven
\let\LWR@closeeleven\LWR@closeten
\let\LWR@closeten\LWR@closenine
\let\LWR@closenine\LWR@closeeight
\let\LWR@closeeight\LWR@closeseven
\let\LWR@closeseven\LWR@closesix
\let\LWR@closesix\LWR@closefive
\let\LWR@closefive\LWR@closefour
\let\LWR@closefour\LWR@closethree
\let\LWR@closethree\LWR@closetwo
\let\LWR@closetwo\LWR@closeone
\let\LWR@closeone#1
\let\LWR@closedepthtwelve\LWR@closedeptheleven
\let\LWR@closedepthelevel\LWR@closedepthten
\let\LWR@closedepthten\LWR@closedepthnine
\let\LWR@closedepthnine\LWR@closedeptheight
\let\LWR@closedeptheight\LWR@closedepthseven
\let\LWR@closedepthseven\LWR@closedepthsix
\let\LWR@closedepthsix\LWR@closedepthfive
\let\LWR@closedepthfive\LWR@closedepthfour
\let\LWR@closedepthfour\LWR@closedepththree
\let\LWR@closedepththree\LWR@closedepthtwo
\let\LWR@closedepthtwo\LWR@closedepthone
\let\LWR@closedepthone#2
}
```

`\popclose` Pops one action and its depth off the stacks.

```
\newcommand*{\popclose}
{
\let\LWR@closeone\LWR@closetwo
\let\LWR@closetwo\LWR@closethree
\let\LWR@closethree\LWR@closefour
\let\LWR@closefour\LWR@closefive
\let\LWR@closefive\LWR@closesix
\let\LWR@closesix\LWR@closeseven
\let\LWR@closeseven\LWR@closeeight
\let\LWR@closeeight\LWR@closenine
\let\LWR@closenine\LWR@closeten
\let\LWR@closeten\LWR@closeeleven
\let\LWR@closeeleven\LWR@closetwelve
\let\LWR@closedepthone\LWR@closedepthtwo
\let\LWR@closedepthtwo\LWR@closedepththree
\let\LWR@closedepththree\LWR@closedepthfour
\let\LWR@closedepthfour\LWR@closedepthfive
\let\LWR@closedepthfive\LWR@closedepthsix
\let\LWR@closedepthsix\LWR@closedepthseven
\let\LWR@closedepthseven\LWR@closedeptheight
\let\LWR@closedeptheight\LWR@closedepthnine
\let\LWR@closedepthnine\LWR@closedepthten
\let\LWR@closedepthten\LWR@closedeptheleven
\let\LWR@closedeptheleven\LWR@closedepthtwelve
}
```

```
\end{warpHTML}
```

28 Data arrays

These macros are similar to the arrayjobx package, except that `\LWR@setexparray`'s argument is expanded only once when assigned.

`name` has no backslash, `index` can be a number or a text name, and an empty `value` must be `\relax` instead of empty.

To assign an empty value:

```
\LWR@setexparray{name}{index}{\relax}
```

for HTML output:

```
\begin{warpHTML}
```

`\LWR@setexparray` `{`⟨*name*⟩`}` `{`⟨*index*⟩`}` `{`⟨*contents*⟩`}`

```
\NewDocumentCommand{\LWR@setexparray}{m m m}{%
\expandafter\edef\csname #1#2\endcsname{\expandonce#3}%
}
```

\LWR@getexparray {⟨*name*⟩} {⟨*index*⟩}

```
\newcommand*{\LWR@getexparray}[2]{\csuse{#1#2}}
```

```
\end{warpHTML}
```

29 HTML entities

for HTML output:

```
\begin{warpHTML}
```

HTML entites and HTML Unicode entities:

```
\let\LWR@origampersand\&
```

\HTMLentity {⟨*entitytag*⟩}

```
\newcommand*{\HTMLentity}[1]{\LWR@origampersand#1;}
```

\HTMLunicode {⟨*hex_unicode*⟩}

```
\newcommand*{\HTMLunicode}[1]{\HTMLentity{\#x#1}}
```

\&

```
\renewcommand*{\&}{\HTMLentity{amp}}
```

\textless
\textgreater

```
\let\LWR@origtextless\textless
\renewcommand*{\textless}{\HTMLentity{lt}}

\let\LWR@origtextgreater\textgreater
\renewcommand*{\textgreater}{\HTMLentity{gt}}
```

```
\end{warpHTML}
```

30 HTML filename generation

The filename of the homepage is set to `\HomeHTMLFilename.html`. The filenames of additional sections start with `\HTMLFilename`, to which is appended a section number or a simplified section name, depending on `FileSectionNames`.

for HTML & PRINT:
```
\begin{warpall}
```

`\BaseJobname` The `\jobname` of the printed version, even if currently compiling the HTML version. I.e. this is the `\jobname` without `_html` appended. This is used to set `\HomeHTMLFilename` if the user did not provide one.

```
\providecommand*{\BaseJobname}{\jobname}
```

`\HTMLFilename` The prefix for all generated HTML files other than the home page, defaulting to empty. See section 5.7.

```
\providecommand*{\HTMLFilename}{}
```

`\HomeHTMLFilename` The filename of the home page, defaulting to the `\BaseJobname`. See section 5.7.

```
\providecommand*{\HomeHTMLFilename}{\BaseJobname}
```

`\SetHTMLFileNumber` `{`⟨*number*⟩`}`

Sets the file number for the next file to be generated. 0 is the home page. Use just before the next sectioning command, and set it to one less than the desired number of the next section. May be used to generate numbered groups of nodes such as 100+ for one chapter, 200+ for another chapter, etc.

```
\newcommand*{\SetHTMLFileNumber}[1]{%
\setcounter{LWR@htmlfilenumber}{#1}%
}
```

Bool `FileSectionNames` Selects how to create HTML file names.

Defaults to use section names in the filenames.

```
\newbool{FileSectionNames}
\booltrue{FileSectionNames}
```

```
\end{warpall}
```

for HTML output:
```
\begin{warpHTML}
```

Ctr `LWR@htmlfilenumber` Records the number of each HTML file as it is being created. Number 0 is the home page.

```
\newcounter{LWR@htmlfilenumber}
\setcounter{LWR@htmlfilenumber}{0}
```

`\LWR@htmlsectionfilename` {⟨*htmlfilenumber or name*⟩}

Prints the filename for a given section: `\HTMLFilename{}filenumber/name.html`

```
\newcommand*{\LWR@htmlsectionfilename}[1]{%
\LWR@traceinfo{LWR@htmlsectionfilename A}%
```

Section 0 or empty is given the home filename. The filename must be detokenized for underscores.

```
\LWR@traceinfo{about to assign temp}%
\edef\LWR@tempone{#1}%
\LWR@traceinfo{about to compare with ??}%
\ifthenelse{\equal{\LWR@tempone}{??}}%
{%
\LWR@traceinfo{found ??}%
}{%
\LWR@traceinfo{not found ??}%
}%
\LWR@traceinfo{about to compare with zero or empty}%
\ifthenelse{%
\equal{\LWR@tempone}{0}%
\OR \equal{\LWR@tempone}{}%
\OR \equal{\LWR@tempone}{??}%
}%
{%
\LWR@traceinfo{LWR@htmlsectionfilename B \HomeHTMLFilename.html}%
\HomeHTMLFilename.html%
}%
```

For a LaTeX section named “Index” or “index” without a prefix, create a filename with a leading underscore to avoid colliding with the HTML filename `index.html`:

```
{%
\LWR@traceinfo{LWR@htmlsectionfilename C \LWR@tempone}%
\ifthenelse{%
\equal{\HTMLFilename}{} \AND \equal{\LWR@tempone}{Index} \OR \equal{\LWR@tempone}{index}%
}%
{%
\LWR@traceinfo{prefixing the index name with an underscore.}%
\_#1.html}%
```

Otherwise, create a filename with the chosen prefix:

```
{\HTMLFilename#1.html}%
}%
\LWR@traceinfo{LWR@htmlsectionfilename Z}%
}
```

\LWR@htmlrefsectionfilename {⟨*label*⟩}

Prints the filename for the given label

```
\newcommand*{\LWR@htmlrefsectionfilename}[1]{%
\LWR@traceinfo{LWR@htmlrefsectionfilename A: !#1!}%
\LWR@htmlsectionfilename{\LWR@htmlfileref{#1}}%
\LWR@traceinfo{LWR@htmlrefsectionfilename B}%
}
```

```
\end{warpHTML}
```

31 Homepage link

for HTML output:

```
\begin{warpHTML}
```

\LinkHome \LinkHome may be used wherever you wish to place a link back to the homepage. The filename must be detokenized for underscores.

```
\newcommand*{\LinkHome}{%
\LWR@subhyperrefclass{%
\HomeHTMLFilename.html}%
{Home}{linkhome}%
}
```

\LWR@topnavigation Creates a link to the homepage at the top of the page for use when the window is too narrow for the sideTOC.

```
\newcommand*{\LWR@topnavigation}{
\LWR@htmlelementclassline{nav}{topnavigation}{\LinkHome}
}
```

\LWR@botnavigation Creates a link to the homepage at the bottom of the page for use when the window is too narrow for the sideTOC.

```
\newcommand*{\LWR@botnavigation}{
\LWR@htmlelementclassline{nav}{botnavigation}{\LinkHome}
}
```

```
\end{warpHTML}
```

32 \PrintStack diagnostic tool

⚠ Diagnostics tool: Prints the LaTeX nesting depth values for the stack levels. Must have `\LWR@startpars` active while printing the stack, so `\PrintStack` may be called from anywhere in the normal text flow.

for HTML output:

```
\begin{warpHTML}
```

`\PrintStack` Prints the closedepth stack.

```
\newcommand*{\PrintStack}{
\LWR@startpars
\LWR@closedepthone{} \LWR@closedepththwo{} \LWR@closedepththree{}
\LWR@closedepthfour{} \LWR@closedepthfive{} \LWR@closedepthsix{}
\LWR@closedepthseven{} \LWR@closedeptheight{} \LWR@closedepthnine{}
\LWR@closedepthten{} \LWR@closedeptheleven{} \LWR@closedepthtwelve{}
}
```

```
\end{warpHTML}
```

33 Closing stack levels

for HTML output:

```
\begin{warpHTML}
```

Close one nested level:

```
\newcommand*{\LWR@closeoneprevious}{%

\LWR@closeone{}

\popclose{}
}
```

`\LWR@closeprevious` `{`⟨*depth*⟩`}` Close everything up to the given depth:

```
\newcommand*{\LWR@closeprevious}[1]{
```

Close any pending paragraph:

```
\LWR@stoppars
```

Close anything nested deeper than the desired depth:

```
\whiledo{\not\(\LWR@closedepthone<#1\)}{\LWR@closeoneprevious}
}
```

```
\end{warpHTML}
```

34 Forcing a new PDF page

for HTML output:

```
\begin{warpHTML}
```

`\LWR@forcenewpage` New PDF page a before major environment.

This is used just before major environments, such as `verse`. Reduces the chance of an environment overflowing the HTML PDF output page.

```
\newcommand{\LWR@forcenewpage}{%
\LWR@stoppars\LWR@orignewpage\LWR@startpars%
}
```

```
\end{warpHTML}
```

35 HTML tags, spans, divs, elements

for HTML output:

```
\begin{warpHTML}
```

35.1 Mapping LaTeX Sections to HTML Sections

```
\newcommand*{\LWR@tagpart}{h2}
\newcommand*{\LWR@tagpartend}{/h2}
\newcommand*{\LWR@tagchapter}{h3}
\newcommand*{\LWR@tagchapterend}{/h3}
\newcommand*{\LWR@tagsection}{h4}
\newcommand*{\LWR@tagsectionend}{/h4}
\newcommand*{\LWR@tagsubsection}{h5}
\newcommand*{\LWR@tagsubsectionend}{/h5}
\newcommand*{\LWR@tagsubsubsection}{h6}
\newcommand*{\LWR@tagsubsubsectionend}{/h6}
\newcommand*{\LWR@tagparagraph}{span class="paragraph"{}}
\newcommand*{\LWR@tagparagraphend}{/span}
\newcommand*{\LWR@tagsubparagraph}{span class="subparagraph"{}}
```

```
\newcommand*{\LWR@tagsubparagraphend}{/span}

\newcommand*{\LWR@tagregularparagraph}{p}
```

35.2 HTML tags

`\LWR@htmltagc` {⟨*tag*⟩} Break ligatures and use upright apostrophes in HTML tags.

`\protect` is in case the tag appears in TOC, LOF, LOT.

```

\newcommand*{\LWR@htmltagc}[1]{%
{%
\protect\LWR@origttfamily%
\protect\LWR@origtextless#1\protect\LWR@origtextgreater%
}%
}
```

Env `LWR@nestspan` Disable `minipage`, `\parbox` inside a `<span>`.

⚠ `\begin{LWR@nestspan}` must follow the opening `<span>` tag to allow a paragraph to start if the span is at the beginning of a new paragraph.

⚠ `\end{LWR@nestspan}` must follow the `/span` or an extra `<p>` may appear.

```
\newenvironment*{LWR@nestspan}
{%
\addtocounter{LWR@spandepth}{1}%
\RenewDocumentEnvironment{minipage}{O{t} o O{t} m}{}{}%
}%
{\addtocounter{LWR@spandepth}{-1}}
```

`\LWR@htmlspan` {⟨*tag*⟩} {⟨*text*⟩}

⚠ `\LWR@spandepth` is used to ensure that paragraph tags are not generated inside a span. The exact sequence of when to add and subtract the counter is important to correctly handle the paragraph tags before and after the span.

```
\NewDocumentCommand{\LWR@htmlspan}{m +m}{%
\LWR@ensuredoingapar%
\LWR@htmltagc{#1}%
\begin{LWR@nestspan}%
#2%
\LWR@htmltagc{/#1}%
\end{LWR@nestspan}%
}
```

\LWR@htmlspanclass {⟨*class*⟩} [⟨*style*⟩] {⟨*text*⟩}

```
\NewDocumentCommand{\LWR@htmlspanclass}{m o +m}{%
\LWR@ensuredoingapar%
\LWR@subhtmlelementclass{span}{#1}[#2]%
\begin{LWR@nestspan}%
#3%
\LWR@htmltagc{/span}%
\end{LWR@nestspan}%
}
```

\LWR@htmltag {⟨*tag*⟩}

Print an HTML tag: <tag>

```
\newcommand*{\LWR@htmltagb}[1]{%
\LWR@htmltagc{#1}%
\endgroup%
}

\newcommand*{\LWR@htmltag}{%
\begingroup\catcode`\_=12
\LWR@htmltagb%
}
```

35.3 Block tags and comments

In the following, \origttfamily breaks ligatures, which may not be used for HTML codes:

\LWR@htmlopencomment
\LWR@htmlclosecomment

```
\newcommand*{\LWR@htmlopencomment}{%
{\LWR@origttfamily\LWR@origtextless{}!{-}{-}}%
}

\newcommand*{\LWR@htmlclosecomment}{%
{\LWR@origttfamily{-}{-}\LWR@origtextgreater{}}%
}
```

\LWR@htmlcomment {⟨*comment*⟩}

```
\newcommand{\LWR@htmlcomment}[1]{%
\LWR@htmlopencomment{}%
{%
```

```
\LWR@origttfamily% break ligatures
#1%
}%
\LWR@htmlclosecomment{}}
```

\LWR@htmlblockcomment {⟨*comment*⟩}

```
\newcommand{\LWR@htmlblockcommentb}[1]
{\LWR@stoppars\LWR@htmlcomment{#1}\LWR@startpars\endgroup}

\newcommand{\LWR@htmlblockcomment}
{%
\begingroup\catcode`\_=12%
\LWR@htmlblockcommentb%
}
```

\LWR@htmlblocktag {⟨*tag*⟩} print a stand-alone HTML tag

```
\newcommand*{\LWR@htmlblocktag}[1]{%
\LWR@stoppars%
\LWR@htmltag{#1}%
\LWR@startpars%
}
```

35.4 Div class and element class

\LWR@subhtmlelementclass {⟨*element*⟩} {⟨*class*⟩} [⟨*style*⟩]

Factored and reused in several places.

```
\NewDocumentCommand{\LWR@subhtmlelementclass}{m m o}{%
\IfValueTF{#3}%
{% option
\ifthenelse{\equal{#3}{}}%
{\LWR@htmltag{#1 class="#2"}}% empty option
{\LWR@htmltag{#1 class="#2" style="#3"}}% non-empty option
}% option
{\LWR@htmltag{#1 class="#2"}}% no option
}
```

\LWR@htmlelementclass {⟨*element*⟩} {⟨*class*⟩} [⟨*style*⟩]

```
\NewDocumentCommand{\LWR@htmlelementclass}{m m o}{%
\LWR@stoppars%
\LWR@subhtmlelementclass{#1}{#2}[#3]%
```

```
\LWR@startpars%
}
```

\LWR@htmlelementclassend {⟨*element*⟩} {⟨*class*⟩}

```
\newcommand*{\LWR@htmlelementclassend}[2]{%
\LWR@stoppars%
\LWR@htmltag{/#1}%
\ifbool{HTMLDebugComments}{%
\LWR@htmlcomment{End of #1 ``#2''}%
}{}%
\LWR@startpars%
}
```

\LWR@htmldivclass {⟨*class*⟩} [⟨*style*⟩]

```
\NewDocumentCommand{\LWR@htmldivclass}{m o}{%
\LWR@htmlelementclass{div}{#1}[#2]%
}
```

\LWR@htmldivclassend {⟨*class*⟩}

```
\newcommand*{\LWR@htmldivclassend}[1]{%
\LWR@htmlelementclassend{div}{#1}%
}
```

35.5 Single-line elements

A single-line element, without a paragraph tag for the line of text:

\LWR@htmlelementclassline {⟨*element*⟩} {⟨*class*⟩} [⟨*style*⟩] {⟨*text*⟩}

```
\NewDocumentCommand{\LWR@htmlelementclassline}{m m o +m}{%
\LWR@stoppars
\LWR@subhtmlelementclass{#1}{#2}[#3]%
#4%
\LWR@htmltag{/#1}
\LWR@startpars
}
```

35.6 HTML5 semantic elements

`\LWR@htmlelement` {⟨*element*⟩}

```
\newcommand*{\LWR@htmlelement}[1]{%
\LWR@htmlblocktag{#1}
}
```

`\LWR@htmlelementend` {⟨*element*⟩}

```
\newcommand*{\LWR@htmlelementend}[1]{%
\LWR@stoppars
\LWR@htmltag{/#1}
\LWR@startpars
}

\end{warpHTML}
```

35.7 High-level block and inline classes

These are high-level commands which allow the creation of arbitrary block or inline sections which may be formatted with CSS.

For other direct-formatting commands, see section 68.

Env `BlockClass` {⟨*class*⟩} [⟨*style*⟩] High-level interface for div classes.

Ex: `\begin{BlockClass}{class}` text `\end{BlockClass}`

for HTML output:

```
\begin{warpHTML}
\NewDocumentEnvironment{BlockClass}{m o}
{
\LWR@htmldivclass{#1}[#2]
}
{
\LWR@htmldivclassend{#1}
}
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\NewDocumentEnvironment{BlockClass}{m o}{}{}
\end{warpprint}
```

`\BlockClassSingle` {⟨*class*⟩} {⟨*text*⟩} A single-line `<div>`, without a paragraph tag for the line of text.

for HTML output:
```
\begin{warpHTML}
\newcommand{\BlockClassSingle}[2]{%
\LWR@htmlelementclassline{div}{#1}{#2}%
}
\end{warpHTML}
```

for PRINT output:
```
\begin{warpprint}
\newcommand{\BlockClassSingle}[2]{#2}
\end{warpprint}
```

`\InlineClass` {⟨*class*⟩} [⟨*style*⟩] {⟨*text*⟩} High-level interface for inline span classes.

for HTML output:
```
\begin{warpHTML}
\NewDocumentCommand{\InlineClass}{m o +m}{%
\LWR@htmlspanclass{#1}[#2]{#3}%
}
\end{warpHTML}
```

for PRINT output:
```
\begin{warpprint}
\NewDocumentCommand{\InlineClass}{m o +m}{#3}
\end{warpprint}
```

35.8 Closing HTML tags

for HTML output:
```
\begin{warpHTML}
```

Sections H1, H2, etc. do not need a closing HTML tag, but we add a comment for readability:

```
\newcommand*{\LWR@printclosepart}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing part}}{}}
\newcommand*{\LWR@printclosechapter}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing chapter}}{}}
\newcommand*{\LWR@printclosesection}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing section}}{}}
\newcommand*{\LWR@printclosesubsection}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing subsection}}{}}
\newcommand*{\LWR@printclosesubsubsection}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing subsubsection}}{}}
\newcommand*{\LWR@printcloseparagraph}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing paragraph}}{}}
\newcommand*{\LWR@printclosesubparagraph}
    {\ifbool{HTMLDebugComments}{\LWR@htmlcomment{Closing subparagraph}}{}}
```

Lists require closing HTML tags:

```
\newcommand*{\LWR@printcloselistitem}
    {\LWR@htmltag{/li}}
\newcommand*{\LWR@printclosedescitem}
    {\LWR@htmltag{/dd}}
\newcommand*{\LWR@printcloseitemize}
    {\LWR@htmltag{/ul}}
\newcommand*{\LWR@printcloseenumerate}
    {\LWR@htmltag{/ol}}
\newcommand*{\LWR@printclosedescription}
    {\LWR@htmltag{/dl}}

\end{warpHTML}
```

36 Paragraph handling

These commands generate the HTML paragraph tags when allowed and required.

Paragraph tags are or are not allowed depending on many conditions. Section 37 has high-level commands which allow paragraph-tag generation to start/stop. Even when allowed (`\LWR@doingstartpars`), tags are not generated until a LaTeX paragraph is being used (`\LWR@doingapar`). `LWR@lateximagedepth` is used to prevent nesting tags inside a `lateximage`. `LWR@spandepth` is used to prevent nesting paragraph tags inside a paragraph, which became important inside `\fbox` commands and other spans.

for HTML output:

```
\begin{warpHTML}
```

Ctr `LWR@spandepth` Do not create paragraph tags inside of an HTML span.

```
\newcounter{LWR@spandepth}
\setcounter{LWR@spandepth}{0}
```

Bool `LWR@doingstartpars` Tells whether paragraphs may be generated.

```
\newbool{LWR@doingstartpars}
\boolfalse{LWR@doingstartpars}
```

Bool `LWR@doingapar` Tells whether have actually generated and are currently processing paragraph text.

```
\newbool{LWR@doingapar}
\global\boolfalse{LWR@doingapar}
```

`\LWR@ensuredoingapar` If are about to print something visible, and if allowed to start a new paragraph, ensure that are `LWR@doingapar`, so that paragraph tags are placed:

```
\newcommand*{\LWR@ensuredoingapar}{%
\ifbool{LWR@doingstartpars}%
{\global\booltrue{LWR@doingapar}}%
{}%
}
```

\LWR@openparagraph

```
\newcommand*{\LWR@openparagraph}
{%
```

See if paragraph handling is enabled:

```
\ifbool{LWR@doingstartpars}%
{% handling pars
```

See if have already started a `lateximage` or a `<span>`. If so, do not generate nested paragraph tags.

```
\ifthenelse{%
\cnttest{\value{LWR@lateximagedepth}}{>}{0} \OR%
\cnttest{\value{LWR@spandepth}}{>}{0}%
}% nested par tags?
```

If so: Do nothing if already started a `lateximage` page. Cannot nest a `lateximage`. Also do nothing if already inside a `<span>`. Do not nest paragraph tags inside a `<span>`.

```
{}% no nested par tags
```

Else: No `lateximage` or `<span>` has been started yet, so it's OK to generate paragraph tags.

```
{% yes nest par tags
\LWR@htmltagc{\LWR@tagregularparagraph}%
```

Manually indent item list labels to avoid left margin intrustion:

LaTeX default list environments use `\@itemdepth` and `\@enumdepth`, but lwarp uses the enumitem package, which uses `\@listdepth`.

See if are nested inside an item list:

```
\ifnumcomp{\@listdepth}{>}{0}%
{%
```

If so, leave some horizontal room in the LaTeX PDF output for list labels:

```
\LWR@orighspace{1in}%
}{}%
```

Now have started a paragraph.

```
\global\booltrue{LWR@doingapar}%
```

At the endof each paragraph, generate closing tag and do regular /par stuff. (Attempting to use the everyhook cr hook for \LWR@closeparagraph does not work well.)

```
\let\par\LWR@closeparagraph%
}% end of yes nest par tags
}% end of handling pars
{}% not handling pars
}
```

\LWR@closeparagraph

```
\newcommand*{\LWR@closeparagraph}
{%
```

See if paragraph handling is enabled:

```
\ifbool{LWR@doingapar}%
```

If currently in paragraph mode:

```
{% handling pars
```

See if already started a lateximage or a :

```
\ifthenelse{%
\cnttest{\value{LWR@lateximagedepth}}{>}{0} \OR%
\cnttest{\value{LWR@spandepth}}{>}{0}%
}%
```

Do nothing if already started a lateximage or a , but add a parbreak if in a span but not a lateximage.

```
{% no nested par tags
\ifthenelse{%
\cnttest{\value{LWR@spandepth}}{>}{0}%
\AND%
\cnttest{\value{LWR@lateximagedepth}}{=}{0}%
}%
{%
\ifbool{LWR@intabularmetadata}{}{\unskip\LWR@htmltagc{br /}}%
```

```
}%
{}%
}% no nested par tags
```

If have not already started a `lateximage` or a `<span>`:

```
{% yes nest par tags
```

Print a closing tag:

```
\unskip%
\LWR@htmltagc{/\LWR@tagregularparagraph}%
```

No longer doing a paragraph:

```
\global\boolfalse{LWR@doingapar}%
% Disable the special \env{minipage} \& \cs{hspace} interaction
% until a new minipage is found:
%    \begin{macrocode}
\global\boolfalse{LWR@minipagethispar}%
}% end of yes nest par tags
}% end of handling pars
```

Add a parbreak if in a span, but not in a table outside a row:

```
{% not handling pars
\ifthenelse{\cnttest{\value{LWR@spandepth}}{>}{0}}%
{\ifbool{LWR@intabularmetadata}{}{\unskip\LWR@htmltagc{br /}}}%
{}%
}% not handling pars
```

Finish with regular paragraph processing

```
\LWR@origpar%
}
```

```
\end{warpHTML}
```

37 Paragraph start/stop handling

These commands allow/disallow the generation of HTML paragraph tags.

Section 36 has the commands which actually generate the tags.

The `everyhook` package is used to generate the opening paragraph tags. The closing tags are generated by `\par`.

for HTML output:
```
\begin{warpHTML}
```

`\LWR@startpars` Begin handling HTML paragraphs. This allows an HTML paragraph to start, but one has not yet begun.

```
\newcommand*{\LWR@startpars}%
{%
```

See if currently handling HTML paragraphs:

```
\ifbool{LWR@doingstartpars}%
```

If already in paragraph mode, do nothing.

```
{}%
```

If not currently in paragraph mode:

```
{%
```

At the start of each paragraph, generate an opening tag:

```
\PushPreHook{par}{\LWR@openparagraph}%
```

At the end of each paragraph, generate closing tag and do regular `/par` actions:

```
\let\par\LWR@closeparagraph

}% an intentionally blank line
```

Are now handling paragraphs, but have not yet actually started one:

```
\global\setbool{LWR@doingstartpars}{true}%
```

No <par> tag yet to undo:

```
\global\boolfalse{LWR@doingapar}%
}
```

`\LWR@stoppars` Stop handling HTML paragraphs. Any currently open HTML paragraph is closed, and no more will be opened.

```
\newcommand*{\LWR@stoppars}%
{%
```

See if currently handling HTML paragraphs:

```
\ifbool{LWR@doingapar}%
```

if currently in an HTML paragraph:

```
{%
```

Print a closing tag:

```
\unskip%
\LWR@htmltagc{/\LWR@tagregularparagraph}%
```

No longer have an open HTML paragraph:

```
\global\boolfalse{LWR@doingapar}%
```

Disable the special `minipage` & `\hspace` interaction until a new minipage is found:

```
\global\boolfalse{LWR@minipagethispar}

}% an intentionally blank line
```

If was not in an HTML paragraph:

```
{}%
```

See if currently allowing HTML paragraphs:

```
\ifbool{LWR@doingstartpars}%
```

If so: clear the `par` hook to no longer catch paragraphs:

```
{%
\ClearPreHook{par}%
}%
```

Else: do nothing

```
{}%
```

no longer in paragraph mode

```
\global\setbool{LWR@doingstartpars}{false}%
```

no `<p>` tag to undo:

```
\global\boolfalse{LWR@doingapar}%
}

\end{warpHTML}
```

38 Page headers and footers

for HTML & PRINT:
```
\begin{warpall}
```

In the following, catcode is manually changes back and forth without groups, since new macros are being defined which must not be contained within the groups.

```
\newcommand{\LWR@firstpagetop}{} % for the home page alone
\newcommand{\LWR@pagetop}{} % for all other pages
\newcommand{\LWR@pagebottom}{}

\newcommand{\LWR@setfirstpagetopb}[1]{%
\renewcommand{\LWR@firstpagetop}{#1}
\catcode`\_=8
}
```

`\HTMLFirstPageTop` {⟨*text and logos*⟩}

```
\newcommand{\HTMLFirstPageTop}{%
\catcode`\_=12
\LWR@setfirstpagetopb
}

\newcommand{\LWR@setpagetopb}[1]{%
\renewcommand{\LWR@pagetop}{#1}
\catcode`\_=8
}
```

`\HTMLPageTop` {⟨*text and logos*⟩}

```
\newcommand{\HTMLPageTop}{%
\catcode`\_=12
\LWR@setpagetopb
}

\newcommand{\LWR@setpagebottomb}[1]{%
\renewcommand{\LWR@pagebottom}{#1}
\catcode`\_=8
}
```

`\HTMLPageBottom` {⟨*text and logos*⟩}

```
\newcommand{\HTMLPageBottom}{%
\catcode`\_=12
\LWR@setpagebottomb
}
```

```
\end{warpall}
```

39 CSS

for HTML output:

```
\begin{warpHTML}
```

`\LWR@currentcss` The CSS filename to use. This may be changed mid-document using `\CSSFilename`, allowing different CSS files to be used for different sections of the document.

```
\newcommand*{\LWR@currentcss}{lwarp.css}
```

`\CSSFilename` {⟨*new-css-filename.css*⟩} Assigns the CSS file to be used by the following HTML pages.

```
\newcommand*{\LWR@newcssb}[1]{%
\renewcommand*{\LWR@currentcss}{#1}
\catcode`\_=8
}

\newcommand*{\CSSFilename}{
\catcode`\_=12
\LWR@newcssb
}
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\newcommand*{\CSSFilename}[1]{}
\end{warpprint}
```

40 HTML meta description and author

for HTML & PRINT:

```
\begin{warpall}
```

`\HTMLAuthor` {⟨*authorname*⟩} The author to place into an HTML meta tag.

```
\newcommand{\theHTMLAuthor}{\theauthor}

\newcommand{\HTMLAuthor}[1]{\renewcommand{\theHTMLAuthor}{#1}}

\end{warpall}
```

for HTML & PRINT:
```
\begin{warpall}
```

This is placed inside an HTML meta tag at the start of each file. This may be changed mid-document using `\HTMLDescription`, allowing different HTML descriptions to be used for different sections of the document.

⚠ Do not use double quotes, and do not exceed 150 characters.

`\HTMLDescription` {⟨*New* HTML *meta description.*⟩} Assigns the HTML file's description meta tag.

```
\newcommand{\LWR@currentHTMLDescription}{}

\newcommand{\HTMLDescription}[1]{%
\renewcommand{\LWR@currentHTMLDescription}{#1}
}

\end{warpall}
```

41 Footnotes

lwarp uses native LaTeX footnote code, although with its own `\box` to avoid the LaTeX output routine. The usual functions work as-is.

Several kinds of footnotes are used: in a regular page, in a minipage, or as thanks in the titlepage. Each of these is handle differently.

41.1 Regular page footnotes

In HTML documents, footnotes are placed at the bottom of the web page using the LaTeX box `\LWR@footnotes`. Using this instead of the original `\footins` box avoids having footnotes be printed by the output routine, since footnotes should be printed per HTML page instead of per PDF page.

See section 41.4 for the implementation.

41.2 Minipage footnotes

See section 67.2 for minipage footnotes.

41.3 Titlepage thanks

See section 48.6 for titlepage footnotes.

41.4 Regular page footnote implementation

for HTML output:
```
\begin{warpHTML}
```

Patch LaTeX footnotes to use a new \box for lwarp footnotes.

```
\newbox\LWR@footnotes
```

Much of the following has unneeded print-mode formatting removed.

```
\long\def\@makefntext#1{\textsuperscript{\@thefnmark} #1}

\def\@makefnmark{\hbox{\textsuperscript{\@thefnmark}}}
```

Footnotes may be in regular text, in which case paragraphs are tagged, or in a table data cell, in which case paragraph tags must be added manually.

```
\long\def\@footnotetext#1{%
\global\setbox\LWR@footnotes=\vbox{\unvbox\LWR@footnotes%
\protected@edef\@currentlabel{%
\csname p@footnote\endcsname\@thefnmark%
}% @currentlabel
\color@begingroup%
\ifbool{LWR@doingstartpars}{}{\LWR@htmltagc{\LWR@tagregularparagraph}}%
\@makefntext{#1}%
\ifbool{LWR@doingstartpars}{\par}{\LWR@htmltagc{/\LWR@tagregularparagraph}}%
\color@endgroup%
}% vbox
}%

\long\def\@mpfootnotetext#1{%
\global\setbox\@mpfootins\vbox{%
\unvbox\@mpfootins
\reset@font\footnotesize
\hsize\columnwidth
\@parboxrestore
\protected@edef\@currentlabel
{\csname p@mpfootnote\endcsname\@thefnmark}%
\color@begingroup
\@makefntext{%
\ignorespaces#1%
}%
```

Don't add the closing paragraph tag if are inside a `lateximage`:

```
\ifthenelse{\cnttest{\value{LWR@lateximagedepth}}{>}{0}}%
{}%
{\LWR@htmltagc{/\LWR@tagregularparagraph}}%

\color@endgroup}%
}
```

Enclose the footnotes in a class, print, then clear:

```
\newcommand*{\LWR@printpendingfootnotes}{%
\ifvoid\LWR@footnotes\else
\LWR@forcenewpage
\begin{BlockClass}{footnotes}
\LWR@origmedskip
\unvbox\LWR@footnotes
\setbox\LWR@footnotes=\vbox{}
\end{BlockClass}
\fi
}
```

Used to print footnotes before sections only if formatting for an EPUB or word processor:

```
\newcommand*{\LWR@epubprintpendingfootnotes}{%
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}%
{\LWR@printpendingfootnotes}%
{}%
}

\end{warpHTML}
```

42 Marginpars

for HTML output:

```
\begin{warpHTML}
```

`\marginpar` [⟨*left*⟩] {⟨*right*⟩}

```
\renewcommand{\marginpar}[2][]{%
\LWR@htmlspanclass{marginpar}{#2}%
}

\end{warpHTML}
```

43 Splitting HTML files

- Files are split according to `FileDepth` and `CombineHigherDepths`.
- Filenames are sanitized by `\LWR@filenamenoblanks`.
- `\LWR@newhtmlfile` finishes an HTML page, adds a comment to tell where and how to split the file, then starts a new HTML page.

for HTML & PRINT:

```
\begin{warpall}
```

Ctr `FileDepth` {⟨*section depth*⟩} determines how deeply to break into new HTML files, similar to `tocdepth`. The default of -5 produces one large HTML file.

```
\newcounter{FileDepth}
\setcounter{FileDepth}{-5}
```

Bool `CombineHigherDepths` Combile higher-level sections together into one file?

```
\newbool{CombineHigherDepths}
\booltrue{CombineHigherDepths}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

`\LWR@thisfilename` The currently-active filename or number.

```
\newcommand*{\LWR@thisfilename}{}
```

`\LWR@thisnewfilename` The filename being sanitized.

```
\newcommand*{\LWR@thisnewfilename}{}
```

`\LWR@filenamenoblanks` {⟨*filename*⟩}

Convert blanks into dashes, removes short words, store result in `\LWR@thisfilename`.

⚠ Be sure that this does not result in filename collisions! Use the optional TOC caption entry parameter for formatting. Remember to `\protect` LaTeX commands which appear in section names and TOC captions.

```
\newcommand*{\LWR@filenamenoblanks}[1]{%
\begingroup
```

Locally temporarily disable direct-formatting commands, not used in filenames:

```
\renewcommand*{\HTMLunicode}[1]{}
\renewcommand*{\HTMLentity}[1]{##1}
\renewcommand*{\LWR@htmltagc}[1]{}
\DeclareExpandableDocumentCommand{\InlineClass}{m o m}{##3}
```

Ampersand becomes “and”, which is a short word and is then removed from the filename.

```
\renewcommand*{\&}{and}
\renewcommand{\textit}[1]{##1}
\renewcommand{\textsc}[1]{##1}
\renewcommand{\textsl}[1]{##1}
\renewcommand{\textbf}[1]{##1}
\renewcommand{\texttt}[1]{##1}
\renewcommand{\textsf}[1]{##1}
\renewcommand{\textrm}[1]{##1}
\renewcommand{\textsuperscript}[1]{##1}
\renewcommand{\textsubscript}[1]{##1}
```

Replaces common symbols and short words with hyphens:

```
\edef\LWR@thisnewfilename{#1}
\fullexpandarg
```

Convert spaces into hyphens:

```
\StrSubstitute{\LWR@thisnewfilename}{ }{-}[\LWR@thisnewfilename]
```

Convert punctutation into hyphens:

```
\StrSubstitute{\LWR@thisnewfilename}{,}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{’}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}%
{\LWR@origampersand}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{+}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{,}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{/}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{:}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{;}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{=}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{?}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{@}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{"}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}%
{\textless}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}%
{\textgreater}{-}[\LWR@thisnewfilename]
```

```
\StrSubstitute{\LWR@thisnewfilename}{\#}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{\%}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{\{}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{\}}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{|}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}%
{\textbackslash}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{^}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{~{}}{-}[\LWR@thisnewfilename]
%     "~{}" for babel
\StrSubstitute{\LWR@thisnewfilename}{[}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{]}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{`}{-}[\LWR@thisnewfilename]
```

Convert short words:

```
\StrSubstitute{\LWR@thisnewfilename}{-s-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-S-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-a-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-A-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-an-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-AN-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-to-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-TO-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-by-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-BY-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-of-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-OF-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-and-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-AND-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-for-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-FOR-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-the-}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{-THE-}{-}[\LWR@thisnewfilename]
```

Convert multiple hyphens:

```
\StrSubstitute{\LWR@thisnewfilename}{-----}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{----}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{---}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{--}{-}[\LWR@thisnewfilename]
\StrSubstitute{\LWR@thisnewfilename}{--}{-}[\LWR@thisnewfilename]
%     emdash
\StrSubstitute{\LWR@thisnewfilename}{-}{-}[\LWR@thisnewfilename]
%     endash
\global\let\LWR@thisfilename\LWR@thisnewfilename% return a global result
\endgroup
}
```

`\LWR@newhtmlfile` {⟨*section name*⟩}

Finishes the current HTML page with footnotes, footer, navigation, then starts a new HTML page with an HTML comment telling where to split the page and what the new filename and CSS are, then adds navigation, side TOC, header, and starts the text body.

```
\newcommand*{\LWR@newhtmlfile}[1]{
```

At the bottom of the ending file:

```
\LWR@htmlelementclassend{section}{textbody}

\LWR@printpendingfootnotes

```

No footer between files if EPUB:

```
\ifbool{FormatEPUB}
{}
{
\LWR@htmlelement{footer}

\LWR@pagebottom

\LWR@htmlelementend{footer}
}
```

No bottom navigation if are finishing the home page or formatting for EPUB or a word-processor.

```
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}
{}
{\ifnumcomp{\value{LWR@htmlfilenumber}}{>}{0}{\LWR@botnavigation}{}}
```

End of this HTML file:

```
\LWR@stoppars
\LWR@htmltag{/body}\LWR@orignewline
\LWR@htmltag{/html}\LWR@orignewline
\LWR@orignewpage

\addtocounter{LWR@htmlfilenumber}{1}%
```

If using a filename, create a version without blanks. The filename without blanks will be placed into `\LWR@thisfilename`. If not using a filename, the file number will be used instead.

```
\ifbool{FileSectionNames}%
{\LWR@filenamenoblanks{#1}}
{\renewcommand*{\LWR@thisfilename}{\theLWR@htmlfilenumber}}
```

Include an HTML comment to instruct lwarpmk where to split the files apart. Uses pipe-separated fields for `split_html.gawk`. Uses monospaced font with ligatures disabled for everything except the title.

```
\LWR@htmlblockcomment{%
|Start file|%
\LWR@htmlsectionfilename{\LWR@thisfilename}|%
}
```

At the top of the starting file:

```
\LWR@stoppars

\LWR@filestart{ -- #1}% there is an EMdash in front of the #1

```

No navigation between files if formatting for an EPUB or word processor:

```
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}
{}
{\LWR@topnavigation}

```

No header if between files if formatting for an EPUB or word processor:

```
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}
{}
{
\LWR@htmlelement{header}

\LWR@pagetop

\LWR@htmlelementend{header}
}

```

Print title only if there is one. Skip if formatting for an EPUB or word processor:

```
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}
{}
{\ifcsvoid{thetitle}{}{\LWR@printthetitle}}

```

No sideTOC if formatting for an EPUB or word processor:

```
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}
{}
{\LWR@sidetoc}

```

Start of the `<textbody>`:

```
\LWR@htmlelementclass{section}{textbody}

```

Keep paragraph tags disabled for now:

```
\LWR@stoppars

```

Track the page numbers:

```
\setcounter{LWR@latestautopage}{\value{page}}
}

\end{warpHTML}
```

44 Sectioning

Sectioning and cross-references have been emulated from scratch, rather than try to patch several layers of existing LaTeX code and packages. Formatting is handled by CSS, so the emulated code has much less work to do than the print versions.

Unicode ⚠ Section names and the resulting filenames with accented characters are partially supported, depending on the ability of `pdflatex` to generate characters and `pdftotext` to read them. If extra symbols appear in the text, it may be that `pdflatex` is actually producing a symbol over or under a character, resulting in `pdftotext` picking up the accent symbol separately.

XeLaTeX and LuaLaTeX directly support accented section and file names.

for HTML output:

```
\begin{warpHTML}
```

44.1 Book class commands

`\mainmatter` ⚠ Declare the main matter section of the document. Does not reset the page number, which must be consecutive arabic numbers for the HTML conversion.

```
\newbool{LWR@mainmatter}
\DeclareDocumentCommand{\mainmatter}{}{%
\booltrue{LWR@mainmatter}%
}
```

`\frontmatter` Declare the front matter section of the document, using arabic numbering for the internal numbering. Does not reset the page number.

```
\DeclareDocumentCommand{\frontmatter}{}{%
\boolfalse{LWR@mainmatter}%
}
```

`\backmatter` Declare the back matter section of the document. Does not reset the page number.

```
\DeclareDocumentCommand{\backmatter}{}{%
\boolfalse{LWR@mainmatter}
}
```

44.2 Sectioning support macros

`\LWR@sectionnumber` `{`⟨*section type*⟩`}`

Typeset a section number and its trailing space with CSS formatting:

```
\newcommand*{\LWR@sectionnumber}[1]{%
\InlineClass{sectionnumber}{#1} %
}
```

`autosec` A tag used by the TOC and index.

`\LWR@createautosec` `{`⟨*section type*⟩`}`

Create an autosection tag.

```
\newcommand*{\LWR@createautosec}[1]{%
\LWR@htmltag{#1 id="autosec-\thepage"{}}%
}
```

`\LWR@pushoneclose` `{`⟨*depth*⟩`}` `{`⟨*printclose*⟩`}` Stacks the new sectioning level's closing tag, to be used when this section is closed some time later.

⚠ `\LWR@stoppars` must be executed first.

```
\NewDocumentCommand{\LWR@pushoneclose}{m m}{\pushclose{#2}{#1}}
```

`\LWR@startnewdepth` {⟨*depth*⟩} {⟨*printclose*⟩}

Closes currently stacked tags of a lesser level, then opens the new nesting level by saving this new sectioning level's closing tag for later use.

⚠ `\LWR@stoppars` must be executed first.

```
\NewDocumentCommand{\LWR@startnewdepth}{m m}{%
```

Close any stacked sections up to this new one.

```
\LWR@closeprevious{#1}%
```

Push a new section depth:

```
\LWR@pushoneclose{#1}{#2}%
}
```

Ctr `LWR@prevFileDepth` Remembers the previous `LWR@FileDepth`.

Initialized to a deep level so that any section will trigger a new HTML page after the home page.

```
\newcounter{LWR@prevFileDepth}
\setcounter{LWR@prevFileDepth}{\LWR@depthsubparagraph}
```

`\LWR@section` * [⟨*TOC name*⟩] {⟨*name*⟩} {⟨*sectiontype*⟩}

The common actions for the high-level sectioning commands.

```
\DeclareDocumentCommand{\LWR@section}{m m m m}{%
\LWR@traceinfo{LWR@section}%
\LWR@stoppars%
```

Cancel special `minipage` horizontal space interaction:

```
\global\boolfalse{LWR@minipagethispar}%
```

Start a new HTML file if not starred, and is a shallow sectioning depth:

```
\LWR@traceinfo{LWR@section: testing whether to start a new HTML file}%
\IfBooleanTF{#1}{% starred
```

Generate a new LaTeX page so that TOC and index page number points to the section:

```
\LWR@orignewpage%

```

```
}{% not starred
\ifthenelse{%
\cnttest{\csuse{LWR@depth#4}}{<=}{\value{FileDepth}}%
\AND%
\(%
\NOT\boolean{CombineHigherDepths}\OR%
\cnttest{\csuse{LWR@depth#4}}{<=}{\value{LWR@prevFileDepth}}%
\)%
}%
```

If so: start a new HTML file:

```
{% new file
\LWR@traceinfo{LWR@section: new HTML file}%
```

See if there was an optional TOC name entry:

```
\IfNoValueTF{#2}%
```

If no optional entry

```
{\LWR@newhtmlfile{#3}}%
```

If yes an optional entry

```
{\LWR@newhtmlfile{#2}}%
}% new file
```

Else: No new HTML file:

```
{% not new file
```

Generate a new LaTeX page so that TOC and index page number points to the section:

```
\LWR@orignewpage%

}% not new file
}% not starred
```

Remember this section's name for `\nameref`:

```
\LWR@traceinfo{LWR@section: about to LWR@setlatestname}%
\IfValueTF{#2}{\LWR@setlatestname{#2}}{\LWR@setlatestname{#3}}%
```

Print an opening comment with the level and the name; ex: "section" "Introduction"

```

```

```
\ifbool{HTMLDebugComments}{%
\LWR@htmlcomment{Opening #4 ``#3''{}}
}{}

```

For inline sections paragraph and subparagraph, start a new paragraph now:

```
\ifthenelse{%
\cnttest{\csuse{LWR@depth#4}}{>=}{\LWR@depthparagraph}%
}%
{\LWR@startpars}
{}
```

Create the opening tag with an autosec:

```
\LWR@createautosec{\csuse{LWR@tag#4}}%
```

If not starred, step counter and add to TOC:

```
\IfBooleanTF{#1}%
{}% starred
{% not starred
```

Only add a numbered TOC entry if section number is not too deep:

```
\ifthenelse{%
\cnttest{\csuse{LWR@depth#4}}{<=}{\value{secnumdepth}}%
}%
{% if secnumdepth
```

If in the main matter, step the counter and add the TOC entry. For `article` class, lwarp assumes that all is mainmatter.

```
\LWR@traceinfo{LWR@section: about to test main matter}%
\ifbool{LWR@mainmatter}%
{%
\LWR@traceinfo{LWR@section: yes mainmatter}%
\refstepcounter{#4}%
```

Add main matter numbered TOC entry with the TOC name or the regular name:

```
\LWR@traceinfo{LWR@section: about to addcontentsline}%
\addcontentsline{toc}{#4}%
{\protect\LWR@sectionnumber{\csuse{the#4}}%
\IfValueTF{#2}{#2}{#3}}%
\LWR@traceinfo{LWR@section: finished addcontentsline}%
}% end of if main matter
```

If not main matter, add unnumbered TOC name or regular name:

```
{% not main matter
\LWR@traceinfo{LWR@section: no main matter}%
\addcontentsline{toc}{#4}{\IfValueTF{#2}{#2}{#3}}%
}% end of not main matter
}% end of secnumdepth
```

Deeper than secnumdepth, so add an unnumbered TOC entry:

```
{%
\addcontentsline{toc}{#4}{\IfValueTF{#2}{#2}{#3}}%
}%
```

For part, print the section type:

```
\ifbool{LWR@mainmatter}%
{%
\ifthenelse{%
\(\cnttest{\csuse{LWR@depth#4}}{<=}%
{\value{secnumdepth}}\) \AND
\(\cnttest{\csuse{LWR@depth#4}}{<=}{\LWR@depthpart}\)
}%
{\csuse{#4name}~{}}%
{}%
```

Print the section number:

```
\LWR@traceinfo{LWR@section: about to print section number}%
\ifthenelse{%
\cnttest{\csuse{LWR@depth#4}}{<=}{\value{secnumdepth}}%
}%
{\protect\LWR@sectionnumber{\csuse{the#4}}}%
{}%
\LWR@traceinfo{LWR@section: finished print section number}%
}{}%
}% end of not starred
```

Print the section name:

```
#3
```

close the heading tag, such as /H2

```
\LWR@htmltag{\csuse{LWR@tag#4end}}%
```

Generate a LaTeX label:

```
\label{autopage-\thepage}%
```

Start paragraph handing unless is an inline paragraph or subparagraph:

```
\ifthenelse{%
\cnttest{\csuse{LWR@depth#4}}{<}{\LWR@depthparagraph}}%
{\LWR@startpars}%
{}%
```

If not starred, remember the previous depth to possibly trigger a new HTML page.

A starred section does not trigger a new HTML page at the beginning of this macro, so it should not affect it here at the end either. This became an issue when a `\listoftables` was tested in the middle of the document. The `\chapter*` for the list was not allowing a new HTML page for the section following it while `CombineHigherDepths` was true.

```
\IfBooleanTF{#1}{}{% not starred
\setcounter{LWR@prevFileDepth}{\csuse{LWR@depth#4}}%
}% not starred
\LWR@traceinfo{LWR@section: done}%
}
```

44.3 \section and friends

`\part` * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\@ifundefined{chapter}
{}
{%
\DeclareDocumentCommand{\part}{s o m}{%
\LWR@epubprintpendingfootnotes%
\LWR@stoppars%

\LWR@startnewdepth{\LWR@depthpart}{\LWR@printclosepart}%

\LWR@section{#1}{#2}{#3}{part}%
}
}
```

`\chapter` * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\@ifundefined{chapter}
{}
{%
\DeclareDocumentCommand{\chapter}{s o m}{%
\LWR@traceinfo{chapter #3}%
\LWR@epubprintpendingfootnotes%
```

```
\LWR@stoppars%
\LWR@startnewdepth{\LWR@depthchapter}{\LWR@printclosechapter}%

\LWR@traceinfo{chapter: about to LWR@section}%
\LWR@section{#1}{#2}{#3}{chapter}%
\LWR@traceinfo{chapter: done}%
}
}
```

\section * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\DeclareDocumentCommand{\section}{s o m}{%
\LWR@epubprintpendingfootnotes%
\LWR@stoppars%

\LWR@startnewdepth{\LWR@depthsection}{\LWR@printclosesection}%

\LWR@section{#1}{#2}{#3}{section}%
}
```

\subsection * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\DeclareDocumentCommand{\subsection}{s o m}{%
\LWR@epubprintpendingfootnotes%
\LWR@stoppars%

\LWR@startnewdepth{\LWR@depthsubsection}{\LWR@printclosesubsection}%

\LWR@section{#1}{#2}{#3}{subsection}%
}
```

\subsubsection * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\DeclareDocumentCommand{\subsubsection}{s o m}{%
\LWR@epubprintpendingfootnotes%
\LWR@stoppars%

\LWR@startnewdepth{\LWR@depthsubsubsection}%
{\LWR@printclosesubsubsection}%

\LWR@section{#1}{#2}{#3}{subsubsection}%
}
```

\paragraph * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\DeclareDocumentCommand{\paragraph}{s o m}{%
\LWR@epubprintpendingfootnotes%
```

```
\LWR@stoppars%

\LWR@startnewdepth{\LWR@depthparagraph}{\LWR@printcloseparagraph}%

\LWR@section{#1}{#2}{#3}{paragraph}%
}
```

\subparagraph * [⟨*TOC name*⟩] {⟨*name*⟩}

```
\DeclareDocumentCommand{\subparagraph}{s o m}{%
\LWR@epubprintpendingfootnotes%
\LWR@stoppars%

\LWR@startnewdepth{\LWR@depthsubparagraph}{\LWR@printclosesubparagraph}%

\LWR@section{#1}{#2}{#3}{subparagraph}%
}
```

```
\end{warpHTML}
```

45 Starting a new file

for HTML & PRINT:

```
\begin{warpall}
```

\HTMLLanguage Default language for the HTML `lang` tag.

```
\newcommand*{\LWR@currentHTMLLanguage}{en-US}

\newcommand*{\HTMLLanguage}[1]{%
\renewcommand*{\LWR@currentHTMLLanguage}{#1}%
}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

\LWR@filestart {⟨*title_suffix*⟩}

Creates the opening HTML tags.

```
\newcommand*{\LWR@filestart}[1]{
```

Locally temporarily disable direct-formatting commands:

```
\begingroup
\renewcommand{\textit}[1]{##1}% not used in filenames
\renewcommand{\textsc}[1]{##1}
\renewcommand{\textsl}[1]{##1}
\renewcommand{\textbf}[1]{##1}
\renewcommand{\texttt}[1]{##1}
\renewcommand{\textsf}[1]{##1}
\renewcommand{\textrm}[1]{##1}
\renewcommand{\textsuperscript}[1]{##1}
\renewcommand{\textsubscript}[1]{##1}
\renewcommand*{\HTMLunicode}[1]{}
\renewcommand*{\HTMLentity}[1]{}
\RenewDocumentCommand{\LWR@htmlspanclass}{m o +m}{##3}
\DeclareExpandableDocumentCommand{\InlineClass}{m o m}{##3}
```

Create the page's HTML header:

```
\LWR@htmltag{!DOCTYPE html}\LWR@orignewline
```

The language is user-adjustable:

```
\LWR@htmltag{html lang="\LWR@currentHTMLLanguage"{}}\LWR@orignewline
```

Start of the meta data:

```
\LWR@htmltag{head}\LWR@orignewline
```

Charset is fixed at UTF-8:

```
\LWR@htmltag{meta charset="UTF-8" /}\LWR@orignewline
```

Author:

```
\ifcsempty{theHTMLAuthor}{}{
\LWR@htmltag{meta name="author" content="\theHTMLAuthor" /}\LWR@orignewline
}
```

lwarp is the generator:

```
\LWR@htmltag{meta name="generator" content="LaTeX lwarp package" /}%
    \LWR@orignewline
```

If there is a description, add it now:

```
\ifdefempty{\LWR@currentHTMLDescription}{}{%
\LWR@htmltag{%
meta name="description" content="\LWR@currentHTMLDescription" /}%
    \LWR@orignewline
}%
```

Mobile-friendly viewport:

```
\LWR@htmltag{meta name="viewport" %
content="width=device-width, initial-scale=1.0" /}%
    \LWR@orignewline
```

IE patch:

```
\LWR@htmltag{!{-}{-}[if lt IE 9]}\LWR@orignewline
\LWR@htmltag{%
script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"{}}%
\LWR@htmltag{/script}\LWR@orignewline
\LWR@htmltag{![endif]{-}{-}}\LWR@orignewline
```

The page's title:

```
\ifcsvoid{thetitle}{}{%
\LWR@htmltag{title}\thetitle#1\LWR@htmltag{/title}\LWR@orignewline%
}%
```

The page's stylesheet:

```
\LWR@htmltag{%
link rel="stylesheet" type="text/css" href="\LWR@currentcss" /}%
\LWR@orignewline
```

Optional MathJax support. The HTML tags must be turned off during the verbatim input, and the paragraph handling which was turned on at the end of verbatim input must be immediately turned off again.

```
\ifbool{mathjax}%
{%
\boolfalse{LWR@verbtags}
\VerbatimInput{lwarp_mathjax.txt}%
\booltrue{LWR@verbtags}
\LWR@stoppars
}% end of mathjax
{}%
```

End of the header:

```
\LWR@htmltag{/head}\LWR@orignewline
```

Start of the body:

```
\LWR@htmltag{body}\LWR@orignewline
\endgroup
}
```

```
\end{warpHTML}
```

46 Starting HTML output

for HTML output:

```
\begin{warpHTML}
```

`\LWR@LwarpStart` Executed at the beginning of the entire document.

```
\newcommand*{\LWR@LwarpStart}
{%
```

If formatting for a word processor, force filedepth to single-file only, force HTML debug comments off.

```
\ifbool{FormatWordProcessor}{%
\setcounter{FileDepth}{-5}%
\boolfalse{HTMLDebugComments}%
}{}
```

Expand and detokenize `\HomeHTMLFilename` and `\HTMLFilename`:

```
\edef\LWR@strresult{\HomeHTMLFilename}
\edef\HomeHTMLFilename{\detokenize\expandafter{\LWR@strresult}}
\edef\LWR@strresult{\HTMLFilename}
\edef\HTMLFilename{\detokenize\expandafter{\LWR@strresult}}
```

Force onecolumn:

```
\LWR@origonecolumn%
```

Reduce chance of line overflow in verbatim environments:

```
\LWR@origscriptsize%
```

In PDF output, don't allow line breaks to interfere with HTML tags:

```
\LWR@origraggedright%
\let\\\LWR@endofline%
```

Spread the lines for `pdftotext` to read them well:

```
\linespread{1.3}%
```

For `pdftotext` to reliably identify paragraph splits:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{2ex}
```

For the `lateximages` record file:

```
\immediate\openout\LWR@file=lateximages.txt
```

Removes space after the caption in the HTML:

```
\setlength{\belowcaptionskip}{-3ex}
```

Redefine the plain page style to be empty when used by index pages:

```
\renewcommand{\ps@plain}{}
```

`\centering` `\raggedleft` `\raggedright`

Not used in the HTML environment:

```
\renewcommand*{\centering}{}
\renewcommand*{\raggedleft}{}
\renewcommand*{\raggedright}{}
```

Plug in some new actions. This is done just before the document start so that they won't be over-written by some other package.

Tabular:

```
\let\LWR@origtabular\tabular
\let\LWR@origendtabular\endtabular
\let\tabular\LWR@tabular
\let\endtabular\endLWR@tabular
```

Float captions:

```
\let\LWR@origcaption\caption
```

Labels: `\ltx@label` is used in amsmath environments and is also patched by cleveref.

Label in HTML

```
\let\LWR@origltx@label\ltx@label
\let\ltx@label\LWR@htmlmathlabel
```

Logos:

```
\let\TeX\LWR@TeX
\let\LaTeX\LWR@LaTeX
\let\LuaTeX\LWR@LuaTeX
\let\LuaLaTeX\LWR@LuaLaTeX
\let\XeTeX\LWR@XeTeX
```

```
\let\XeLaTeX\LWR@XeLaTeX
\let\ConTeXt\LWR@ConTeXt
```

Graphics:

```
\let\rotatebox\LWR@rotatebox
\let\scalebox\LWR@scalebox
\let\reflectbox\LWR@reflectbox
```

Not yet started any paragraph handling:

```
\global\boolfalse{LWR@doingapar}
\global\boolfalse{LWR@doingstartpars}
```

Start a new HTML file and a header:

```
\LWR@filestart{}
\LWR@htmltag{header}\LWR@orignewline
\LWR@startpars
\LWR@firstpagetop
\LWR@stoppars
\LWR@htmltag{/header}\LWR@orignewline
\LWR@htmltag{section class="textbody"{}}
\LWR@origpagestyle{empty}
```

Document and page settings:

```
\mainmatter
\LWR@origpagenumbering{arabic}
```

Set default titlepage thanks footnote marks. See section 48.6.

```
\if@titlepage
   \thanksmarkseries{arabic}
\else
   \thanksmarkseries{fnsymbol}
\fi
```

Initial default patch for fancyvrb:

```
\fvset{frame=none}%
```

The ampersand is redefined active, and acts depending on whether it is inside a tabular.

```
\catcode`\&=\active
```

Allow HTML paragraphs to begin:

```
\LWR@startpars
}
```

```
\end{warpHTML}
```

47 Ending HTML output

for HTML output:

```
\begin{warpHTML}
```

`\LWR@requesttoc` {⟨*boolean*⟩} {⟨*suffix*⟩} Requests that a toc, lof, or lot be generated.

```
\newcommand*{\LWR@requesttoc}[2]{%
\ifbool{#1}
{
\expandafter\newwrite\csuse{tf@#2}
\immediate\openout \csuse{tf@#2} \jobname.#2\relax
}{}
}
```

`\LWR@LwarpEnd` Final stop of all HTML output:

```
\newcommand*{\LWR@LwarpEnd}
{
\LWR@stoppars
\LWR@closeprevious{\LWR@depthfinished}
```

At the bottom of the ending file:

Close the textbody:

```
\LWR@htmlelementclassend{section}{textbody}
```

Print any pending footnotes:

```
\LWR@printpendingfootnotes
```

Create the footer:

```
\LWR@htmlelement{footer}

\LWR@pagebottom

\LWR@htmlelementend{footer}
```

No bottom navigation if are finishing the home page, or if formatting for an EPUB or word processor.

Presumably has a table-of-contents.

```
\ifthenelse{\boolean{FormatEPUB}\OR\boolean{FormatWordProcessor}}
{}
{
\ifnumcomp{\value{LWR@htmlfilenumber}}{>}{0}{\LWR@botnavigation}{}
}
```

```
\LWR@stoppars% final stop of all paragraphs
```

Finish the HTML file:

```
\LWR@htmltag{/body}\LWR@orignewline
\LWR@htmltag{/html}\LWR@orignewline
```

Seems to be required sometimes:

```
\LWR@orignewpage
```

For `lateximage` commands:

```
\immediate\closeout\LWR@file
}
```

```
\end{warpHTML}
```

48 Titles and the **titling** package

Supports and extends the titling package.

Additional functions include `\published` and `\subtitle`, and the `\author` command has an additional `\affiliation` command to provide an affiliation and other additional information for each author in the title page. The affiliation information is removed when using `\theauthor` in the main text.

The titling package maintains the definitions of `\thetitle`, `\theauthor`, etc., after the title has been typeset. These commands are to be used to refer to the document's title and author, etc., in the main text. These definitions have the `\thanks` and `\affiliation` removed, and for author the `\and` is replaced to generate a simple inline list of authors separated by commas.

`\printtitle`, `\printauthor`, etc., are provided for use inside the `titlepage` or `titlingpage` environments, and these retain the `\thanks` and `\affiliation`.

Several additional hooks are provided in addition to titling:

\maketitlehookaa **\maketitlehookaa:** Between "published" and the title.

\maketitlehookaaa **\maketitlehookaaa:** Between the title and the subtitle.

\prepublished **\prepublished:** Before the "published" field.

\postpublished **\postpublished:** After the "published" field.

\presubtitle **\presubtitle:** Before the subtitle.

\postsubtitle **\postsubtitle:** After the subtitle.

\printthanks \printthanks has been added to force the printing of thanks inside a titlingpage environment when \maketitle is not used.

⚠ No footnotes!

Inside a \titlepage or \titlingpage environment, use \thanks for footnotes, do not use \footnote.

At the end of the titlingpage environment, footnote marks are forced to reset to zero.

Inside a titlingpage environment with the article document class, thanks marks will be fnsymbol instead of arabic. arabic is usually used when inside titlepage environments where the title page is on its own page, but is not automatically used inside a titlingpage environment.

To force the thanks marks to be arabic:

```
\begin{titlingpage}
\thanksmarkseries{arabic}
...
```

48.1 Setting the title, etc.

The following provide setting commands for both HTML and print outputs.

\published
\title
\subtitle
\author
\date

\@title, \@subtitle, \@author, etc. store the values as originally assigned, including any \thanks, \and, or \affiliation. These are low-level macros intended to be used by other macros only inside a titlepage or titlingpage, and are used by \maketitle. The author is printed inside a single-column table, which becomes multiple single-column tables if multiples authors are included.

\printpublished
\printtitle
\printsubtitle
\printauthor
\printdate

\printtitle, \printsubtitle, etc. are user-level macros intended to be used in titlepage and titlingpage environments in cases where \maketitle is not desired. These commands preserve the \thanks, etc., and should not be used in the main text. The author is printed inside a single-column table, which becomes multiple single-column tables if multiples authors are included.

\thepublished
\thetitle
\thesubtitle
\theauthor
\thedate

\thetitle, \thesubtitle, \theauthor, etc. are user-level sanitized versions which have removed the \thanks and \affiliation, and \and is changed for inline text usage. The author is printed inline without \affiliation or \thanks, with \and placing commas between multiple authors. Thus, these commands are to be used in the main text whenever the user wishes to refer to the document's title and such. One practical use for this is to place the authors at the bottom of each HTML page, such as:

\HTMLPageBottom {⟨*text*⟩}

```
\HTMLPageBottom{
\begin{center}\textcopyright~2016 \theauthor\end{center}
}
```

\author {⟨*author*⟩}
\and

While using \maketitle, the author is treated as a single-column table and the \and feature finishes the current table then starts a new one for the next author. Each author thus is placed into its own table, and an affiliation may be placed on its own line such as

```
\author{Name \\ Affiliation \and Second Name \\ Second Affiliation}
```

After \maketitle has completed, \theauthor retains the definition of the author, but \and is changed to become a comma and a space, intending to print the authors names separated by spaces. This fails when affiliations are included on their own table rows.

A solution, provide here, is to define a macro \affiliation which during \maketitle starts a new table row and adds the affiliation, but after \maketitle is finished \affiliation is re-defined to throw away its argument, thus printing only the author names when \author is later used inline.

\affiliation {⟨*text*⟩}

Adds the affiliation to the author for use in \maketitle. Nullified when later used for inline use of \theauthor.

for HTML output:

```
\begin{warpHTML}
\newrobustcmd{\affiliation}[1]{\\ \InlineClass{affiliation}{#1}}
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\newrobustcmd{\affiliation}[1]{\\ \textsc{\small#1}}
\end{warpprint}
```

The following are based on the original titling code:

for HTML & PRINT:

```
\begin{warpall}
```

`\author` {⟨*text*⟩}

Redefined to nullify `\affiliation`, etc. before printing the authors inline.

`\@author` retains the entire author with its `\thanks`, while `\theauthor` will have `\thanks` removed and `\and` simplified.

```
\renewcommand{\author}[1]{%
\gdef\@author{#1}
\begingroup
  \renewcommand{\thanks}[1]{}
  \renewcommand{\and}{\unskip, }
  \renewcommand{\thanksmark}[1]{}
  \renewcommand{\thanksgap}[1]{}
  \renewcommand{\affiliation}[1]{}
  \protected@xdef\theauthor{#1}
\endgroup}
```

`\published` {⟨*text*⟩}

```
\newcommand{\published}[1]{%
\gdef\@published{#1}
\begingroup
  \renewcommand{\thanks}[1]{}
  \renewcommand{\thanksmark}[1]{}
  \renewcommand{\thanksgap}[1]{}
  \protected@xdef\thepublished{#1}
\endgroup
}
\newcommand{\@published}{}
\newcommand{\thepublished}{}
```

`\subtitle` {⟨*text*⟩}

```
\newcommand{\subtitle}[1]{%
\gdef\@subtitle{#1}
\begingroup
  \renewcommand{\thanks}[1]{}
  \renewcommand{\thanksmark}[1]{}
```

```
  \renewcommand{\thanksgap}[1]{}
  \protected@xdef\thesubtitle{#1}
\endgroup
}
\newcommand{\@subtitle}{}
\newcommand{\thesubtitle}{}

\end{warpall}
```

48.2 Changes to HTML titlepage and titlingpage

for HTML output:

```
\begin{warpHTML}
```

Env `titlepage` Sets up a titlepage div with a LaTeX PDF minipage inside.

```
\renewenvironment*{titlepage}
{
\LWR@forcenewpage
\BlockClass{titlepage}\LWR@subminipage
}
{\LWR@endsubminipage\endBlockClass}
```

Env `titlingpage`

```
\renewenvironment*{titlingpage}
{%
```

Start an HTML titlepage div:

```
\begin{titlepage}
```

Prepare for a custom version of `\maketitle` inside the `titlingpage`:

```
\LWR@maketitlesetup
\let\maketitle\LWR@titlingmaketitle
}
{
```

At the end of the environment, end the HTML titlepage div:

```
\end{titlepage}
```

Reset the footnote counter:

```
\@bscontmark
}
```

```
\end{warpHTML}
```

for HTML & PRINT:

```
\begin{warpall}
```

`\printthanks` Forces the `\thanks` to be printed.

This is necessary in a `titlingpage` environment when `\maketitle` was not used.

```
\newcommand*{\printthanks}{\@thanks}
```

Env `titlingpage` At the end of the titlingpage for both print and HTML, reset footnote markers to zero.

```
\AtEndEnvironment{titlingpage}{\@bscontmark}
```

```
\end{warpall}
```

48.3 Printing the title, etc. in HTML

The following are for printing the title, etc. in a `titlepage` or a `titlingpage` in HTML:

for HTML output:

```
\begin{warpHTML}
```

Patch the pre/post title/author/date to add HTML tags, then initilize:

```
\newcommand{\prepublished}[1]{%
\def\@bsprepublished{\BlockClass{published}#1}%
}

\newcommand{\postpublished}[1]{%
\def\@bspostpublished{#1\endBlockClass}%
}

\renewcommand{\pretitle}[1]{%
\def\@bspretitle{#1\LWR@stoppars\LWR@htmltag{h1}}%
}

\renewcommand{\posttitle}[1]{%
\def\@bsposttitle{\LWR@htmltag{/h1}\LWR@startpars#1}%
}

\newcommand{\presubtitle}[1]{%
\def\@bspresubtitle{\BlockClass{subtitle}#1}%
}

```

```
\newcommand{\postsubtitle}[1]{%
\def\@bspostsubtitle{#1\endBlockClass}%
}

\renewcommand{\preauthor}[1]{%
\def\@bspreauthor{\BlockClass{author}#1}%
}

\renewcommand{\postauthor}[1]{%
\def\@bspostauthor{#1\endBlockClass}%
}

\renewcommand{\predate}[1]{%
\def\@bspredate{#1\BlockClass{titledate}}%
}

\renewcommand{\postdate}[1]{%
\def\@bspostdate{\endBlockClass#1}%
}

\prepublished{\begin{center}}
\postpublished{\par\end{center}}

\pretitle{\begin{center}}
\posttitle{\par\end{center}}

\presubtitle{\begin{center}}
\postsubtitle{\par\end{center}}

\preauthor{\begin{center}%
\begin{tabular}[t]{c}%
}
\postauthor{\end{tabular}\par\end{center}}

\predate{\begin{center}}
\postdate{\par\end{center}}
```

\printpublished

```
\newcommand*{\printpublished}{
\ifthenelse{\equal{\thepublished}{}}
{}
{
\begin{BlockClass}{published}
\@published
\end{BlockClass}
}
}
```

\printtitle

```
\newcommand*{\printtitle}
{
\LWR@stoppars
\LWR@htmltag{h1}%
\@title%
\LWR@htmltag{/h1}
\LWR@startpars
}
```

\LWR@printthetitle A private version which prints the title without footnotes, used to title each HTML page.

```
\newcommand*{\LWR@printthetitle}
{
\LWR@stoppars
\LWR@htmltag{h1}%
\thetitle%
\LWR@htmltag{/h1}
\LWR@startpars
}
```

\printsubtitle

```
\newcommand*{\printsubtitle}{
\ifthenelse{\equal{\thesubtitle}{}}
{}
{
\begin{BlockClass}{subtitle}
\@subtitle
\end{BlockClass}
}
}
```

\printauthor

```
\newcommand*{\printauthor}{
\begin{BlockClass}{author}
\begin{tabular}{c}\@author\end{tabular}
\end{BlockClass}
}
```

\printdate

```
\newcommand*{\printdate}{%
\begin{BlockClass}{titledate}
```

```
\@date
\end{BlockClass}
}
```

```
\end{warpHTML}
```

48.4 Printing the title, etc. in print form

The following are for printing the title, etc. in a `titlepage` or a `titlingpage` in print form:

for PRINT output:

```
\begin{warpprint}
```

`\printpublished`

```
\newcommand*{\printpublished}{{\Large\scshape\@published}}
```

`\printtitle`

```
\newcommand*{\printtitle}{{\Huge\@title}}
```

`\printsubtitle`

```
\newcommand*{\printsubtitle}{{\Large\itshape\@subtitle\bigskip}}
```

`\printauthor`

```
\newcommand*{\printauthor}
    {{\large\begin{tabular}[t]{c}\@author\end{tabular}}}
```

`\printdate`

```
\newcommand*{\printdate}{{\small\textit{\@date}}}
```

48.5 `\maketitle` for print output

`\maketitle` From the titling package, patched to add the publisher and subtitle.

```
\providecommand{\maketitle}{}
\if@titlepage
  \renewcommand{\maketitle}{\begin{titlepage}%
```

```
    \let\footnotesize\small
    \let\footnoterule\relax
    \let \footnote \thanks
    \@bsmarkseries
      \def\@makefnmark{\rlap{\@textsuperscript{%
         \normalfont\@bsthanksheadpre \tamark \@bsthanksheadpost}}}%
      \long\def\@makefntext##1{\makethanksmark ##1}
    \null\vfil
    \vskip 60\p@
    \vspace*{\droptitle}
    \maketitlehooka
    \ifcsempty{@published}
{}
{{\@bsprepublished \@published \@bspostpublished}\maketitlehookaa}
    {\@bspretitle \@title \@bsposttitle}
    \ifcsempty{@subtitle}
{}
{\maketitlehookaaa{\@bspresubtitle \@subtitle \@bspostsubtitle}}
    \maketitlehookb
    {\@bspreauthor \@author \@bspostauthor}
    \maketitlehookc
    {\@bspredate \@date \@bspostdate}
    \maketitlehookd
    \par
    \@thanks
    \vfil\null
    \end{titlepage}%
    \@bscontmark  %  \setcounter{footnote}{0}%
%%%    \@bsmtitlempty
  } % end titlepage defs
\else
  \renewcommand{\maketitle}{\par
    \begingroup
      \@bsmarkseries
      \def\@makefnmark{\rlap{\@textsuperscript{%
         \normalfont\@bsthanksheadpre \tamark \@bsthanksheadpost}}}%
      \long\def\@makefntext##1{\makethanksmark ##1}
      \if@twocolumn
        \ifnum \col@number=\@ne
          \@maketitle
        \else
          \twocolumn[\@maketitle]%
        \fi
      \else
        \newpage
        \global\@topnum\z@
        \@maketitle
      \fi
      \thispagestyle{plain}\@thanks
    \endgroup
```

```
    \@bscontmark  %  \setcounter{footnote}{0}%
%%%    \@bsmtitlempty
  } % end non-titlepage

  \def\@maketitle{%
    \newpage
    \null
    \vskip 2em%
          \vspace*{\droptitle}
    \maketitlehooka
    \ifcsempty{@published}
{}
{{\@bsprepublished \@published \@bspostpublished}\maketitlehookaa}
    {\@bspretitle \@title \@bsposttitle}
    \ifcsempty{@subtitle}
{}
{\maketitlehookaaa{\@bspresubtitle \@subtitle \@bspostsubtitle}}
    \maketitlehookb
    {\@bspreauthor \@author \@bspostauthor}
    \maketitlehookc
    {\@bspredate \@date \@bspostdate}
    \maketitlehookd
    \par
    \vskip 1.5em}
\fi

\providecommand{\maketitlehookaa}{}

\providecommand{\maketitlehookaaa}{}

\newcommand{\prepublished}[1]{%
\def\@bsprepublished{#1}%
}

\newcommand{\postpublished}[1]{%
\def\@bspostpublished{#1}%
}

\newcommand{\presubtitle}[1]{%
\def\@bspresubtitle{#1}%
}
```

`\presubtitle` Hook after printing the subtitle.

```
\newcommand{\postsubtitle}[1]{%
\def\@bspostsubtitle{#1}%
}
```

Initial settings:

```
\if@titlepage
\prepublished{
\vspace*{-\baselineskip}\vspace*{-\medskipamount}\vspace*{-2em}
\begin{center}}
\postpublished{\par\end{center}\vskip 2em}

\presubtitle{\unskip\begin{center}\unskip}
\postsubtitle{\par\end{center}\vskip 2em}
\else
\prepublished{\begin{center}}
\postpublished{\par\end{center}\vskip 0.5em}

\presubtitle{\begin{center}\unskip}
\postsubtitle{\par\end{center}\vskip 0.5em}
\fi
```

```
\end{warpprint}
```

48.6 \maketitle for HTML output

An HTML div of class `titlepage` is created, inside of which a LaTeX PDF minipage is generated (without HTML tags), allowing the `\thanks` footnotes to be generated immediately at the end of the title page during HTML output. This is desirable when a large table of contents immediately follows the title.

`\thanks` are a form of footnotes used in the title page. See section 41 for other kinds of footnotes.

See `\thanksmarkseries{series}`, below, to set the style of the footnote marks.

for HTML output:

```
\begin{warpHTML}
```

`\LWR@maketitlesetup` Patches `\thanks` macros to use LaTeX minipage footnotes.

```
\newcommand*{\LWR@maketitlesetup}{%
```

Select which kind of footnote marks to use:

```
\@bsmarkseries
\@mpbsmarkseries
```

Redefine the footnote mark:

```
\def\@makefnmark{\textsuperscript{\thefootnote}}
```

\thefootnote ⇒ \nameuse{arabic}{footnote}, or
\thefootnote ⇒ \nameuse{fnsymbol}{footnote}

Redefine the footnote text:

```
\long\def\@makefntext##1{%
```

Make the footnote mark and some extra horizontal space for the tags:

```
\makethanksmark \LWR@orighspace{1in}
```

\makethanksmark ⇒ \thanksfootmark ⇒ \tamark ⇒
\@thefnmark ⇒ \itshape a (or similar)

Print the text:

```
##1%
}%
}
```

\@fnsymbol {⟨*counter*⟩}

Re-defined to use an HTML entity for the double vertical bar symbol. The original definition used \| which was not being found by pdftotext.

```
\def\@fnsymbol#1{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or
   \mathsection\or \mathparagraph\or \text{\HTMLunicode{2016}}\or
   **\or \dagger\dagger \or \ddagger\ddagger \else\@ctrerr\fi}}
```

\maketitle Creates an HTML titlepage div and typesets the title, etc.

Code from the titling package is adapted, simplified, and modified for HTML output.

```
\renewcommand*{\maketitle}{%
```

An HTML titlepage div is used for all classes.

```
\begin{titlepage}
```

Set up special patches:

```
\LWR@maketitlesetup
```

Typeset the title, etc:

```
\@maketitle
```

Immediately generate any \thanks footnotes:

```
\@thanks
```

Close the HTML titlepage div:

```
\end{titlepage}
```

Reset the footnote counter:

```
\@bscontmark
}
```

\@maketitle Typesets the title, etc. for HTML:

```
\DeclareDocumentCommand{\@maketitle}{}{%
\maketitlehooka
\ifcsempty{@published}
{}
{{\@bsprepublished \@published \@bspostpublished}\maketitlehookaa}
{\@bspretitle \@title \@bsposttitle}
\ifcsempty{@subtitle}
{}
{\maketitlehookaaa{\@bspresubtitle \@subtitle \@bspostsubtitle}}
\maketitlehookb
{\@bspreauthor \@author \@bspostauthor}
\maketitlehookc
{\@bspredate \@date \@bspostdate}
\maketitlehookd
}

\providecommand{\maketitlehookaa}{}
\providecommand{\maketitlehookaaa}{}
```

\LWR@titlingmaketitle \maketitle for use inside an HTML titlingpage environment.

```
\newcommand*{\LWR@titlingmaketitle}{%
```

Typeset the title, etc:

```
\@maketitle
```

Immediately generate any \thanks footnotes:

```
\@thanks
}
```

\thanksmarkseries {⟨*series*⟩}

Sets the type of footnote marks used by \thanks, where type is 'arabic', 'roman', 'fnsymbol', etc. Modified to use the LaTeX PDF minipage which is included with the title page.

```
\renewcommand{\thanksmarkseries}[1]{%
\def\@mpbsmarkseries{%
\renewcommand*{\thempfootnote}{\@nameuse{#1}{mpfootnote}}}%
\def\@bsmarkseries{\renewcommand{\thefootnote}{\@nameuse{#1}{footnote}}}%
}
```

```
\end{warpHTML}
```

49 Abstract

The following code replaces the LaTeX default, and will itself be replaced later if the abstract package is loaded.

for HTML output:

```
\begin{warpHTML}
```

\abstractname User-redefinable title for the abstract.

Also over-written by the babel package.

```
\providecommand*{\abstractname}{Abstract}
```

Env abstract

```
\DeclareDocumentEnvironment{abstract}{}
{
\LWR@forcenewpage
\BlockClass{abstract}
\BlockClassSingle{abstracttitle}{\abstractname}
}
{
\endBlockClass
}
```

```
\end{warpHTML}
```

50 Quote and verse

50.1 Citations and attributions

\attribution for use inside quote, quotation, verse:

ex: `\attribution{author name} --- \citetitle{book name}`

for HTML output:

```
\begin{warpHTML}
\newcommand{\attribution}[1]{%
\InlineClass{attribution}{--\,#1}}% emdash
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\newcommand{\attribution}[1]{\textsc{---\,#1}}
\end{warpprint}
```

\citetitle for use inside quote, quotation, verse:

for HTML output:

```
\begin{warpHTML}
\newcommand{\citetitle}[1]{%
\InlineClass{citetitle}{--\,#1}}% emdash
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\newcommand{\citetitle}[1]{\textsl{---\,#1}}
\end{warpprint}
```

50.2 Quotes, quotations

for HTML output:

```
\begin{warpHTML}
```

Env quote

```
\renewenvironment*{quote}
{
\LWR@forcenewpage
\LWR@htmlblocktag{blockquote}
}
{\LWR@htmlblocktag{/blockquote}}

\renewenvironment*{quotation}
{
\LWR@forcenewpage
\LWR@htmlblocktag{blockquotation}
```

```
}
{\LWR@htmlblocktag{/blockquotation}}

\end{warpHTML}
```

50.3 Verse

\attrib The documentation for the verse and memoir packages suggest defining an \attrib command, which may already exist in current documents, but it will only work for print output. lwarp provides \attribution, which works for both print and HTML output. To combine the two so that \attrib is used for print and \attribution is used for HTML:

```
\begin{warpHTML}

\let\attrib\attribution

\end{warpHTML}
```

Len \leftskip
Len \leftmargini
Len \TMLvleftskip
Len \TMLleftmargini

These lengths are used by verse and memoir to control the left margin, and they may already be set by the user for print output. New lengths \HTMLvleftskip and \HTMLleftmargini are provided to control the margins in HTML output. These new lengths may be set by the user before any verse environment, and persist until they are manually changed again. One reason to change \HTMLleftmargini is if there is a wide \flagverse in use, such as the word "Chorus", in which case the value of \HTMLleftmargini should be set to a wide enough length to contain "Chorus". The default is wide enough for a stanza number.

Horizontal spacing relies on pdftotext's ability to discern the layout (-layout option) of the text in the HTML-tagged PDF output. For some settings of \HTMLleftmargini or \HTMLleftskip the horizontal alignment may not work out exactly, in which case a label may be shifted by one space.

for HTML & PRINT:

```
\begin{warpall}
```

The following lengths may be set in either print or HTML output, but are only used in HTML. This allows the user to set \vleftskip and \leftmargini for print output, and optionally select different values for HTML.

Len \TMLvleftskip Sets \vleftskip inside a verse environment in HTML.

```
\newlength{\HTMLvleftskip}
\setlength{\HTMLvleftskip}{1em}
```

Len \TMLleftmargini Sets \leftmargini inside a verse environment in HTML.

```
\newlength{\HTMLleftmargini}
\setlength{\HTMLleftmargini}{4.5em}

\end{warpall}
```

51 Verbatim

for HTML output:

```
\begin{warpHTML}
```

Env `verbatim`

```
\AfterEndPreamble{
\AtBeginEnvironment{verbatim}{%
\LWR@forcenewpage
\LWR@atbeginverbatim{verbatim}\unskip\vspace*{-\baselineskip}%
}
\AfterEndEnvironment{verbatim}{\unskip\vspace*{-\baselineskip}\LWR@afterendverbatim}
}

\end{warpHTML}
```

52 Fancyvrb

for HTML & PRINT:

```
\begin{warpall}
```

Len `\VerbatimHTMLWidth` Width to use in HTML `Verbatim` environment.

This width is used when placing line numbers to the right. Ignored during print output.

```
\newlength{\VerbatimHTMLWidth}
\setlength{\VerbatimHTMLWidth}{4in}
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

Bool `LWR@verbtags` Used to temporarily turn off verbatim tags while doing `VerbatimInput` in the HTML head.

```
\newbool{LWR@verbtags}
\booltrue{LWR@verbtags}
```

For \VerbatimFootnotes:

```
\renewcommand{\VerbatimFootnotes}{
\PackageError{lwarp}
{Verbatim footnotes are not yet supported by lwarp.}
{This may be improved some day.}
}
```

\LWR@atbeginverbatim {⟨*class*⟩}

Encloses a verbatim environment with the given CSS class.

```
\newcommand*{\LWR@atbeginverbatim}[1]
{%
```

Avoid excessive space between lines:

```
\setlength{\parskip}{0ex}%
```

Stop generating HTML paragraph tags:

```
\LWR@stoppars%
```

Create a new pre of the given class:

```
\ifbool{LWR@verbtags}{\LWR@htmltag{pre class="#1"{}}}{}%
```

Use a mono-spaced font to preserve horizontal positioning. If horizontal alignment is important for the user, use a mono-spaced font in the CSS for the verse class.

```
\LWR@origttfamily%
```

Do not produce HTML tags for \hspace inside a verse par. Restore plain LaTeX \hspace functionality:

```
\let\hspace\LWR@orighspace%
}
```

\LWR@afterendverbatim Finishes enclosing a verbatim environment.

```
\newcommand*{\LWR@afterendverbatim}{%
```

Remove excess vertical space at the end of the pre:

```
\unskip%
```

At the end of the environment, close the pre:

```
\ifbool{LWR@verbtags}{\noindent\LWR@htmltag{/pre}

}{}%
```

Resume regular paragraph handling:

```
\LWR@startpars%
}
```

`\LWR@Verbatimclass` Holds the class of the following verbatim.

```
\newcommand*{\LWR@Verbatimclass}{fancyvrb}
```

Env `VerbatimClass` {⟨*class*⟩} [⟨*Verbatim options*⟩]

Creates a `Verbatim` enclosed in a `<div>` of the given class.

```
\NewDocumentEnvironment{VerbatimClass}{m O{}}
{%
\renewcommand*{\LWR@Verbatimclass}{#1}%
\LWR@origVerbatim[#2]%
}
{\endVerbatim}
```

After the preamble is loaded, after any patches to `Verbatim`:

```
\AfterEndPreamble{
```

Remember the original defintion of `Verbatim`:

```
\let\LWR@origVerbatim\Verbatim
```

Env `Verbatim` Patched to place the environment in a `fancyvrb` div, and the label in a `fancyvrblabel` div. Also corrects the left margin for line numbers. Also uses `VerbatimHTMLWidth` to control placement of line numbers on the right. Aligning the right margin requires knowing the width.

```
\renewcommand*{\Verbatim}{%
\LWR@forcenewpage
\renewcommand*{\LWR@Verbatimclass}{fancyvrb}%
\LWR@origVerbatim%
}
```

The following patches to `Verbatim` are executed at the start and end of the environment, depending on the choice of `frame`. Original code is from the fancyvrb package.

```
\newcommand*{\LWR@fvstartnone}{%
\LWR@traceinfo{fvstartnone}%
\ifbool{LWR@verbtags}{\hbox to\z@{\LWR@htmltagc{div class="\LWR@Verbatimclass"}}}{}%
\hbox to\z@{\LWR@atbeginverbatim{verbatim}}%
}

\newcommand*{\LWR@fvendnone}{%
\LWR@traceinfo{fvendnone}%
\hbox to\z@{\LWR@afterendverbatim}%
\ifbool{LWR@verbtags}{\hbox to\z@{\LWR@htmltagc{/div}}}{}%
}

\newcommand*{\LWR@fvstartsingle}{%
\LWR@traceinfo{fvstartsingle}%
\LWR@fvstartnone%
\FV@BeginListFrame@Single%
}

\newcommand*{\LWR@fvendsingle}{%
\LWR@traceinfo{fvendsingle}%
\FV@EndListFrame@Single%
\LWR@fvendnone%
}

\newcommand*{\LWR@fvstartline}{%
\LWR@traceinfo{fvstartline}%
\LWR@fvstartnone%
\FV@BeginListFrame@Lines%
}

\newcommand*{\LWR@fvendline}{%
\LWR@traceinfo{fvendline}%
\FV@EndListFrame@Lines%
\LWR@fvendnone%
}
```

The following patches select the start/left/right/end behaviors depending on `frame`. Original code is from the fancyvrb package.

```
\def\FV@Frame@none{%
\let\FV@BeginListFrame\LWR@fvstartnone%
\let\FV@LeftListFrame\relax%
\let\FV@RightListFrame\relax%
\let\FV@EndListFrame\LWR@fvendnone}

\def\FV@Frame@single{%
\let\FV@BeginListFrame\LWR@fvstartsingle%
\let\FV@LeftListFrame\FV@LeftListFrame@Single%
\let\FV@RightListFrame\FV@RightListFrame@Single%
```

```
\let\FV@EndListFrame\LWR@fvendsingle}

\def\FV@Frame@lines{%
\let\FV@BeginListFrame\LWR@fvstartline%
\let\FV@LeftListFrame\relax%
\let\FV@RightListFrame\relax%
\let\FV@EndListFrame\LWR@fvendline}

\def\FV@Frame@topline{%
\let\FV@BeginListFrame\LWR@fvstartline%
\let\FV@LeftListFrame\relax%
\let\FV@RightListFrame\relax%
\let\FV@EndListFrame\LWR@fvendnone}

\def\FV@Frame@bottomline{%
\let\FV@BeginListFrame\LWR@fvstartnone%
\let\FV@LeftListFrame\relax%
\let\FV@RightListFrame\relax%
\let\FV@EndListFrame\LWR@fvendline}

\def\FV@Frame@leftline{%
% To define the \FV@FrameFillLine macro (from \FV@BeginListFrame)
\ifx\FancyVerbFillColor\relax%
\let\FV@FrameFillLine\relax%
\else%
\@tempdima\FV@FrameRule\relax%
\multiply\@tempdima-\tw@%
\edef\FV@FrameFillLine{%
{\noexpand\FancyVerbFillColor{\vrule\@width\number\@tempdima sp}%
\kern-\number\@tempdima sp}}%
\fi%
\let\FV@BeginListFrame\LWR@fvstartnone%
\let\FV@LeftListFrame\FV@LeftListFrame@Single%
\let\FV@RightListFrame\relax%
\let\FV@EndListFrame\LWR@fvendnone}
```

Adds the optional label to the top and bottom edges. Original code is from the fancyvrb package.

```
\def\FV@SingleFrameLine#1{%
  \hbox to\z@{%
%     \kern\leftmargin
    \ifnum#1=\z@\relax
      \let\FV@Label\FV@LabelBegin
    \else
      \let\FV@Label\FV@LabelEnd
    \fi
    \ifx\FV@Label\relax
%       \FancyVerbRuleColor{\vrule \@width\linewidth \@height\FV@FrameRule}%
```

```
    \else
      \ifnum#1=\z@
%         \setbox\z@\hbox{\strut\enspace\FV@LabelBegin\enspace\strut}%
        \ifx\FV@LabelPositionTopLine\relax
\else
        \LWR@htmltagc{div class="fancyvrblabel"}
\LWR@origtextrm{\FV@LabelBegin}% \textrm preserves emdash
\LWR@htmltagc{/div}
\fi
      \else
%         \setbox\z@\hbox{\strut\enspace\FV@LabelEnd\enspace\strut}%
        \ifx\FV@LabelPositionBottomLine\relax
\else
        \LWR@htmltagc{div class="fancyvrblabel"}
\LWR@origtextrm{\FV@LabelEnd}
\LWR@htmltagc{/div}
\fi
      \fi

    \fi
    \hss
}
}
```

Processes each line, adding optional line numbers. Original code is from the fancyvrb package.

```
\def\FV@ListProcessLine#1{%
   \hbox to \hsize{%
%      \kern\leftmargin
      \hbox to \VerbatimHTMLWidth {%
       \ifcsvoid{FV@LeftListNumber}{}{\kern 2.5em}%
\FV@LeftListNumber%
%       \FV@LeftListFrame
      \FancyVerbFormatLine{#1}%
      \hss%
%       \FV@RightListFrame
      \FV@RightListNumber%
}%
      \hss% required to avoid underfull hboxes
}
}
```

Env `BVerbatim`

```
\AtBeginEnvironment{BVerbatim}
{
\LWR@forcenewpage
\LWR@atbeginverbatim{bverbatim}
```

```
}

\AfterEndEnvironment{BVerbatim}
{
\leavevmode\par\vspace{-\baselineskip}
\LWR@afterendverbatim
}
```

Env `LVerbatim` No changes required.

End of the modifications to make at the end of the preamble:

```
} % \AfterEndPreamble
```

`\UseVerbatim` {⟨*text*⟩}

No changes required.

```
\end{warpHTML}
```

53 Theorems

`\newtheorem` {⟨*text*⟩} [⟨*counter*⟩] -or- [⟨*oldname*⟩] {⟨*text*⟩}

A few minor changes are made to supply HTML tags.

- The entire theorem is placed into a div of class `theoremcontents`.
- The label for each theorem is placed inside a span of class `theoremlabel`.
- The contents are placed inside a div of class `theoremcontents`.

for HTML output:

```
\begin{warpHTML}
```

`\@begintheorem` {⟨*name*⟩} {⟨*number*⟩}

```
\renewcommand{\@begintheorem}[2]{%
\LWR@forcenewpage
\BlockClass{theoremcontents}
\InlineClass{theoremlabel}{#1\ #2\ }
}
```

\@opargbegintheorem {⟨*name*⟩} {⟨*number*⟩} {⟨*oparg*⟩}

```
\renewcommand{\@opargbegintheorem}[3]{%
\LWR@forcenewpage
\BlockClass{theoremcontents}
\InlineClass{theoremlabel}{#1\ #2\ (#3)\ }
}
```

\@endtheorem

```
\renewcommand*{\@endtheorem}{%
\endBlockClass% theoremcontents
}
```

```
\end{warpHTML}
```

54 Lists

⚠ French If using babel with French, use

```
\frenchbsetup{StandardLists=true}
```

to preserve the special HTML and enumitem list handling.

enumitem enumitem is pre-loaded during HTML output. Many of the spacing options are rendered irrelevant by pdftotext and HTML. Numbering, labels, and \newlist function correctly.

54.1 Itemize

for HTML output:

```
\begin{warpHTML}
```

```
\let\LWR@origitem\item
```

\LWR@itemizeitem [⟨*label*⟩]

Handles \item inside an itemize or enumerate.

See \LWR@openparagraph where extra \hspace is used to leave room for the label while inside a list during paragraph construction.

```
\newcommand*{\LWR@itemizeitem}{%
\LWR@stoppars%
\LWR@startnewdepth{\LWR@depthlistitem}{\LWR@printcloselistitem{}}%
\LWR@htmltag{li}%
```

```
\LWR@startpars%
\LWR@origitem%
}
```

To have a blank item, use `\mbox{}`. This forces a new line in print output, matching the new line which will appear in HTML output. Ex:

```
begin{itemize}
item \mbox{}
   \begin{itemize}
...
```

Env `itemize` [⟨*enumitem options*⟩]

```
\AtBeginEnvironment{itemize}{\LWR@itemizestart}

\newcommand*{\LWR@itemizestart}{%
\LWR@stoppars%
\LWR@pushoneclose{\LWR@depthlist}{\LWR@printcloseitemize{}}%
\LWR@htmltag{ul style="list-style-type:none"{}}%
\LWR@startpars%
\let\item\LWR@itemizeitem%
}

\AtEndEnvironment{itemize}{\LWR@itemizeend}

\newcommand*{\LWR@itemizeend}{%
\LWR@stoppars%
\LWR@closeprevious{\LWR@depthlistitem}%
\LWR@closeoneprevious{}%
\LWR@startpars%
}
```

54.2 Enumerate

An HTML unordered list is used with customized LaTeX-generated labels.

Env `enumerate` [⟨*enumitem options*⟩]

```
\AtBeginEnvironment{enumerate}{\LWR@enumeratestart}

\newcommand*{\LWR@enumeratestart}{%
\LWR@stoppars%
```

```
\LWR@pushoneclose{\LWR@depthlist}{\LWR@printcloseitemize{}}%
\LWR@htmltag{ul style="list-style-type:none"{}}%
\LWR@startpars%
\let\item\LWR@itemizeitem%
}


\AtEndEnvironment{enumerate}{\LWR@enumerateend}

\newcommand*{\LWR@enumerateend}{%
\LWR@stoppars%
\LWR@closeprevious{\LWR@depthlistitem}%
\LWR@closeoneprevious{}%
\LWR@startpars%
}
```

54.3 Description

`\LWR@descitem` [⟨*label*⟩] Handles an `\item` inside a description.

```
\newcommand*{\LWR@descitem}[1][]%
{%
\LWR@stoppars%
\LWR@setlatestname{#1}%
\LWR@startnewdepth{\LWR@depthlistitem}{\LWR@printclosedescitem{}}%
\LWR@origitem[]%
```

Be sure the label doesn't print to the left of the rest of the file:

```
\LWR@orighspace{1in}
\LWR@htmltag{dt}#1\LWR@htmltag{/dt}%
\LWR@orignewline%
\LWR@htmltag{dd}%
\LWR@startpars%
}
```

Env `description` [⟨*enumitem options*⟩]

```
\AtBeginEnvironment{description}{\LWR@descriptionstart}

\newcommand*{\LWR@descriptionstart}{%
\LWR@stoppars%
\LWR@pushoneclose{\LWR@depthlist}{\LWR@printclosedescription{}}%
\LWR@htmltag{dl}%
\LWR@startpars%
\let\item\LWR@descitem%
```

```
}

\AtEndEnvironment{description}{\LWR@descriptionend}

\newcommand*{\LWR@descriptionend}{%
\LWR@stoppars%
\LWR@closeprevious{\LWR@depthlistitem}%
\LWR@closeoneprevious{}%
\LWR@startpars%
}
```

`\newlist` {⟨*name*⟩} {⟨*type*⟩} {⟨*maxdepth*⟩}

`\renewlist` {⟨*name*⟩} {⟨*type*⟩} {⟨*maxdepth*⟩}

For enumitem lists, new lists must have the start and end actions assigned to the new environment. Renewed lists already have their actions assigned, and thus need no changes.

```
\let\LWR@orignewlist\newlist

\renewcommand*{\newlist}[3]{%
\LWR@orignewlist{#1}{#2}{#3}%
\AtBeginEnvironment{#1}{\csuse{LWR@#2start}}%
\AtEndEnvironment{#1}{\csuse{LWR@#2end}}%
}
```

```
\end{warpHTML}
```

55 Tabular

This is arguably the most complicated part of the entire package. Numerous tricks are employed to handle the syntax which is involved.

Limitations:

column types

- Vertical rules are not yet supported.
- `*` in a column specification is not used (so far). Repeat the column type the correct number of times.
- Only one each of `@`, `!`, `>`, and `<` may be used at each column, and they are used in that order.
- `\newcolumntype` is ignored; unknown column types are set to `l`.

- tabularx ignores the width, but `X` columns do produce paragraph columns or multicolumns.

⚠ `\multirow` & `\multicolumn`

- Multirow and multicolumn cannot be used at the same time. (No rectangular holes wider than one column or taller than one row.)

⚠ `\multirow`

- For multirow, insert `\mrowcell` into any empty multi-row cells. This will be a null function for the print output, and is a placeholder for parsing the table for HTML output.

`\multirow` with rules

- If a multirow reaches to the bottom of a table, and `\bottomrule` does not go over to that edge, try adding a line of empty cells below the `\bottomrule`. This may be a browser bug.

rule at last row

- If a `\midrule` is desired after the last row, an additional row of blank cells must be used.

⚠ paragraphs

- Multiple paragraphs in one cell of a p, b, m column must have `\newline` between paragraphs.

`\cmidrule` width, trim

- `\cmidrule` does not support width or trim options due to CSS limitations.

`longtable` headings

- For longtable, place headings and footings which do not apply to HTML inside `\warpprintonly{}`.

⚠ `\warpprintonly`

- For `\toprule` and `\bottomrule`, when combined with a `warpprint` or `warpHTML` environment, if a "misplaced `\noalign`" error occurs, change

  ```
  This & That \endhead
  ```

  to

  ```
  \warpprintonly{This & That \endhead}
  ```

  and likewise with the other `\end` headings. Keep the `\endfirsthead` row unchanged, as it is still relevent to HTML output.

⚠ S columns

- For S columns (from the siunitx package), while producing print output, anything non-numeric must be placed inside { } braces, including commands such as `\multirow`. While producing HTML output, though, anything placed inside braces is not seen by lwarp's tabular handling algorithm. To resolve this problem, make a copy of the row, with one version for print output, containing the extra braces, and another version for HTML output, without the extra braces, such as:

  ```
  \warpprintonly{1 & 2 & {\multirow{2}{2cm}{Text}} & 3 \\}
  \warpHTMLonly{1 & 2 & \multirow{2}{2cm}{Text} & 3 \\}
  ```

55.1 Token lookahead

Used by `\LWR@futurenonspacelet` to look at the next token.

for HTML output: 4537 `\begin{warpHTML}`

\LWR@mynexttoken

```
\newcommand\LWR@mynexttoken\relax
```

\futurelet copies the next token then executes a function to analyze

\LWR@futurenonspacelet does the same, but ignores intervening white space

Based on the booktabs style:

\LWR@futurenonspacelet

```
\def\LWR@futurenonspacelet#1{\def\LWR@cs{#1}%
\afterassignment\LWR@fnslone\let\nexttoken= }
\def\LWR@fnslone{\expandafter\futurelet\LWR@cs\LWR@fnsltwo}
\def\LWR@fnsltwo{%
\expandafter\ifx\LWR@cs\@sptoken\let\next=\@BTfnslthree%
\else\let\next=\nexttoken\fi\next}
\def\@BTfnslthree{\afterassignment\LWR@fnslone\let\next= }
```

\LWR@getmynexttoken Looks ahead and copies the next token into \LWR@mynexttoken.

```
\newcommand*{\LWR@getmynexttoken}{%
% nothing must follow this next line
\LWR@futurenonspacelet\LWR@mynexttoken\LWR@tabledatacolumntag
}
```

55.2 Booleans

Bool LWR@startedrow True if should print a row tag before this column.

```
\newbool{LWR@startedrow}
\boolfalse{LWR@startedrow}
```

Bool LWR@doinghline True if the next row will have an hline above it.

```
\newbool{LWR@doinghline}
\boolfalse{LWR@doinghline}
```

Bool LWR@doingtbrule True if the next row will have a top/bottom rule above it.

```
\newbool{LWR@doingtbrule}
\boolfalse{LWR@doingtbrule}
```

Bool LWR@tableparcell True if are handling a paragraph inside a table cell, so must close the paragraph

tag before moving on.

```
\newbool{LWR@tableparcell}
```

Bool `LWR@skippingmrowcell` True if are doing an empty multi-row cell, and thus there is no data tag to close.

```
\newbool{LWR@skippingmrowcell}
```

Bool `LWR@intabularmetadata` True if are in a tabular but not in a data cell. Used to prevent extra HTML breaks if not inside table data.

```
\newbool{LWR@intabularmetadata}
\boolfalse{LWR@intabularmetadata}
```

55.3 Handling & and !

For technical discussion regarding problems redefining `\&`, See:
http://tex.stackexchange.com/questions/11638/
where-do-i-find-futurelets-nasty-behaviour-documented/11860#11860

`\LWR@closetabledatacell` If `LWR@skippingmrowcell` then there is no data tag to close. Otherwise, close any paragraphs, then close the data tag.

```
\newcommand*{\LWR@closetabledatacell}{%
\global\booltrue{LWR@intabularmetadata}%
\ifbool{LWR@exitingtabular}{}%
{% not exiting tabular
\ifbool{LWR@skippingmrowcell}{}%
{% not skippingmrowcell
```

Insert any `<` then any `@` and `!` column contents:

```
\unskip%
\LWR@getexparray{LWR@colafterspec}{\theLWR@tablecolspos}%
\LWR@getexparray{LWR@colatspec}{\theLWR@tablecolspos}%
\LWR@getexparray{LWR@colbangspec}{\theLWR@tablecolspos}%
```

Close paragraphs:

```
\ifbool{LWR@tableparcell}{\LWR@stoppars}{}%
\global\boolfalse{LWR@tableparcell}%
```

Close the table data cell:

```
\unskip\LWR@htmltag{/td}\LWR@orignewline%
}% not skipping mrowcell
```

```
}% not exiting tabular
\global\boolfalse{LWR@skippingmrowcell}%
}
```

`LWR@tabulardepth` tracks whether `&` is being used inside a `tabular`.

```
\newcounter{LWR@tabulardepth}
\setcounter{LWR@tabulardepth}{0}

```

When not used inside a `tabular`, `&` performs its original function as recorded here (with catcode 4).

```
\def\LWR@origampmacro{&}
```

See below for why the group is used.

```
\begingroup
```

`&` Will behave depending on whether it is being used inside `tabular`.

`&` is redefined to test whether it is inside a tabular environment, in which case it performs special processing for HTML conversion. If not, it behaves normally.

The `\catcode` allows the `&` character to be redefined.

```
\catcode`\&=\active

\gdef&{%
\ifthenelse{\cnttest{\value{LWR@tabulardepth}}{>}{0}}%
{%
```

If not skipping a multirow cell, close the current data cell.

```
\unskip%
\LWR@closetabledatacell%
```

Move to the next column.

```
\addtocounter{LWR@tablecolspos}{1}%
```

Look at the next token to decide multi or single column data tag.

```
\LWR@getmynexttoken%
}%
```

If not inside a tabular, performs the original action:

```
{\LWR@origampmacro}%
}
\endgroup
```

Outside the group, & is left its original catcode for now.

tikz package seems to require & be left alone until after tikz has been loaded.

\LWR@lwarpStart finally makes & active at the beginning of the HTML conversion.

55.4 Handling \\

Inside tabular, \\ is redefined to \LWR@tabularendofline

Throws away options \\[dim] or *

\LWR@tabularendofline

```
\NewDocumentCommand{\LWR@tabularendofline}{s o}
{%
\LWR@closetabledatacell%
```

Finish the previous row:

```
\LWR@htmltag{/tr}\LWR@orignewline
\global\booltrue{LWR@intabularmetadata}
```

Not yet started a table row:

```
\global\boolfalse{LWR@startedrow}
```

Additional setup:

```
\global\boolfalse{LWR@doinghline}%
\global\boolfalse{LWR@doingtbrule}%
\LWR@clearmidrules%
```

Start at first column:

```
\setcounter{LWR@tablecolspos}{1}
```

Look at the next token to decide between single column data tag or a special case:

```
\LWR@getmynexttoken%
}
```

55.5 Variables

```
\newcommand*{\LWR@colsresult}{}%temp storage for column format results
\newcommand*{\LWR@pposition}{}
\newcommand*{\LWR@pleft}{}
\newcommand*{\LWR@pright}{}
```

`\LWR@tablecolspec` Holds the parsed column specification, of total width `LWR@tabletotalcols`.

Will contain a string such as `llrrccpc`, exactly one letter per column, without `@`, `>`, `<`, or the vertical pipe.

```
\newcommand*{\LWR@tablecolspec}{}
```

`\LWR@strresult` Holds the result of `Str` functions.

```
\newcommand*{\LWR@strresult}{}
```

`\LWR@origcolspec` Holds the original column specs given to `tabular`.

```
\newcommand*{\LWR@origcolspec}{}
```

Ctr `LWR@tablecolswidth` Holds the width of the table specification.

(This is not the total # columns.)

```
\newcounter{LWR@tablecolswidth}
```

Ctr `LWR@tablecolspos` Where are currently looking into the table column specification.

```
\newcounter{LWR@tablecolspos}
```

Ctr `LWR@tabletotalcols` Holds the final number of table columns.

```
\newcounter{LWR@tabletotalcols}
```

Ctr `LWR@tabletotalcolsnext` Holds the next column while parsing. Is one more than `LWR@tabletotalcols`.

```
\newcounter{LWR@tabletotalcolsnext}
```

`LWR@colatspec` A data array of specifications for `@` columns. The leftmost's index is `leftedge`, the others are counter values. See section 28.

`LWR@colbangspec` A data array of specifications for `!` columns. The leftmost's index is `leftedge`, the others are counter values. See section 28.

`LWR@colbeforespec` A data array of specifications for `>` columns.

`LWR@colafterspec` A data array of specifications for `<` columns.

55.6 Parsing @, >, <, ! columns

\LWR@parseatcolumn Handles @{text} columns.

```
\newcommand*{\LWR@parseatcolumn}{%
```

Move to the next token after the '@':

```
\LWR@traceinfo{at column}%
\addtocounter{LWR@tablecolspos}{1}%
```

Read the next token into \LWR@strresult, expanding once:

```
\LWR@traceinfo{about to read the next token:}%
\expandarg%
\StrChar{\LWR@origcolspec}{\theLWR@tablecolspos}[\LWR@strresult]
\fullexpandarg%
```

Store the result into a data array, expanding once out of \LWR@strresult:

```
\LWR@traceinfo{have now read the next token}%
\ifthenelse{\cnttest{\value{LWR@tabletotalcols}}=0}
{% left edge of the table:
\LWR@traceinfo{at the left edge}%
\LWR@setexparray{LWR@colatspec}{leftedge}{\LWR@strresult}%
}%
{% not at the left edge:
\LWR@traceinfo{not at the left edge}%
\LWR@setexparray{LWR@colatspec}{\theLWR@tabletotalcols}{\LWR@strresult}%
\LWR@traceinfo{at \theLWR@tabletotalcols: %
  \LWR@getexparray{LWR@colatspec}{\theLWR@tabletotalcols})!}%
\let\LWR@strresult\relax%
\booltrue{LWR@validtablecol}%
}%
}
```

\LWR@parsebangcolumn

```
\newcommand*{\LWR@parsebangcolumn}{%
```

Move to the next token after the '!':

```
\LWR@traceinfo{bang column}%
\addtocounter{LWR@tablecolspos}{1}%
```

Read the next token into \LWR@strresult, expanding once:

```
\LWR@traceinfo{about to read the next token:}%
\expandarg%
\StrChar{\LWR@origcolspec}{\theLWR@tablecolspos}[\LWR@strresult]
\fullexpandarg%
```

Store the result into a data array, expanding once out of `\LWR@strresult`:

```
\LWR@traceinfo{have now read the next token}%
\ifthenelse{\cnttest{\value{LWR@tabletotalcols}}=0}
{% left edge of the table:
\LWR@traceinfo{at the left edge}%
\LWR@setexparray{LWR@colbangspec}{leftedge}{\LWR@strresult}%
}%
{% not at the left edge:
\LWR@traceinfo{not at the left edge}%
\LWR@setexparray{LWR@colbangspec}{\theLWR@tabletotalcols}{\LWR@strresult}%
\LWR@traceinfo{bang \theLWR@tabletotalcols: \LWR@colbangspec(\theLWR@tabletotalcols)!}%
}%
\let\LWR@strresult\relax%
\booltrue{LWR@validtablecol}%
}
```

`\LWR@parsebeforecolumn` Handles `>{text}` columns.

```
\newcommand*{\LWR@parsebeforecolumn}{%
```

Move to the next token after the '>':

```
\addtocounter{LWR@tablecolspos}{1}%
```

Read the next token, expanding once into `\LWR@strresult`:

```
\expandarg%
\StrChar{\LWR@origcolspec}{\theLWR@tablecolspos}[\LWR@strresult]%
\fullexpandarg%
```

Store the result into a data array, expanding once out of `\LWR@strresult`:

```
\LWR@setexparray{LWR@colbeforespec}{\theLWR@tabletotalcolsnext}{\LWR@strresult}%
\let\LWR@strresult\relax%
\booltrue{LWR@validtablecol}%
}
```

`\LWR@parseaftercolumn` Handles `<{text}` columns.

```
\newcommand*{\LWR@parseaftercolumn}{%
```

Move to the next token after the '<':

```
\addtocounter{LWR@tablecolspos}{1}%
```

Read the next token, expanding once into `\LWR@strresult`:

```
% \StrChar{#1}{\theLWR@tablecolspos}[\LWR@strresult]
\expandarg%
\StrChar{\LWR@origcolspec}{\theLWR@tablecolspos}[\LWR@strresult]%
\fullexpandarg%
```

Store the result into a data array, expanding once out of `\LWR@strresult`:

```
\LWR@setexparray{LWR@colafterspec}{\theLWR@tabletotalcols}{\LWR@strresult}%
\let\LWR@strresult\relax%
\booltrue{LWR@validtablecol}%
}
```

`\LWR@parseskipcolumn` Handles columns to skip, such as the vertical bar.

```
\newcommand*{\LWR@parseskipcolumn}{%
\booltrue{LWR@validtablecol}%
}
```

55.7 Parsing 'l', 'c', or 'r' columns

`\LWR@parsenormalcolumn` {⟨*thiscolumn*⟩}

Add to the accumulated column specs, advance counters, and pre-clear another column of at, before, and after specs.

```
\newcommand*{\LWR@parsenormalcolumn}[1]{%
\appto\LWR@tablecolspec{#1}%
\addtocounter{LWR@tabletotalcols}{1}%
\addtocounter{LWR@tabletotalcolsnext}{1}%
\LWR@setexparray{LWR@colatspec}{\theLWR@tabletotalcolsnext}{\relax}%
\LWR@setexparray{LWR@colbangspec}{\theLWR@tabletotalcolsnext}{\relax}%
\LWR@setexparray{LWR@colbeforespec}{\theLWR@tabletotalcolsnext}{\relax}%
\LWR@setexparray{LWR@colafterspec}{\theLWR@tabletotalcolsnext}{\relax}%
\booltrue{LWR@validtablecol}%
}
```

55.8 Parsing ‘p’, ‘m’, or ‘b’ columns

`\LWR@parsepcolumn` `{`⟨*thiscolumn*⟩`}` The width will be ignored.

```
\newcommand*{\LWR@parsepcolumn}[1]{%
```

Converts to the given column type.

```
\LWR@parsenormalcolumn{#1}%
```

skips the following width

```
\addtocounter{LWR@tablecolspos}{1}%
}
```

55.9 Parsing ‘D’ columns

From the dcolumn package.

`\LWR@parseDcolumn` `{`⟨*thiscolumn*⟩`}` The three parameters will be ignored.

```
\newcommand*{\LWR@parseDcolumn}[1]{%
```

Converts to the given column type.

```
\LWR@parsenormalcolumn{#1}%
```

Skips the following three parameters.

```
\addtocounter{LWR@tablecolspos}{3}%
}
```

55.10 Parsing the column specifications

⚠ HTML CSS cannot exactly match the LaTeX concept of a baseline for a table row. Table 7 shows the LaTeX results for various vertical-alignment choices, with the baseline of the first column drawn across all the columns for comparison. See the p column specification in table 8 for details.

Table 8 describes how each kind of column is converted to HTML.

Bool `LWR@validtablecol` True if found a valid table column type.

```
\newbool{LWR@validtablecol}
```

Table 7: Tabular baseline

l	p	m	b	r
l	par par par	mid mid mid	bot bot bot	r

Table 8: Tabular HTML column conversions

l, r, c: Converted to table cells without paragraph tags.
Uses CSS `vertical-align:middle` so that top or bottom-aligned cells may go above or below this cell.

p: Converted to table cells with paragraph tags. Ref: Table 7, LaTeX places the top line of a parbox aligned with the rest of the text line, so CSS `vertical-align:bottom` is used to have the HTML result appear with the paragraph extending below the L, R, C cells at the middle, if possible. This may be confusing as a P cell may not top-align with an L,R,C cell in the HTML conversion, especially in the presence of a B cell, and two P cells side-by-side will be aligned at the bottom instead of the top. Some adjustment of the CSS may be desired, changing `td.tdp`, `td.tdP`, `td.tdprule`, and `td.tdPrule` to `vertical-align: middle`. Another possibility is to change L,R,C, and P to `vertical-align: top` and not worry about the alignment of B and M cells or trying to approximate LaTeX baselines.

m: With paragraph tags, CSS `vertical-align:middle`.

b: With paragraph tags, CSS `vertical-align:top` so that the bottom of the text is closest to the middle of the text line.

P, M, B: Horizontally-centered versions.

S: Converted to 'r'. From the siunitx package.

D: Converted to 'c'. From the dcolumn package.

@, !, >, <: One each, in that order.

Unknown: Converted to 'l'.

`\newcolumn`: Currently treated as unknown.

`\LWR@parsetablecols` {⟨*colspecs*⟩}

Scans the column specification left to right.

Builds `\LWR@tablecolspec` with the final specification, one column per entry. The number of final columns is stored in `LWR@tabletotalcols`.

```
\newcommand*{\LWR@parsetablecols}[1]{%
\LWR@traceinfo{LWR@parsetablecols started}%
```

Remember the original supplied column spec:

```
\renewcommand*{\LWR@origcolspec}{#1}%
```

Clear the parsed resulting column spec:

```
\renewcommand*{\LWR@tablecolspec}{}%
```

Total number of columns found so far. Also pre-initialize the first several columns of specs:

```
\setcounter{LWR@tabletotalcols}{0}%
\setcounter{LWR@tabletotalcolsnext}{1}%
\LWR@setexparray{LWR@colatspec}{leftedge}{\relax}%
\LWR@setexparray{LWR@colatspec}{1}{\relax}%
\LWR@setexparray{LWR@colatspec}{2}{\relax}%
\LWR@setexparray{LWR@colatspec}{3}{\relax}%
\LWR@setexparray{LWR@colbangspec}{leftedge}{\relax}%
\LWR@setexparray{LWR@colbangspec}{1}{\relax}%
\LWR@setexparray{LWR@colbangspec}{2}{\relax}%
\LWR@setexparray{LWR@colbangspec}{3}{\relax}%
\LWR@setexparray{LWR@colbeforespec}{1}{\relax}%
\LWR@setexparray{LWR@colbeforespec}{2}{\relax}%
\LWR@setexparray{LWR@colbeforespec}{3}{\relax}%
\LWR@setexparray{LWR@colafterspec}{1}{\relax}%
\LWR@setexparray{LWR@colafterspec}{2}{\relax}%
\LWR@setexparray{LWR@colafterspec}{3}{\relax}%
```

Starting at the first column specification:

```
\setcounter{LWR@tablecolspos}{1}%
```

Place the colspecs string length into `\LWR@strresult`, and remember the number of characters in the column specification:

```
\LWR@traceinfo{about to StrLen}%
\noexpandarg%
\StrLen{#1}[\LWR@strresult]%
\fullexpandarg%
```

```
\LWR@traceinfo{finished StrLen}%
\setcounter{LWR@tablecolswidth}{\LWR@strresult}%
```

Scan through the column specifications:

```
\whiledo{\not\value{LWR@tablecolspos}>\value{LWR@tablecolswidth}}{%
```

Place the next single-character column type into `\LWR@strresult`:

```
\noexpandarg%
\StrChar{#1}{\theLWR@tablecolspos}[\LWR@strresult]%
\fullexpandarg%
```

Not yet found a valid column type

```
\boolfalse{LWR@validtablecol}%
```

Note that the parameter for a `p{spec}` column is a token list which will NOT match l,c,r,p.

```
\IfStrEq{\LWR@strresult}{l}{\LWR@parsenormalcolumn{l}}{}%
\IfStrEq{\LWR@strresult}{c}{\LWR@parsenormalcolumn{c}}{}%
\IfStrEq{\LWR@strresult}{r}{\LWR@parsenormalcolumn{r}}{}%
\IfStrEq{\LWR@strresult}{L}{\LWR@parsenormalcolumn{l}}{}%
\IfStrEq{\LWR@strresult}{C}{\LWR@parsenormalcolumn{c}}{}%
\IfStrEq{\LWR@strresult}{R}{\LWR@parsenormalcolumn{r}}{}%
\IfStrEq{\LWR@strresult}{J}{\LWR@parsenormalcolumn{l}}{}%
\IfStrEq{\LWR@strresult}{S}{\LWR@parsenormalcolumn{r}}{}%
\IfStrEq{\LWR@strresult}{\detokenize{@}}{\LWR@parseatcolumn}{}%
\IfStrEq{\LWR@strresult}{!}{\LWR@parsebangcolumn}{}%
\IfStrEq{\LWR@strresult}{>}{\LWR@parsebeforecolumn}{}%
\IfStrEq{\LWR@strresult}{<}{\LWR@parseaftercolumn}{}%
\IfStrEq{\LWR@strresult}{|}{\LWR@parseskipcolumn}{}%
\IfStrEq{\LWR@strresult}{p}{\LWR@parsepcolumn{p}}{}%
\IfStrEq{\LWR@strresult}{m}{\LWR@parsepcolumn{m}}{}%
\IfStrEq{\LWR@strresult}{b}{\LWR@parsepcolumn{b}}{}%
```

From the dcolumn package:

```
\IfStrEq{\LWR@strresult}{D}{\LWR@parseDcolumn{c}}{}%
```

From the tabularx package. X column has no parameter, but will be given paragraph tags.

```
\IfStrEq{\LWR@strresult}{X}{\LWR@parsenormalcolumn{X}}{}%
```

Many people define centered versions "P", "M", and "B":
`\newcolumntype{P}[1]{>{\centering\arraybackslash}p{#1}}`

```
\IfStrEq{\LWR@strresult}{P}{\LWR@parsepcolumn{P}}{}%
\IfStrEq{\LWR@strresult}{M}{\LWR@parsepcolumn{M}}{}%
\IfStrEq{\LWR@strresult}{B}{\LWR@parsepcolumn{B}}{}%
```

If this column was an invalid column type, convert it to a `p` column:

```
\ifbool{LWR@validtablecol}{}{%
\LWR@parsenormalcolumn{l}%
}%
\addtocounter{LWR@tablecolspos}{1}%
}%
}%
```

55.11 Starting a new row

`\LWR@maybenewtablerow` If have not yet started a new table row, begin one now. Creates a new row tag, adding a class for `hline` or `tbrule` if necessary.

```
\newcommand*{\LWR@maybenewtablerow}
{%
\ifbool{LWR@startedrow}%
{}% started the row
{% not started the row
```

Remember that now have started the row:

```
\global\booltrue{LWR@startedrow}%
```

Create the row tag, with a class if necessary.

```
\global\booltrue{LWR@intabularmetadata}%
\ifbool{LWR@doinghline}%
{\LWR@htmltag{tr class="hline"{}}\LWR@orignewline}%
{% not doing hline
\ifbool{LWR@doingtbrule}%
{\LWR@htmltag{tr class="tbrule"{}}\LWR@orignewline}%
{\LWR@htmltag{tr}\LWR@orignewline}%
}% end of not doing hline
}% end of not started the row
}
```

55.12 Data opening tag

`\LWR@tabledatasinglecolumntag` Print a table data opening tag with style for alignment

```
\newcommand*{\LWR@tabledatasinglecolumntag}%
{%
\LWR@maybenewtablerow%
```

If have found the end of tabular command, do not create the next data cell:

```
\ifbool{LWR@exitingtabular}{}%
{% not exiting tabular
```

Fetch the current column's alignment character into `\LWR@strresult`:

```
\StrChar{\LWR@tablecolspec}{\theLWR@tablecolspos}[\LWR@strresult]%
```

print the start of a new table data cell:

```
\LWR@htmltag{td class="td%
```

append this column's spec:

```
\LWR@strresult%
```

If this column has a cmidrule, add "rule" to the end of the HTML class tag:

```
\ifthenelse{\equal{\LWR@getexparray{LWR@midrules}{\theLWR@tablecolspos}}{Y}}{rule}{}%
"{}}%
```

If this is a p, m, b, or X column, allow paragraphs:

```
\ifthenelse{%
\equal{\LWR@strresult}{p}\OR%
\equal{\LWR@strresult}{m}\OR%
\equal{\LWR@strresult}{b}\OR%
\equal{\LWR@strresult}{P}\OR%
\equal{\LWR@strresult}{M}\OR%
\equal{\LWR@strresult}{B}\OR%
\equal{\LWR@strresult}{X}%
}%
{% allow pars
\LWR@startpars%
\global\booltrue{LWR@tableparcell}%
}% allow pars
{}% no pars
```

Print the `@` and `!` contents before first column, and then the `>` contents:

```
\ifthenelse{\cnttest{\value{LWR@tablecolspos}}=1}%
{%
\LWR@getexparray{LWR@colatspec}{leftedge}%
\LWR@getexparray{LWR@colbangspec}{leftedge}%
```

```
}% left edge
{}% not left edge
\LWR@getexparray{LWR@colbeforespec}{\theLWR@tablecolspos}%
\global\boolfalse{LWR@intabularmetadata}%
}% not exiting tabular
}%
```

55.13 Midrules

`LWR@midrules` `LWR@midrules` is a data array (section 28) of columns containing `Y` if a midrule should be created for each column.

Ctr `LWR@midrulecounter` Indexes across the `LWR@midrules` data array.

```
\newcounter{LWR@midrulecounter}
```

`\LWR@clearmidrules` Start new midrules. Called at beginning of tabular and also at `\\`.

Clears all `LWR@midrules` markers for this line.

```
\newcommand*{\LWR@clearmidrules}
{%
\setcounter{LWR@midrulecounter}{1}%
\whiledo{%
\cnttest{\value{LWR@midrulecounter}}{<=}{\value{LWR@tablecolswidth}}%
}%
{%
\LWR@setexparray{LWR@midrules}{\theLWR@midrulecounter}{\relax}%
\addtocounter{LWR@midrulecounter}{1}%
}%
}
```

`\LWR@subcmidrule` `[`⟨*width*⟩`] {`⟨*trim*⟩`} {`⟨*leftcolumn*⟩`} {`⟨*rightcolumn*⟩`}`

Marks `LWR@midrules` data array elements to be "Y" from left to right columns.

```
\newcommand*{\LWR@subcmidrule}[4]{%
\setcounter{LWR@midrulecounter}{#3}%
\whiledo{\cnttest{\value{LWR@midrulecounter}}{<=}{#4}}%
{%
\LWR@setexparray{LWR@midrules}{\theLWR@midrulecounter}{Y}%
\addtocounter{LWR@midrulecounter}{1}%
}% end of the whiledo
}
```

`\LWR@docmidrule` `[`⟨*width*⟩`] {`⟨*trim*⟩`} {`⟨*leftcolumn-rightcolumn*⟩`}`

Marks `LWR@midrules` array elements to be "Y" from left to right columns.

```
\NewDocumentCommand{\LWR@docmidrule}{o d() >{\SplitArgument{1}{-}}m}%
{\LWR@subcmidrule{#1}{#2}#3}
```

55.14 Multicolumns

55.14.1 Parsing multicolumns

```
\newcounter{LWR@tablemulticolswidth}
\newcounter{LWR@tablemulticolspos}
```

`\LWR@printmccoltype` `{`⟨*colspec*⟩`}` Print any valid column type found. Does not print `@`, `>`, or `<` columns or their associated tokens.

This is printed as part of the table data tag's `class`.

```
\newcommand*{\LWR@printmccoltype}[1]{%
\LWR@traceinfo{lwr@printmccoltype -#1-}%
```

Get one token of the column spec:

```
\StrChar{#1}{\theLWR@tablemulticolspos}[\LWR@strresult]%
```

Add to the HTML tag depending on which column type is found:

```
\IfStrEq{\LWR@strresult}{l}{l}{}%
\IfStrEq{\LWR@strresult}{c}{c}{}%
\IfStrEq{\LWR@strresult}{r}{r}{}%
\IfStrEq{\LWR@strresult}{p}{p}{}%
\IfStrEq{\LWR@strresult}{m}{m}{}%
\IfStrEq{\LWR@strresult}{b}{b}{}%
\IfStrEq{\LWR@strresult}{P}{P}{}%
\IfStrEq{\LWR@strresult}{M}{M}{}%
\IfStrEq{\LWR@strresult}{B}{B}{}%
\IfStrEq{\LWR@strresult}{S}{r}{}%
\IfStrEq{\LWR@strresult}{X}{p}{}%
\LWR@traceinfo{lwr@printmccoltype done}%
}
```

`\LWR@multicolpartext` Print the data with paragraph tags:

```
\newcommand*{\LWR@multicolpartext}{%
\LWR@startpars%
\LWR@multicoltext%
\LWR@stoppars%
}
```

\LWR@multicolother {⟨*colspec*⟩} For @, >, <, print the next token without paragraph tags:

```
\newcommand*{\LWR@multicolother}[1]{%
\addtocounter{LWR@tablemulticolspos}{1}%
\StrChar{#1}{\theLWR@tablemulticolspos}[\LWR@strresult]%
\LWR@strresult%
```

A valid column data type was found:

```
\booltrue{LWR@validtablecol}%
}
```

\LWR@multicolskip Nothing to print for this column type.

```
\newcommand*{\LWR@multicolskip}{%
```

A valid column data type was found:

```
\booltrue{LWR@validtablecol}%
}
```

\LWR@printmccoldata {⟨*colspec*⟩} Print the data for any valid column type found.

```
\newcommand*{\LWR@printmccoldata}[1]{%
\LWR@traceinfo{lwr@printmccoldata -#1}%
```

Not yet found a valid column type:

```
\boolfalse{LWR@validtablecol}%
```

Get one token of the column spec:

```
\StrChar{#1}{\theLWR@tablemulticolspos}[\LWR@strresult]%
```

Print the text depending on which column type is found. Also handles @, >, < as it comes to them.

```
\IfStrEq{\LWR@strresult}{l}{\LWR@multicoltext}{}%
\IfStrEq{\LWR@strresult}{c}{\LWR@multicoltext}{}%
\IfStrEq{\LWR@strresult}{r}{\LWR@multicoltext}{}%
\IfStrEq{\LWR@strresult}{D}{%
\addtocounter{LWR@tablemulticolspos}{3}% skip parameters
\LWR@multicoltext%
}{}%
\IfStrEq{\LWR@strresult}{p}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{m}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{b}{\LWR@multicolpartext}{}%
```

```
\IfStrEq{\LWR@strresult}{P}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{M}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{B}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{S}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{X}{\LWR@multicolpartext}{}%
\IfStrEq{\LWR@strresult}{|}{\LWR@multicolskip}{}%
\IfStrEq{\LWR@strresult}{\detokenize{@}}{\LWR@multicolother{#1}}{}%
\IfStrEq{\LWR@strresult}{\detokenize{!}}{\LWR@multicolother{#1}}{}%
\IfStrEq{\LWR@strresult}{\detokenize{>}}{\LWR@multicolother{#1}}{}%
\IfStrEq{\LWR@strresult}{\detokenize{<}}{\LWR@multicolother{#1}}{}%
```

If an invalid column type:

```
\ifbool{LWR@validtablecol}{}{\LWR@multicoltext}%
```

Tracing:

```
\LWR@traceinfo{lwr@printmccoldata done}%
}
```

`\parsemultricolumnalignment` {⟨*1: colspec*⟩} {⟨*2: printresults*⟩}

Scan the multicolumn specification and execute the printfunction for each entry.

Note that the spec for a `p{spec}` column, or `@`, `>`, `<`, is a token list which will NOT match `l`, `c`, `r`, or `p`.

```
\newcommand*{\LWR@parsemulticolumnalignment}[2]{%
\setcounter{LWR@tablemulticolspos}{1}%
\StrLen{#1}[\LWR@strresult]%
\setcounter{LWR@tablemulticolswidth}{\LWR@strresult}%
```

Scan across the tokens in the column spec:

```
\whiledo{%
\not\value{LWR@tablemulticolspos}>\value{LWR@tablemulticolswidth}%
}%
{%
```

Execute the assigned print function for each token in the column spec:

```
#2{#1}%
```

Move to the next token in the column spec:

```
\addtocounter{LWR@tablemulticolspos}{1}%
}%
}
```

55.14.2 High-level multicolumn interface

\LWR@domulticolumn {⟨*1: numcols*⟩} {⟨*2: colspec*⟩} {⟨*3: text*⟩}

```
\newcommand{\LWR@multicoltext}{}

\NewDocumentCommand{\LWR@domulticolumn}{m m +m}{%
\LWR@traceinfo{lwr@domulticolumn -#1- -#2-}%
```

Remember the text to be inserted, and remember that a valid column type was found:

```
\renewcommand{\LWR@multicoltext}{%
#3%
\booltrue{LWR@validtablecol}%
}%
```

Row processing:

```
\LWR@maybenewtablerow%
```

Begin the opening table data tag:

```
\LWR@htmltag{td colspan="#1"
 class="td%
```

Print the column type:

```
\LWR@parsemulticolumnalignment{#2}{\LWR@printmccoltype}%
```

If this column has a cmidrule, add "rule" to the end of the HTML class tag.

If this position had a "Y" then add "rule".

```
\ifthenelse{\equal{\LWR@getexparray{LWR@midrules}{\theLWR@tablecolspos}}{Y}}{rule}{}%
```

Close the class tag's opening quote:

```
"%
}% end of the opening table data tag
\global\boolfalse{LWR@intabularmetadata}%
\LWR@parsemulticolumnalignment{#2}{\LWR@printmccoldata}%
}
```

55.14.3 Longtable captions

Bool LWR@starredlongtable Per the caption pacakge, step the counter if longtable*.

```
\newbool{LWR@starredlongtable}
\boolfalse{LWR@starredlongtable}
```

Per the caption package. User-redefinable float type.

```
\providecommand*{\LTcaptype}{table}
```

\LWR@longtabledatacaptiontag * [⟨*toc entry*⟩] {⟨*caption*⟩}

```
\NewDocumentCommand{\LWR@longtabledatacaptiontag}{s o +m}
{%
```

Remember the latest name for `\nameref`:

```
\IfValueTF{#2}{% optional given?
\ifthenelse{\equal{#2}{}}% optional empty?
{\LWR@setlatestname{#3}}% empty
{\LWR@setlatestname{#2}}% given and non-empty
}% optional given
{\LWR@setlatestname{#3}}% no optional
```

create a multicolumn across all the columns

```
\LWR@domulticolumn{\theLWR@tabletotalcols}{P}{% \LWR@domulticolumn
% \IfBooleanTF{#1}% star?
% {\IfValueTF{#2}{\LWR@origcaption*[#2]{#3}}{\LWR@origcaption*{#3}}}
% {\IfValueTF{#2}{\LWR@origcaption[#2]{#3}}{\LWR@origcaption{#3}}}
\IfBooleanTF{#1}% star?
```

Star version, show a caption but do not make a LOT entry:

```
{% yes star
\LWR@htmlblocktag{figcaption}%
#3%
\LWR@htmlblocktag{/figcaption}%
}%
{% No star:
```

Not the star version:

Don't step the counter if `\caption[]{A caption.}`

```
\ifbool{LWR@starredlongtable}%
{%
\ifthenelse{\equal{#2}{}}% TOC entry
{}%
{%
\refstepcounter{\LTcaptype}%
```

```
\protected@edef\@currentlabel{%
\csuse{p@\LTcaptype}\csuse{the\LTcaptype}}%
}%
}{}%
```

Create an HTML caption. Afterwards, maybe make a LOT entry.

```
\LWR@htmlblocktag{figcaption}%
\csuse{fnum@\LTcaptype}\CaptionSeparator#3%
\LWR@htmlblocktag{/figcaption}%
```

See if an optional caption was given:

```
\ifthenelse{\equal{#2}{}}% TOC entry empty
```

if the optional caption was given, but empty, do not form a TOC entry

```
{}%
```

If the optional caption was given, but might only be `[]`:

```
{% TOC entry not empty
\IfNoValueTF{#2}% No TOC entry?
```

The optional caption is `[]`:

```
{% No TOC entry
\addcontentsline%
{\csuse{ext@\LTcaptype}}%
{\LTcaptype}%
{%
\protect\numberline%
{\csuse{p@\LTcaptype}\csuse{the\LTcaptype}}%
{\ignorespaces #3\protect\relax}%
}%
}% end of No TOC entry
```

The optional caption has text enclosed:

```
{% yes TOC entry
\addcontentsline%
{\csuse{ext@\LTcaptype}}%
{\LTcaptype}%
{%
\protect\numberline%
{\csuse{p@\LTcaptype}\csuse{the\LTcaptype}}%
{\ignorespaces #2\protect\relax}%
}%
}% end of yes TOC entry
```

```
}% end of TOC entry not empty
}% end of no star
}% end of \LWR@domulticolumn

\addtocounter{LWR@tablecolspos}{\theLWR@tabletotalcols}
\addtocounter{LWR@tablecolspos}{-1}

}
```

55.14.4 \tabledatamulticolumntag

\LWR@tabledatamulticolumntag {⟨*numcols*⟩} {⟨*alignment*⟩} {⟨*text*⟩}

```
\NewDocumentCommand{\LWR@tabledatamulticolumntag}{m m +m}%
{%
\LWR@domulticolumn{#1}{#2}{#3}%
\addtocounter{LWR@tablecolspos}{#1}%
\addtocounter{LWR@tablecolspos}{-1}%
}
```

55.15 Multirow

Pkg multirow

\LWR@tabledatamultirowtag {⟨*numrows*⟩} [⟨*bigstruts*⟩] {⟨*width*⟩} [⟨*fixup*⟩] {⟨*text*⟩}

```
\NewDocumentCommand{\LWR@tabledatamultirowtag}{m o m o m}%
{%
\LWR@maybenewtablerow%
```

Print the start of a new table data cell:

```
\LWR@htmltag{td rowspan="#1" class="td%
```

Append this column's spec:

```
\StrChar{\LWR@tablecolspec}{\theLWR@tablecolspos}%
```

If this column has a cmidrule, add "rule" to the end of the HTML class tag:

```
\ifthenelse{\equal{\LWR@getexparray{LWR@midrules}{\theLWR@tablecolspos}}{Y}}{rule}{}%
"{}}
```

While printing the text, redefine \\ to generate a new line

```
\begingroup \let\\\LWR@endofline #5 \endgroup
\LWR@stoppars%
\global\boolfalse{LWR@intabularmetadata}%
}%
```

55.16 Utility macros inside a table

```
\newcommand*{\LWR@donothing}{}
\newcommand*{\LWR@domidrule}{\booltrue{LWR@doinghline}}
\newcommand*{\LWR@dotbrule}{\booltrue{LWR@doingtbrule}}
```

55.17 Checking for a new table cell

`\LWR@tabledatacolumntag` Open a new HTML table cell unless the next token is for a macro which does not create data, such as `\hline`, `\toprule`, etc:

```
\newbool{LWR@exitingtabular}
\newcommand*{\LWR@tabledatacolumntag}%
{%
```

`\show\LWR@mynexttoken` to see what tokens to look for

If not any of the below, start a new table cell:

```
\let\mynext\LWR@tabledatasinglecolumntag%
```

If exiting the tabular:

```
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\end}}%
{\booltrue{LWR@exitingtabular}}{}%
```

`longtable` can have a caption in a cell

```
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\caption}}%
{\let\mynext\LWR@donothing}{}%
```

Look for other things which would not start a table cell:

```
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\multicolumn}}%
{\let\mynext\LWR@donothing}{}%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\multirow}}%
{\let\mynext\LWR@donothing}{}%
```

if come to an `\mrowcell`, this is a cell to be skipped over

```
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\mrowcell}}%
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\hline}}%
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\toprule}}%
{%
\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\midrule}}%
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\cmidrule}}%
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\cline}}%
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\bottomrule}}%
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\warpprintonly}}
{\let\mynext\LWR@donothing}{}%
%
\ifthenelse{\isequivalentto{\LWR@mynexttoken}{\warpHTMLonly}}
{\let\mynext\LWR@donothing}{}%
```

no action for an `\end` token

Add similar to the above for any other non-data tokens which might appear in the table.

Start the new table cell if was not any of the above:

```
\mynext%
}
```

```
\end{warpHTML}
```

55.18 \mrowcell

`\mrowcell` ⚠ The user must insert `\mrowcell` into any multirow cells which must be skipped. This command has no action during print output.

for HTML & PRINT:

```
\begin{warpall}
\newcommand*{\mrowcell}{}
\end{warpall}
```

55.19 New \tabular definition

for HTML output:
```
\begin{warpHTML}
```

Env `LWR@tabular` [⟨*vertposition*⟩] {⟨*colspecs*⟩}

The new tabular environment will be `\let` in `\LWR@LwarpStart`, since siunitx might redefine tabular in the user's document.

```
\newenvironment*{LWR@tabular}[2][]
{%
\LWR@traceinfo{tabular started}%
\begingroup%
\addtocounter{LWR@tabulardepth}{1}%
```

Not yet started a table row:

```
\global\boolfalse{LWR@startedrow}%
```

Not yet doing an hline:

```
\global\boolfalse{LWR@doinghline}%
```

Not yet doing a top/bottom rule:

```
\global\boolfalse{LWR@doingtbrule}%
```

Have not yet found the end of tabular command:

```
\boolfalse{LWR@exitingtabular}%
```

Create the `table` tag:

```
\global\booltrue{LWR@intabularmetadata}%
\LWR@forcenewpage
\LWR@htmlblocktag{table}%
```

Parse the table columns:

```
\LWR@parsetablecols{#2}%
```

Table col spec is: `\LWR@tablecolspec` which is a string of `llccrr`, etc.

Do not place the table inside a paragraph:

```
\LWR@stoppars%
```

Track column # for setting text-align:

```
\setcounter{LWR@tablecolspos}{1}%
```

Start looking for midrules:

```
\LWR@clearmidrules%
```

\\ becomes a macro to end the table row:

```
\let\\\LWR@tabularendofline%
```

The following may appear before a data cell is created, so after doing their actions, we look ahead with `\LWR@getmynextoken` to see if the next token might create a new data cell:

```
\renewcommand*{\hline}{\LWR@domidrule\LWR@getmynexttoken}%
\newcommand*{\midrule}{\LWR@domidrule\LWR@getmynexttoken}%
\NewDocumentCommand{\cmidrule}{o d() m}%
{\LWR@docmidrule[##1](##2){##3}\LWR@getmynexttoken}%
\RenewDocumentCommand{\cline}{m}%
{\LWR@docmidrule{##1}\LWR@getmynexttoken}%
\newcommand*{\toprule}{\LWR@dotbrule\LWR@getmynexttoken}%
\newcommand*{\bottomrule}{\LWR@dotbrule\LWR@getmynexttoken}%
```

The following create data cells and will have no more data in this cell, so we do not want to look ahead for a possible data cell, so do not want to use `\LWR@getmynexttoken`.

```
\let\multicolumn\LWR@tabledatamulticolumntag%
\let\multirow\LWR@tabledatamultirowtag%
\renewcommand*{\mrowcell}{\global\booltrue{LWR@skippingmrowcell}}%
\let\caption\LWR@longtabledatacaptiontag%
```

Reset for new processing:

```
\global\boolfalse{LWR@tableparcell}%
\global\boolfalse{LWR@skippingmrowcell}%
```

Look ahead for a possible table data cell:

```
\LWR@getmynexttoken%
}%
```

Ending the environment:

```
{%
\LWR@closetabledatacell%
```

```
\LWR@htmlblocktag{/tr}%
\LWR@htmlblocktag{/table}%
\global\boolfalse{LWR@intabularmetadata}%

\addtocounter{LWR@tabulardepth}{-1}%
\endgroup%
}

\end{warpHTML}
```

55.20 Array

Pkg `array`

array is also automatically loaded by siunitx.

56 Cross-references

Sectioning commands have been emulated from scratch, so the cross-referencing commands are custom-written for them. Emulating both avoids several layers of patches.

The zref package is used to remember section name, file, and lateximage depth and number for each label.

Table 9 shows the data structures related to cross-referencing.

for HTML output:

```
\begin{warpHTML}
```

56.1 Setup

`\@currentlabelname` To remember the most recently defined section name, description, or caption, for `\nameref`.

```
\newcommand*{\@currentlabelname}{}
```

`\LWR@stripperiod` {⟨*text*⟩} [⟨.⟩]

Removes a trailing period.

```
\def\LWR@stripperiod#1.\ltx@empty#2\@nil{#1}%
```

Table 9: Cross-referencing data structures

Original LaTeX: (print and HTML)

\refstepcounter: Steps the couunter and sets `\@currentlabel`.

\@currentlabel: `\p@<ctr>\the<ctr>` Updated by `\refstepcounter`.

\label: Writes to the `.aux` file:
`\newlabel{<label>}{{\@currentlabel}{\thepage}}`

\newlabel: When the `.aux` file is read, sets `\r@<label>`.

\r@<label>: Set to: `{{\@currentlabel}{\thepage}}`

\ref: Returns the first part of `\r@<label>`.

\pageref: Returns the second part of `\r@<label>`.

Added by lwarp: (HTML only)

\label: Adds HTML tags (section 56.3), plus `\splabel` data (section 56.2):

zLWR@name: The section name for this label.
zLWR@htmlfilenumer: The filenumber or name for this label.
zLWR@lateximagedepth: The `lateximagedepth` for this label.
zLWR@lateximagenumber: The `lateximagenumber` for this label.

\nameref: Emualted from hyperref for lwarp. See section 56.4.

\ref and \nameref: Adds HTML tags. See section 56.4.

Added by amsmath: (print and HTML)

\label: Execution is delayed until the math environment is completed.

\ltx@label: LaTeX `\label`, (HTML: patched by lwarp,) later patched by cleveref.

Added by cleverref: (print and HTML)

\refstepcounter: Added: sets `\cref@currentlabel`.

\cref@currentlabel: (`<type>=<ctr>` unless an alias is used):
`[<type>][\arabic{<ctr>}][<parent ctrs>]{\p@<ctr>\the<ctr>}` Also see section 41.4 for use with footnotes.

\label: Writes to the `.aux` file:
`\newlabel{<label>@cref}{{\cref@currentlabel}{\thepage}}`

\newlabel: (Unchanged.) When the `.aux` file is read, sets `\r@<label>@cref`.

\r@<label>@cref: Set to: `{{\cref@currentlabel}{\thepage}}`

Utility functions: See `\cref@getlabel`, `\cref@gettype`, `\cref@getcounter`, `\cref@getprefix`.

Cross-referencing names: `\crefname` and `\Crefname` assign human-readable names for references to this counter type.

Additionally patched by lwarp: (HTML only)

\cref, etc.: Modified for lwarp. See section 65.

\label inside math: See section 60.4.1.

Footnotes: See `\noteentry` in section 41.4.

`\LWR@setlatestname` {⟨*object name*⟩}

Removes `\label`, strips any final period, and remembers the result.

```
\newcommand*{\LWR@setlatestname}[1]{%
```

Remove `\label` and other commands from the name, the strip any final period. See zref-titleref and gettitlestring.

```
\GetTitleStringExpand{#1}%
\edef\@currentlabelname{\detokenize\expandafter{\GetTitleStringResult}}%
\edef\@currentlabelname{%
\expandafter\LWR@stripperiod\@currentlabelname%
\ltx@empty.\ltx@empty\@nil%
}%
}
```

56.2 Zref setup

See:
http://tex.stackexchange.com/questions/57194/
extract-section-number-from-equation-reference

Create a new property list called special:

```
\zref@newlist{special}
```

Define a new property which has the name of the most recently declared section:

```
\zref@newprop{zLWR@name}{\@currentlabelname}
```

Define a new property which has either a filename or a file number:

```
\zref@newprop{zLWR@htmlfilenumber}{%
\ifbool{FileSectionNames}{\LWR@thisfilename}{\theLWR@htmlfilenumber}%
}%
```

Additional properties for lateximages:

```
\zref@newprop{zLWR@lateximagedepth}{\arabic{LWR@lateximagedepth}}
\zref@newprop{zLWR@lateximagenumber}{\arabic{LWR@lateximagenumber}}
```

`zLWR@htmlfilenumber` property holds the file number or name

Add a `LWR@htmlfilenumber` property, and lateximage properties to special:

```
\zref@addprop{special}{zLWR@name}
\zref@addprop{special}{zLWR@htmlfilenumber}
\zref@addprop{special}{zLWR@lateximagedepth}
\zref@addprop{special}{zLWR@lateximagenumber}
```

Returns the selected field:

```
\newcommand*{\LWR@spref}[2]{%
\zref@extractdefault{#1}{#2}{??}}
```

`\LWR@nameref` {⟨*label*⟩} Returns the section name for this label:

```
\newcommand*{\LWR@nameref}[1]{%
\LWR@spref{#1}{zLWR@name}%
}
```

`\LWR@htmlfileref` {⟨*label*⟩} Returns the file number for this label:

```
\newcommand*{\LWR@htmlfileref}[1]{%
% DO NOT USE \LWR@traceinfo HERE!  Will be expanded.
\LWR@spref{#1}{zLWR@htmlfilenumber}%
}
```

`\LWR@lateximagedepthref` {⟨*label*⟩} Returns the `lateximagedepth` for this label:

```
\newcommand*{\LWR@lateximagedepthref}[1]{%
\LWR@spref{#1}{zLWR@lateximagedepth}%
}
```

`\LWR@lateximagenumberref` {⟨*label*⟩} Returns the `lateximagenumber` for this label:

```
\newcommand*{\LWR@lateximagenumberref}[1]{%
\LWR@spref{#1}{zLWR@lateximagenumber}%
}
```

`\LWR@splabel` {⟨*label*⟩} Sanitize the name and then creates the label:

```
\newcommand*{\LWR@splabel}[1]{%
\LWR@setlatestname{\@currentlabelname}%
\zref@labelbylist{#1}{special}}
```

56.3 Labels

`\LWR@subsublabel` {⟨*label*⟩} Creates an HTML id tag.

```
\newcommand*{\LWR@subsublabel}[1]{%
```

Create an HTML id tag unless are inside a lateximage, since it would appear in the image:

```
\ifthenelse{\cnttest{\value{LWR@lateximagedepth}}{>}{0}}%
{}%
{% not lateximage
```

If not doing a lateximage, create an HTML ID tag: (To be factored...)

```
\ifbool{LWR@doingstartpars}%
{% pars allowed
\ifbool{LWR@doingapar}
{% par started
\LWR@htmltag{a id="#1"{}}\LWR@htmltag{/a}%
}% par started
{% par not started
\LWR@stoppars%
\LWR@htmltag{a id="#1"{}}\LWR@htmltag{/a}%
\LWR@startpars%
}% par not started
}% pars allowed
{% pars not allowed
\LWR@htmltag{a id="#1"{}}\LWR@htmltag{/a}%
}% pars not allowed
}% not lateximage
}
```

`\LWR@newlabel` {⟨*label*⟩} [⟨*type*⟩]

`\label` during HTML output when not in math mode, removing extra spaces around the label, as done by regular LaTeX `\label`.

clevereref later encases this to add its own cross-referencing.

The optional ⟨*type*⟩ is per the ntheorem package, and is ignored.

```
\NewDocumentCommand{\LWR@newlabel}{m o}{%
\LWR@traceinfo{LWR@newlabel: starting}%
\LWR@traceinfo{LWR@newlabel: !#1!}%
% \@bsphack%
```

Create a traditional LaTeX label, as modified by cleveref:

```
\LWR@origlabel{#1}%
```

Create a special label which holds the section number, LWR@htmlfilenumber, LWR@lateximagedepth, and LWR@lateximagenumber:

```
\LWR@traceinfo{LWR@newlabel: filesectionnames is \ifbool{FileSectionNames}{true}{false}}%
\LWR@traceinfo{LWR@newlabel: LWR@thisfilename is !\LWR@thisfilename!}%
\LWR@traceinfo{LWR@newlabel: LWR@htmlfilenumber is \theLWR@htmlfilenumber}%
\LWR@splabel{#1}%
\LWR@subsublabel{#1}%
% \@esphack%
\LWR@traceinfo{LWR@newlabel: done}%
}
```

56.4 References

\LWR@startref {⟨*label*⟩} (Common code for \ref and \nameref.)

Open an HTML tag reference to a filename, # character, and a label.

```
\newcommand*{\LWR@startref}[1]
{%
\edef\LWR@lidref{\LWR@lateximagedepthref{#1}}%
\LWR@traceinfo{LWR@startref A: !#1!}%
```

Create the filename part of the link:

```
\LWR@htmltag{a href="%
\LWR@traceinfo{LWR@startref B}%
\LWR@htmlrefsectionfilename{#1}%
\LWR@traceinfo{LWR@startref C}%
\#%
```

Create the destination id:

See if LWR@lateximagedepth is unknown:

```
\LWR@traceinfo{LWR@startref D: !#1!}%
\ifthenelse{\equal{\LWR@lidref}{??}}%
```

"??" if LWR@lateximagedepth is unknown, so create a link with an unknown destination:

```
{%
\LWR@traceinfo{LWR@startref D0: ??}%
??}%
```

If `LWR@lateximagedepth` is known. Use a lateximage if the depth is greater than zero, or a regular link otherwise:

```
{%
\LWR@traceinfo{LWR@startref D1: \LWR@lidref}%
\ifthenelse{\cnttest{\LWR@lidref}{>}{0}}%
{%
\LWR@traceinfo{LWR@startref D2: \LWR@lidref}%
lateximage\LWR@lateximagenumberref{#1}%
}%
{%
\LWR@traceinfo{LWR@startref D3}%
#1%
}%
}%
\LWR@traceinfo{LWR@startref E}%
```

Closing quote:

```
"{}}%
\LWR@traceinfo{LWR@startref F}%
}
```

`\LWR@subnewref` `{`⟨*label*⟩`}` `{`⟨*label or sub@label*⟩`}`

Factored for the subfig package. Uses the original label for the hyper-reference, but prints its own text, such as "`1(b)`".

```
\NewDocumentCommand{\LWR@subnewref}{m m}{%
\LWR@traceinfo{LWR@subnewref #1 #2}%
\LWR@startref{#1}%
\LWR@origref{#2}%
\LWR@htmltag{/a}%
}
```

`\ref` * `{`⟨*label*⟩`}` `\ref` is `\let` to `\LWR@newref`

`\LWR@newref` * `{`⟨*label*⟩`}` Create an internal document reference link, or without a link if starred per hyperref.

```
\NewDocumentCommand{\LWR@newref}{s m}{%
\LWR@traceinfo{LWR@newref #2}%
\IfBooleanTF{#1}%
{\LWR@origref{#2}}%
{\LWR@subnewref{#2}{#2}}%
}
```

\pagerefPageFor Text for starred page references.

```
\newcommand*{\pagerefPageFor}{see }
```

\pageref * {⟨*label*⟩} Create an internal document reference, or just the unlinked number if starred, per hyperref.

```
\NewDocumentCommand{\LWR@newpageref}{s m}{%
\IfBooleanTF{#1}%
{(\pagerefPageFor\LWR@origref{#2})}%
{(\cpageref{#2})}%
}
```

\nameref {⟨*label*⟩}

```
\newcommand*{\nameref}[1]{%
\LWR@traceinfo{nameref A}%
\LWR@startref{#1}%
\LWR@traceinfo{nameref B}%
\LWR@nameref{#1}%
\LWR@traceinfo{nameref C}%
\LWR@htmltag{/a}%
\LWR@traceinfo{nameref D}%
}
```

\Nameref {⟨*label*⟩} In print, adds the page number. In HTML, does not.

```
\let\Nameref\nameref
```

56.5 Hyper-references

⚠ Note that the code currently only sanitizes the underscore character. Additional characters should be rendered inert as well. See the `hyperref.sty` definition of `\gdef\hyper@normalise` for an example.

Pkg `hyperref`

⚠ Do not tell other packages that hyperref is emulated. Some packages patch various commands if hyperref is present, which will probably break something, and the emulation already handles whatever may be emulated anyhow.

⚠ Any reference to `\usepackage{hyperref}` must be placed inside a `warpprint` environment.

```
% DO NOT TELL OTHER PACKAGES TO ASSUME HYPERREF:
% \EmulatesPackage{hyperref}[2015/08/01]% Disabled.  Do not do this.
```

Create a link with a text name:

\LWR@subhyperref {⟨*URL*⟩} {⟨*text*⟩}

```
\NewDocumentCommand{\LWR@subhyperref}{m +m}{%
\LWR@htmltag{a href="{#1}" target="\_{}blank"\LWR@orignewline}#2\LWR@htmltag{/a}%
\LWR@ensuredoingapar%
}
```

\LWR@subhyperrefclass {⟨*URL*⟩} {⟨*text*⟩} {⟨*htmlclass*⟩}

```
\NewDocumentCommand{\LWR@subhyperrefclass}{m +m m}{%
\LWR@htmltag{a href="{#1}"
class="#3"\LWR@orignewline}#2\LWR@htmltag{/a}%
\LWR@ensuredoingapar%
}
```

\href [⟨*options*⟩] {⟨*URL*⟩} {⟨*text*⟩}

Create a link with accompanying text:

```
\NewDocumentCommand{\LWR@hrefb}{O{} m +m}{%
\LWR@subhyperref{#2}{#3}%
\endgroup%
\LWR@ensuredoingapar%
}

\newcommand{\href}{%
\LWR@ensuredoingapar%
\begingroup%
\catcode`\_=12
\LWR@hrefb%
}
```

\nolinkurl {⟨*URL*⟩}

Print the name of the link without creating the link:

```
\newcommand*{\LWR@nolinkurlb}[1]{#1\endgroup\LWR@ensuredoingapar}

\newcommand{\nolinkurl}{%
\LWR@ensuredoingapar%
\begingroup\catcode`\_=12
\LWR@nolinkurlb%
}
```

\url {⟨*URL*⟩}

Create a link whose text name is the address of the link:

```
\newcommand*{\LWR@urlb}[1]{%
\href{#1}{#1}%
\endgroup%
\LWR@ensuredoingapar%
}

\newcommand{\url}{%
\LWR@ensuredoingapar%
\begingroup\catcode`\_=12
\LWR@urlb%
}
```

`\LWR@subinlineimage` [⟨*alttag*⟩] {⟨*class*⟩} {⟨*filename*⟩} {⟨*extension*⟩} {⟨*style*⟩}

```
\newcommand*{\LWR@subinlineimage}[5][]{%
\ifthenelse{\equal{#1}{}}%
{\LWR@htmltag{img src="#3.#4" alt="#3" style="#5" class="#2"{}}}%
{\LWR@htmltag{img src="#3.#4" alt="#1" style="#5" class="#2"{}}}%
}

\end{warpHTML}
```

Table 10: Float data structures

For each `<type>` of float (figure, table, etc.) there exists the following:

counter <type>: A counter called `<type>`, such as `figure`, `table`.

`\<type>name`: Name. `\figurename` prints "Figure", etc.

`\ext@<type>`: File extension. `\ext@figure` prints "lof", etc.

`\fps@<type>`: Placement.

`\the<type>`: Number. `\thetable` prints the number of the table, etc.

`\p@<type>`: Parent's number. Prints the number of the [within] figure, etc.

`\fnum@<type>`: Prints the figure number for the caption.
`\<type>name \the<type>`, "Figure 123".

`\<type>`: Starts the float environment. `\figure` or `\begin{figure}`

`\end<type>`: Ends the float environment. `\endfigure` or `\end{figure}`

`\tf@<ext>`: The LaTeX file identifier for the output file.

`LWR@have<type>`: A boolean rememebering whether a `\listof` was requested for a float of this type.

File with extension lo<f,t,a-z>: An output file containing the commands to build the `\listof`<type><name> "table-of-contents" structure.

Cross-referencing names: For cleveref's `\cref` and related, `\crefname` and `\Crefname` assign human-readable names for references to this float type.

57 Floats

Floats are supported, although partially through emulation.

Table 10 shows the data structure associated with each `<type>` of float.

`\@makecaption` is redefined to print the float number and caption text, separated by `\CaptionSeparator`, which works with the babel package to adjust the caption separator according to the language. French, for example, uses an en-dash instead of a colon: "Figure 123 – Caption text".

57.1 Float captions

for HTML output:
```
\begin{warpHTML}
```

`\LWR@floatbegin` {⟨*type*⟩} [⟨*placement*⟩]

Begins a `\newfloat` environment.

```
\NewDocumentCommand{\LWR@floatbegin}{m o}{%
\ifthenelse{\boolean{FormatWordProcessor}\AND\boolean{HTMLMarkFloats}}{%

=== #1 begin

}{}%
\LWR@stoppars
```

There is a new float, so increment the unique float counter:

```
\addtocounter{LWR@thisfloat}{1}%
\booltrue{LWR@freezethisfloat}%
```

```
\begingroup
```

Settings while inside the environment:

```
\LWR@origraggedright
```

Open an HTML `figure` tag:

```
\LWR@htmltag{figure id="autofloat-\arabic{LWR@thisfloat}" class="#1"}
```

```
\renewcommand*{\@captype}{#1}
\caption@settype{#1}
\LWR@startpars
}
```

`\@float` `\@dlbfloat` Support packages which create floats directly.

```
\let\@float\LWR@floatbegin
\let\@dblfloat\LWR@floatbegin
```

`\LWR@floatend` Ends a `\newfloat` environment.

```
\newcommand*{\LWR@floatend}{%
\LWR@stoppars%
\LWR@htmlelementend{figure}%
```

```
\endgroup%
\boolfalse{LWR@freezethisfloat}%
\LWR@startpars%
\ifthenelse{\boolean{FormatWordProcessor}\AND\boolean{HTMLMarkFloats}}{%

=== end

}{}%
}
```

`\end@float` `\end@dlbfloat` Support packages which create floats directly.

```
\let\end@float\LWR@floatend
\let\end@dblfloat\LWR@floatend
```

Ctr `LWR@thisfloat` A sequential counter for all floats and theorems. This is used to identify the float or theorem then reference it from the List of Figures and List of Tables.

```
\newcounter{LWR@thisfloat}
```

Bool `LWR@freezethisfloat` Prevents multiple increments of `\LWR@thisfloat` inside a float.

```
\newbool{LWR@freezethisfloat}
\boolfalse{LWR@freezethisfloat}
```

`\LWR@maybeincthisfloat`

```
\newcommand*{\LWR@maybeincthisfloat}{%
\ifbool{LWR@freezethisfloat}{}{\addtocounter{LWR@thisfloat}{1}}%
}
```

`\@captype` Remembers which float type is in use.

```
\newcommand*{\@captype}{}
```

57.1.1 Caption inside a float environment

`\CaptionSeparator` How to separate the float number and the caption text.

```
\AtBeginDocument{\providecommand*{\CaptionSeparator}{:~}}
```

`\@makecaption` {⟨*name and num*⟩} {⟨*text*⟩}

Prints the float type and number, the caption separator, and the caption text.

```
\AtBeginDocument{\renewcommand{\@makecaption}[2]{#1\CaptionSeparator#2}}
```

57.1.2 Caption and LOF linking and tracking

When a new HTML file is marked in the LaTeX PDF file, the LaTeX page number at that point is stored in `LWR@latestautopage`, (and the associated filename is remembered by the special LaTeX labels). This page number is used to generate an `autofloat` HTML `<id>` in the HTML output at the start of the new HTML file. Meanwhile, there is a float counter used to generate an HTML `autofloat` `<id>` at the start of the float itself in the HTML file. The `autopage` and `autofloat` values to use for each float are written to the `.lof`, etc. files just before each float's entry. These values are used by `\l@figure`, etc. to create the HTML links in the List of Figures, etc.

Ctr `LWR@nextautofloat` Tracks autofloat for floats. Tracks autopage for floats.

Ctr `LWR@nextautopage` These are updated per float as the `.lof` file is read.

```
\newcounter{LWR@nextautofloat}
\newcounter{LWR@nextautopage}
```

`\LWRsetnextfloat` `{`⟨*autopage*⟩`}` `{`⟨*autofloat*⟩`}`

This is written to the `.lof` file just before each float's usual entry. The autopage and autofloat are remembered for `\l@figure` to use when creating the HTML links.

```
\newcommand*{\LWRsetnextfloat}[2]{%
\setcounter{LWR@nextautopage}{#1}%
\setcounter{LWR@nextautofloat}{#2}%
}
```

Ctr `LWR@latestautopage` Updated each time a new HTML file is begun. `\LWRsetnextfloat` is written with this and the autofloat by the modified `\addcontentsline` just before each float's entry.

```
\newcounter{LWR@latestautopage}
\setcounter{LWR@latestautopage}{1}
```

```
\let\LWR@origcaption@begin\caption@begin
\let\LWR@origcaption@end\caption@end
\let\LWR@orig@@par\@@par
```

`\LWR@caption@begin` Low-level patches to create HTML tags for captions.

```
\newcommand{\LWR@caption@begin}
{
\LWR@traceinfo{LWR@caption@begin}%
```

Keep par and minipage changes local:

```
\begingroup%
```

The caption code was not allowing the closing par tag:

```
\renewcommand{\@@par}{\LWR@closeparagraph\LWR@orig@@par}%
```

No need for a minipage or \parbox inside the caption:

```
\RenewDocumentEnvironment{minipage}{O{t} o O{t} m}{}{}%
\RenewDocumentCommand{\parbox}{O{t} o O{t} m +m}{##5}%
```

Enclose the original caption code inside an HTML tag:

```
\LWR@htmlblocktag{figcaption}%
\LWR@origcaption@begin%
}
```

\LWR@caption@end Low-level patches to create HTML tags for captions.

```
\newcommand{\LWR@caption@end}
{%
\LWR@origcaption@end%
```

Subcaptions were being over-written by the closing HTML tag:

```
\vspace*{\baselineskip}%
```

Closing tag:

```
\LWR@htmlblocktag{/figcaption}%
\endgroup%
% \leavevmode% avoid bad space factor (0) error
\LWR@traceinfo{LWR@caption@end: done}%
}
```

\caption@begin \caption@end Low-level patches to create HTML tags for captions.

```
\AtBeginDocument{
\let\caption@begin\LWR@caption@begin
\let\caption@end\LWR@caption@end
}
```

\captionlistentry Tracks the float number for this caption used outside a float. Patched to create an HTML anchor.

```
\let\LWR@origcaptionlistentry\captionlistentry

\renewcommand*{\captionlistentry}{%
\LWR@maybeincthisfloat%
\LWR@ensuredoingapar%
\LWR@htmltag{a id="autofloat-\arabic{LWR@thisfloat}"{}}\LWR@htmltag{/a}%
\LWR@origcaptionlistentry%
}

\def\LWR@LTcaptionlistentry{%
\LWR@ensuredoingapar%
\LWR@htmltag{a id="autofloat-\arabic{LWR@thisfloat}"{}}\LWR@htmltag{/a}%
  \bgroup
  \@ifstar{\egroup\LWR@LT@captionlistentry}% gobble *
          {\egroup\LWR@LT@captionlistentry}}%
\def\LWR@LT@captionlistentry#1{%
  \caption@listentry\@firstoftwo[\LTcaptype]{#1}}%
```

`\addcontentsline` Patched to write the autopage and autofloat before each float's entry. No changes if writing `.toc` For a theorem, automatically defines `\ext@<type>` as needed, to mimic and reuse the float mechanism.

```
\let\LWR@origaddcontentsline\addcontentsline

\renewcommand*{\addcontentsline}[3]{%
\ifthenelse{\equal{#1}{toc}}{}{%
\ifthenelse{\equal{#1}{thm}}{\csdef{ext@#2}{thm}}{}
\addtocontents{\@nameuse{ext@#2}}{%
\protect\LWRsetnextfloat%
{\arabic{LWR@latestautopage}}%
{\arabic{LWR@thisfloat}}%
}% addtocontents
}% not toc
\LWR@origaddcontentsline{#1}{#2}{#3}%
}
```

`\captionof` Patched to track the float number since this is used outside a float, and also create an HTML anchor for the virtual float.

```
\AtBeginDocument{
\let\LWR@origcaptionof\captionof

\renewcommand*{\captionof}{%
\LWR@maybeincthisfloat%
\LWR@stoppars
\LWR@htmltag{a id="autofloat-\arabic{LWR@thisfloat}"{}}\LWR@htmltag{/a}%
\LWR@origcaptionof%
}
```

```
}
\end{warpHTML}
```

58 Table of Contents, LOF, LOT

This section controls the generation of the TOC, LOF, LOT.

The `.toc`, `.lof`, and `.lot` files are named by the source code `\jobname`.

In HTML, the printed tables are placed inside a div of class `.toc`, `.lof`, or `.lot`.

A "sidetoc" is provided which prints a subset of the TOC on the side of each page other than the homepage.

The regular LaTeX infrastructure is used for TOC, along with some patches to generate HTML output.

for HTML output:

```
\begin{warpHTML}
```

58.1 Reading and printing the TOC

`\LWR@myshorttoc` {⟨*toc/lof/lot*⟩}

Reads in and prints the TOC/LOF/LOT at the current position. While doing so, makes the `@` character into a normal letter to allow formatting commands in the section names.

Unlike in regular LaTeX, the file is not reset after being read, since the TOC may be referred to again in each HTML page, and is used for the sideTOC.

```
\newcommand*{\LWR@myshorttoc}[1]{
\LWR@ensuredoingapar
```

Only if the file exists:

```
\IfFileExists{\jobname.#1}{
```

⚠ Make `@` a regular letter. Many of the commands in the file will have `@` characters in them, so `@` must be made a regular letter.

⚠ disabled For `pdflatex`, also change to `latin1` encoding. When reading back a file with accented characters, the encoding change seems to be required, rather than leaving it `utf8`.

```
\begingroup
% \ifxetexorluatex%
% \else
% \inputencoding{latin1}% currently disabled
% \fi
\makeatletter
```

Read in the TOC file:

```
\@input{\jobname.#1}
% \makeatother
\endgroup
}%
{}%
}
```

`\LWR@subtableofcontents` {⟨*toc/lof/lot*⟩} {⟨*sectionstarname*⟩}

Places a TOC/LOF/LOT at the current position.

```
\NewDocumentCommand{\LWR@subtableofcontents}{m m}{%
```

Closes previous levels:

```
\@ifundefined{chapter}
{\LWR@closeprevious{\LWR@depthsection}}
{\LWR@closeprevious{\LWR@depthchapter}}
```

Prints any pending footnotes so that they appear above the potentially large TOC:

```
\LWR@printpendingfootnotes
```

Place the list into its own chapter (if defined) or section:

```
\@ifundefined{chapter}{\section*{#2}}{\chapter*{#2}}
```

Create a new HTML `nav` containing the TOC/LOF/LOT:

```
\LWR@htmlelementclass{nav}{#1}
```

Create the actual list:

```
\LWR@myshorttoc{#1}
```

Close the `nav`:

```
\LWR@htmlelementclassend{nav}{#1}
}
```

Patch `\@starttoc` to encapsulate the TOC inside HTML tags:

```
\let\LWR@orig@starttoc\@starttoc

\renewcommand{\@starttoc}[1]{
\LWR@htmlelementclass{nav}{#1}
\LWR@orig@starttoc{#1}
\LWR@htmlelementclassend{nav}{#1}
}
```

Patch `\tableofcontents`, etc. to print footnotes first. newfloat uses `\listoffigures` for all future float types.

```
\let\LWR@origtableofcontents\tableofcontents
\let\LWR@origlistoffigures\listoffigures
\let\LWR@origlistoftables\listoftables

\renewcommand*{\tableofcontents}{%
```

Do not print the table of contents if formatting for a word processor, which will presumably auto-generate its own updated table of contents:

```
\ifbool{FormatWordProcessor}{}{
```

Copy the `.toc` file to `.sidetoc` for printing the sideTOC. The original `.toc` file is renewed when `\tableofcontents` is finished.

```
\LWR@copyfile{\jobname.toc}{\jobname.sidetoc}%
\LWR@printpendingfootnotes
\LWR@origtableofcontents
}
}
\renewcommand*{\listoffigures}{
\ifbool{FormatWordProcessor}{}{
\LWR@printpendingfootnotes
\LWR@origlistoffigures
}
}

\renewcommand*{\listoftables}{
\ifbool{FormatWordProcessor}{}{
\LWR@printpendingfootnotes
\LWR@origlistoftables
}
}
```

58.2 High-level TOC commands

`\listof` {⟨*type*⟩} {⟨*title*⟩}

Emulate the `\listof` command from the float package (section 95). Used to create lists of custom float types. Also used to redefine the standard LaTeX `\listoffigures` and `\listoftables` commands.

```
\NewDocumentCommand{\listof}{m +m}{%
\LWR@subtableofcontents{\@nameuse{ext@#1}}{#2}
\expandafter\newwrite\csname tf@\csname ext@#1\endcsname\endcsname
\immediate\openout \csname tf@\csname ext@#1\endcsname\endcsname
    \jobname.\csuse{ext@#1}\relax
}
```

58.3 Side TOC

The "side TOC" is a table-of-contents positioned to the side.

It may be renamed by redefining `\sidetocname`, and may contain paragraphs.

CSS may be used to format the sideTOC:

CSS related to sideTOC:

`nav.sidetoc`: The entire sideTOC.

`div.sidetoctitle`: The title.

`div.sidetoccontents`: The table of contents.

```
\end{warpHTML}
```

for HTML & PRINT:

```
\begin{warpall}
```

Ctr `SideTOCDepth` Controls how deep the side-TOC gets. Use a standard LaTeX section level similar to `tocdepth`.

```
\newcounter{SideTOCDepth}
\setcounter{SideTOCDepth}{1}
```

`\sidetocname` Holds the default name for the sideTOC.

```
\newcommand{\sidetocname}{Contents}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

`\LWR@sidetoc` Creates the actual side-TOC.

```
\newcommand*{\LWR@sidetoc}{
\LWR@forcenewpage
\LWR@stoppars

```

The entire sideTOC is placed into a `nav` of class `sidetoc`.

```
\LWR@htmlelementclass{nav}{sidetoc}

\setcounter{tocdepth}{\value{SideTOCDepth}}

```

The title is placed into a div of class `sidetoctitle`, and may contain paragraphs.

```
\begin{BlockClass}{sidetoctitle}
\sidetocname
\end{BlockClass}
```

The table of contents is placed into a div of class `sidetoccontents`.

```
\begin{BlockClass}{sidetoccontents}
\LinkHome

\LWR@myshorttoc{sidetoc}
\end{BlockClass}
\LWR@htmlelementclassend{nav}{sidetoc}
}
```

58.4 Low-level TOC line formatting

`\numberline` `{`⟨*number*⟩`}`

(Called from each line in the `.aux`, `.lof` files.)

Record this section number for further use:

```
\renewcommand*{\numberline}[1]{%
\LWR@sectionnumber{#1}%
}
```

`\hypertoc` {⟨*1: depth*⟩} {⟨*2: type*⟩} {⟨*3: name*⟩} {⟨*4: page*⟩}

Called by `\l@section`, etc. to create a hyperlink to a section.

The autopage label is always created just after the section.

#1 is depth

#2 is `section`, `subsection`, etc.

#3 the text of the caption

#4 page number

```
\NewDocumentCommand{\hypertoc}{m m +m m}{%
```

Respond to tocdepth:

```
\ifthenelse{\cnttest{#1}{<=}{\value{tocdepth}}}{%
\LWR@startpars%
```

Create an HTML link to filename#autosec-(page), with text of the caption, of the given HTML class.

```
\LWR@subhyperrefclass{%
\LWR@htmlrefsectionfilename{autopage-#4}\#autosec-#4}{#3}{toc#2}%
\LWR@stoppars%
}
{}
}
```

Ctr `lofdepth` TOC depth for figures.

```
\newcounter{lofdepth}
\setcounter{lofdepth}{1}
```

Ctr `lotdepth` TOC depth for tables.

```
\newcounter{lotdepth}
\setcounter{lotdepth}{1}
```

`\hypertocfloat` {⟨*1: depth*⟩} {⟨*2: type*⟩} {⟨*3: ext of parent*⟩} {⟨*4: caption*⟩} {⟨*5: page*⟩}

#1 is depth

#2 is `figure`, `table`, etc.

#3 is `lof`, `lot`, of the parent.

#4 the text of the caption

#5 page number

```
\newcommand{\hypertocfloat}[5]{%
\LWR@startpars
```

If some float-creation package has not yet defined the float type's `lofdepth` counter, etc, define it here:

```
\@ifundefined{c@#3depth}{%
\newcounter{#3depth}%
\setcounter{#3depth}{1}%
}{}%
```

Respond to `lofdepth`, etc.:

```
\LWR@traceinfo{hypertocfloat depth is #1 #3depth is \arabic{#3depth}}%
\ifthenelse{\cnttest{#1}{<=}{\arabic{#3depth}}}{%
\LWR@startpars%
```

Create an HTML link to filename#autofloat-(float number), with text of the caption, of the given HTML class.

```
\LWR@subhyperrefclass{%
\LWR@htmlrefsectionfilename{autopage-\arabic{LWR@nextautopage}}%
\#autofloat-\arabic{LWR@nextautofloat}}%
{#4}{toc#2}%
\LWR@stoppars%
}{}%
}
```

Automatically called by `\contentsline`:

```
\renewcommand{\l@part}[2]{\hypertoc{-1}{part}{#1}{#2}}
\DeclareDocumentCommand{\l@chapter}{m m}
{\hypertoc{0}{chapter}{#1}{#2}}
\renewcommand{\l@section}[2]{\hypertoc{1}{section}{#1}{#2}}
\renewcommand{\l@subsection}[2]{\hypertoc{2}{subsection}{#1}{#2}}
\renewcommand{\l@subsubsection}[2]
{\hypertoc{3}{subsubsection}{#1}{#2}}
\renewcommand{\l@paragraph}[2]{\hypertoc{4}{paragraph}{#1}{#2}}
\renewcommand{\l@subparagraph}[2]{\hypertoc{5}{subparagraph}{#1}{#2}}
\renewcommand{\l@figure}[2]{\hypertocfloat{1}{figure}{lof}{#1}{#2}}
\renewcommand{\l@table}[2]{\hypertocfloat{1}{table}{lot}{#1}{#2}}
```

```
\end{warpHTML}
```

59 Index and glossary

See:
http://tex.stackexchange.com/questions/187038/how-to-mention-section-number-in-index-created-by-imakeidx

Index links are tracked by the counter `LWR@autoindex`. This counter is used to create a label for each index entry, and a reference to this label for each entry in the index listing. This method allows each index entry to link directly to its exact position in the document.

for HTML output:

```
\begin{warpHTML}
```

```
\newcounter{LWR@autoindex}
\setcounter{LWR@autoindex}{0}

\newcounter{LWR@autoglossary}
\setcounter{LWR@autoglossary}{0}
```

`\LWR@indexsection` Controls whether the index will be in a section or a chapter, depending on the documentclass.

```
\@ifundefined{chapter}
{\newcommand*{\LWR@indexsection}{\section{\indexname}}}
{\newcommand*{\LWR@indexsection}{\chapter{\indexname}}}
```

`\printindex`

```
\let\LWR@origprintindex\printindex

\renewcommand*{\printindex}
{
\LWR@indexsection
\LWR@startpars
\LWR@origprintindex
}
```

Env `theindex`

```
\renewenvironment*{theindex}{%
\let\item\LWR@indexitem%
\let\subitem\LWR@indexsubitem%
\let\subsubitem\LWR@indexsubsubitem%
}{}
```

`\LWR@indexitem`

```
\newcommand{\LWR@indexitem}{

\InlineClass{indexitem}{}
}
```

\LWR@indexitem

```
\newcommand{\LWR@indexsubitem}{

\InlineClass{indexsubitem}{}
}
```

\LWR@indexitem

```
\newcommand{\LWR@indexsubsubitem}{

\InlineClass{indexsubsubitem}{}
}
```

\@wrindex {⟨*term*⟩} Redefined to write the LWR@latestautopage counter instead of page

```
\def\@wrindex#1{%
\addtocounter{LWR@autoindex}{1}%
\LWR@newlabel{LWRindex-\theLWR@autoindex}%
\protected@write\@indexfile{}%
{\string\indexentry{#1}{\theLWR@autoindex}}%
\endgroup
\@esphack}
```

\@wrglossary {⟨*term*⟩} Redefined to write the LWR@latestautopage counter instead of page

```
\def\@wrglossary#1{%
\addtocounter{LWR@autoglossary}{1}%
\LWR@newlabel{LWRglossary-\theLWR@autoglossary}%
\protected@write\@glossaryfile{}%
{\string\glossaryentry{#1}{\theLWR@autoglossary}}%
\endgroup
\@esphack}
```

\hyperindexref {⟨*autosecnumber*⟩}

\hyperindexref{web address} is inserted into *.ind by the xindy style file lwarp.xdy

```
\newcommand*{\hyperindexref}[1]{\nameref{LWRindex-#1}}
```

```
\end{warpHTML}
```

for PRINT output: A null command for print mode, in case hyperref was not used:

```
\begin{warpprint}
\newcommand{\hyperindexref}[1]{#1}
\end{warpprint}
```

60 Math

Math may be rendered as SVG graphics or using the MathJax JavaScript display engine.

SVG math option

For SVG math, math is rendered as usual by LaTeX into the initial PDF file using the current font[9], then is captured from the PDF and converted to SVG graphics via a number of utility programs. The SVG format is a scalable-vector web format, so math may be typeset by LaTeX with its fine control and precision, then displayed or printed at any size, depending on (sometimes broken) browser support. An HTML `ALT` tag carries the LaTeX code which generated the math, allowing copy/paste of the LaTeX math expression into other documents.

SVG image font size

The size of the math and text used in the SVG image may be adjusted by setting `\LateximageFontSizeName` to a font size name — *without the backslash*, for ex:

```
\renewcommand{\LateximageFontSizeName}{large}
```

SVG files

As currently implemented, each instance of math creates a new SVG file. In text with many references to math variables, this can result in a large number of files with duplicate content. In the future, some method of content-based naming and checksumming may be used to remove the need for duplicate files.

SVG inline

Another approach would be to in-line the SVG files directly into the HTML. This avoids having a large number of files and potentially speeds loading the images, but dis-allows the possibility of sharing one file among many instances without user intervention.

PNG files

Others have used PNG files, sometimes pre-scaled for print resolution but displayed on-screen at a scaled down size. This allows high-quality print output at the expense of larger files, but SVG files are also larger as well.

MathML

Conversion to MathML might be a better approach, among other things allowing a more compact representation of math than SVG drawings. Problems with MathML include limited browser support and some issues with the fine control of the appearance of the result. Also see section 7 regarding EPUB output with MathJax.

MathJax math option

The popular MathJax alternative (`mathjax.org`) may be used to display math.

Prog `MathJax`

[9] See section 165 regarding fonts and fractions.

When MathJax is enabled, math is rendered twice:

1. As regular LaTeX PDF output placed inside an HTML comment, allowing equation numbering and cross referencing to be almost entirely under the control of LaTeX, and
2. As detokenized printed LaTeX commands placed directly into the HTML output for interpretation by the MathJax display scripts. An additional script is used to pre-set the equation number format and value according to the current LaTeX values, and the MathJax cross-referencing system is ignored in favor of the LaTeX internal system, seamlessly integrating with the rest of the LaTeX code.

MathJax limitations
Prog `MathJax`

Limitations when using MathJax include:

chapter numbers

- In document classes which have chapters, `\tag`ged equations have the chapter number prepended in HTML output, unlike LaTeX. `\tag*` equations (correctly) do not. This may be improved with future versions of the MathJax support script.

  https://groups.google.com/forum/#!topic/mathjax-users/jUtewUcE2bY

subequations

- MathJax itself does not support subequations. This may be improved by parsing the LaTeX math expression to manually insert tags, but this has not yet been done.

footnotes in math

- Footnotes inside equations are not yet supported while using MathJax.

lateximage

- Math appearing inside a lateximage, and therefore also inside a Ti*k*z or `picture` environment, is rendered as SVG math even if MathJax is used in the rest of the document.

siunitx

⚠ siunitx inside an equation

- Usage of siunitx inside a math equation is supported via a third-party MathJax extension. While inside a math expression, do not use `\SI` or `\si` inside `\text`, where it will be rendered as normal text.

  https://github.com/mathjax/MathJax-third-party-extensions/tree/master/siunitx

LaTeX macros

- MathJax does not automatically support custom LaTeX macros, but they may be set up by the user.

custom MathJax macros

As an example of using custom LaTeX macros with MathJax, place the following at the start of the document, after `\begin{document}`:

```
\begin{warpHTML}  % Only for HTML output,
\ifbool{mathjax}  % and only for MathJax output:
{
\(  % New macros for MathJax are placed inside a math expression:
\newcommand{\expval}[1]{\langle#1\rangle}
\newcommand{\abs}[1]{\lvert#1\rvert}
\)
}{}
\end{warpHTML}
```

for HTML output:

```
\begin{warpHTML}
```

\$ Plain dollar signs appearing in the HTML output may be interpreted by MathJax to be math shifts. For a plain text dollar \$, print it inside a span to avoid it being interpreted by MathJax, unless are inside a lateximage, in which case it will not be seen by MathJax.

```
\let\LWR@origtextdollar\$

\renewcommand*{\$}{%
\ifthenelse{\cnttest{\value{LWR@lateximagedepth}}{>}{0}}%
{\LWR@origtextdollar}%
{\LWR@htmltagc{span}\LWR@origtextdollar\LWR@htmltagc{/span}}%
}
```

Ctr `LWR@externalfilecnt` Counter for the external files which are generated and then referenced from the HTML:

```
\newcounter{LWR@externalfilecnt}
```

60.1 Inline and display math

```
\let\LWR@origdollar=$
\let\LWR@secondorigdollar=$% balance for editor syntax highlighting
```

```
\let\LWR@origopenparen\(
\let\LWR@origcloseparen\)
\let\LWR@origopenbracket\[
\let\LWR@origclosebracket\]
```

\$ \$\$ Redefine the dollar sign to place math inside a `lateximage`, or use MathJax:

```
\begingroup
\catcode`\$=\active%
\protected\gdef${\@ifnextchar$\LWR@doubledollar\LWR@singledollar}%
```

`\LWR@doubledollar` Redefine the double dollar sign to place math inside a `lateximage`, or use MathJax:

```
\gdef\LWR@doubledollar$#1$${
\ifbool{mathjax}%
```

For MathJax, print the math between `\[` and `\]`:

```
{\textbackslash[\LWR@HTMLsanitize{#1}\textbackslash]}
```

For SVG, print the math inside a `lateximage`, with an ALT tag of the LaTeX code:

```
{% not mathjax

\begin{lateximage}%
[\textbackslash{[} \LWR@HTMLsanitize{#1} \textbackslash{]}]%
\LWR@origdollar\LWR@origdollar#1\LWR@origdollar\LWR@origdollar%
\end{lateximage}

}
}%
```

`\LWR@singledollar` Redefine the single dollar sign to place math inside a `lateximage`, or use MathJax:

```
\gdef\LWR@singledollar#1${%
\ifbool{mathjax}%
```

For MathJax, print the math between `\(` and `\)`:

```
{\textbackslash(\LWR@HTMLsanitize{#1}\textbackslash)}
```

For SVG, print the math inside a `lateximage`, with an ALT tag of the LaTeX code:

```
{% not mathjax
\begin{lateximage}%
[\textbackslash( \LWR@HTMLsanitize{#1} \textbackslash)]%
\LWR@origdollar#1\LWR@origdollar%
\end{lateximage}%
}%
}%
```

`\(` `\)` Redefine to the above dollar macros.

```
\gdef\(#1\){$#1$}
\gdef\[#1\]{$$#1$$}

\endgroup
```

Remove the old `math` and `displaymath` environments:

```
\let\math\relax
\let\endmath\relax
\let\displaymath\relax
\let\enddisplaymath\relax
```

Env `math` Set math mode then typeset the body of what was between the begin/end. See the environ package for `\BODY`.

```
\NewEnviron{math}{\expandafter\(\BODY\)}
```

Env `displaymath` Set math mode then typeset the body of what was between the begin/end. See the environ package for `\BODY`.

```
\NewEnviron{displaymath}{\expandafter\[\BODY\]\@ignoretrue}
```

When the document begins, the dollar sign must be made active to trigger the new math macros:

```
\AtBeginDocument{\catcode`\$=\active}
```

60.2 MathJax support

Ctr `LWR@nextequation` Used to add one to compute the next equation number.

```
\newcounter{LWR@nextequation}
```

`\LWR@syncmathjax` Sets the MathJax equation format and number for the following equations.

These MathJax commands are printed inside "`\(`" and "`\)`" characters. They are printed to HTML output, not interpreted by LaTeX.

```
\newcommand*{\LWR@syncmathjax}{%
```

If using chapters, place the chapter number in front of the equation. Otherwise, use the simple equation number.

```
\ifcsdef{thechapter}{
\BlockClassSingle{hidden}{
\textbackslash(
\textbackslash{}seteqsection \{\thechapter\}
\textbackslash)
}
```

```
}
{}% not using chapters

```

MathJax doesn't allow setting the equation number to 1:

```
\ifthenelse{\cnttest{\value{equation}}>0}
{
```

Tell MathJax that the next set of equations begins with the current LaTeX equation number, plus one.

```
\setcounter{LWR@nextequation}{\value{equation}}
\addtocounter{LWR@nextequation}{1}
```

Place the MathJax command inside "`\(`" and "`\)`" characters, to be printed to HTML, not interpreted by LaTeX.

```
\BlockClassSingle{hidden}{
\textbackslash(
\textbackslash{}seteqnumber \{\arabic{LWR@nextequation}\}
\textbackslash)
}
}{}% not eq > 1
}
```

`\LWR@restoremathlatexformatting` While producing math, use regular LaTeX formatting instead of HTML tags.

```
\newcommand*{\LWR@restoremathlatexformatting}{%
\let\hspace\LWR@orighspace%
\let\rule\LWR@origrule%
\let\,\LWR@origcomma% disable HTML short unbreakable space
\let\textit\LWR@origtextit%
\let\textbf\LWR@origtextbf%
\let\texttt\LWR@origtexttt%
\let\textsc\LWR@origtextsc%
\let\textsf\LWR@origtextsf%
\let\textrm\LWR@origtextrm%
\renewcommand*{\thefootnote}{\fnsymbol{footnote}}%
\let\textsuperscript\LWR@origtextsuperscript%
\let\textsubscript\LWR@origtextsubscript%
\let~\LWR@origtilde%
\let\enskip\LWR@origenskip%
\let\quad\LWR@origquad%
\let\qquad\LWR@origqquad%
}
```

`\LWR@hidelatexequation` `{`⟨*environment*⟩`}` `{`⟨*contents*⟩`}`

Creates the LaTeX version of the equation inside an HTML comment.

```
\NewDocumentCommand{\LWR@hidelatexequation}{m +m}{%
```

Stop HTML paragraph handling and open an HTML comment:

```
\LWR@stoppars
\LWR@htmlopencomment

```

Start the LaTeX math environment inside the HTML comment:

```
\begingroup
\csuse{LWR@orig#1}
```

While in the math environment, restore various commands to their LaTeX meanings.

```
\LWR@restoremathlatexformatting
```

See `\LWR@htmlmathlabel` in section 60.4.1.

Print the contents of the equation:

```
#2
```

End the LaTeX math environment inside the HTML comment:

```
\csuse{LWR@origend#1}
\endgroup

```

Close the HTML comment and resume HTML paragraph handling:

```
\LWR@htmlclosecomment
\LWR@startpars
}
```

`\LWR@addmathjax` `{`⟨*environment*⟩`}` `{`⟨*contents*⟩`}`

Given the name of a math environment and its contents, create a MathJax instance. The contents are printed to HTML output, not interpreted by LaTeX.

```
\NewDocumentCommand{\LWR@addmathjax}{m +m}{%

```

Enclose the MathJax environment inside printed "`\(`" and "`\)`" characters.

```
\textbackslash{}begin\{#1\}
```

Print the contents, sanitizing for HTML special characters.

```
\LWR@HTMLsanitizeexpand{\detokenize\expandafter{#2}}
```

Close the MathJax environment:

```
\textbackslash{}end\{#1\}

}
```

60.3 Equation environment

Remember existing `equation` environment:

```
\let\LWR@origequation\equation
\let\LWR@origendequation\endequation
```

Remove existing `equation` environment:

```
\let\equation\relax
\let\endequation\relax
```

Env `equation` The new `equation` environment is created with `\NewEnviron` (from the environ package), which stores the contents of its environment in a macro called `\BODY`.

For SVG math output, the contents are typeset using the original `equation` inside a `lateximage`, along with an ALT tag containing a detokenized copy of the LaTeX source for the math.

For MathJax output, the contents are typeset in an original `equation` environment placed inside a HTML comment, with special processing for `\label`s. The contents are also printed to the HTML output for processing by the MathJax script.

```
\NewEnviron{equation}{%

\ifbool{mathjax}
```

MathJax output:

```
{
```

Print commands to syncronize MathJax's equation number and format to the current LaTeX chapter/section and equation number:

```
\LWR@syncmathjax
```

Print the LaTeX math inside an HTML comment:

```
\LWR@hidelatexequation{equation}{\BODY}
}
```

SVG output: Create the `lateximage` along with an HTML ALT tag having an equation number, the LaTeX equation environment commmands, and the contents of the environment's `\BODY`.

```
{% not mathjax
```

Begin the `lateximage` with an ALT tag containing the math source:

```
\begin{lateximage}[(\theequation) \textbackslash{begin\{equation\}}%
 \LWR@HTMLsanitizeexpand{\detokenize\expandafter{\BODY}}%
 \textbackslash{end\{equation\}}]% alt tag
```

Create the actual LaTeX-formatted equation inside the `lateximage` using the contents of the environment.

```
\LWR@origequation
\BODY% contents collected by NewEnviron
\LWR@origendequation
\end{lateximage}%
}

```

After the environment, if MathJax, print the math to the HTML output for MathJax processing:

```
}[\ifbool{mathjax}{\LWR@addmathjax{equation}{\BODY}}{}]
```

60.4 AMS Math environments

60.4.1 Support macros

Bool `LWR@amsmultline` True if processing a multline environment.

To compensate for `multline`-spefific code, `LWR@amsmultline` is used to add extra horizontal space in `\LWR@htmlmathlabel` if is used in an amsmath environment which is not a `multline` environment and not an `equation`.

```
\newbool{LWR@amsmultline}
\boolfalse{LWR@amsmultline}
```

`\LWR@htmlmathlabel` {⟨*label*⟩}

lwarp points `\ltx@label` here. This is used by `\label` when inside a LaTeX AMS math environment's math display environment.

`\LWR@origltx@label` points to the LaTeX original, modified by lwarp, then by amsmath, then by cleveref.

```
\newcommand*{\LWR@htmlmathlabel}[1]{%
\LWR@traceinfo{LWR@htmlmathlabel #1}%
\ifbool{mathjax}{%
```

The combined LaTeX & HTML label is printed in a `\text` field:

```
\text{
```

Shift the label over to the right side of the environment to avoid over-printing the math:

```
\ifbool{LWR@amsmultline}{}{\hspace*{\totwidth@}}
```

Temporarily end the HTML comment, insert the LaTeX & HTML label, then resume the HTML comment. `\@firstofone` is required to remove extra braces introduced by the amsmath package.)

```
\LWR@htmlclosecomment%
\LWR@origltx@label{#1}%
\LWR@htmlopencomment%
}% text
}% mathjax
{%
\LWR@origltx@label{#1}%
}%
}
```

`\LWR@beginhideamsmath` Starts hiding LaTeX math inside an HTML comment.

```
\newcommand*{\LWR@beginhideamsmath}{
\LWR@stoppars

\LWR@htmlopencomment

\begingroup
\LWR@restoremathlatexformatting
}
```

`\LWR@endhideamsmath` Ends hiding LaTeX math inside an HTML comment.

```
\newcommand*{\LWR@endhideamsmath}{
\endgroup

\LWR@htmlclosecomment

\LWR@startpars
}
```

60.4.2 Environment patches

The following amsmath environments already collect their contents in `\@envbody` for further processing.

For SVG math: Each envrionment is encapsulated inside a `lateximage` environment, along with a special `LWRAMSMATHBODY` argument telling `lateximage` to use as the HTML ALT tag the environment's contents which were automatically captured by the $\mathcal{AMS}$ environment.

For MathJax: Each environment is syched with LaTeX's equation numbers, typeset with LaTeX inside an HTML comment, then printed to HTML output for MathJax to process.

Env `multline`

```
\BeforeBeginEnvironment{multline}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\booltrue{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{multline}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\boolfalse{LWR@amsmultline}
\LWR@addmathjax{multline}{\the\@envbody}
}
{\endlateximage}

}
```

Env `multline*`

```
\BeforeBeginEnvironment{multline*}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\booltrue{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{multline*}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\boolfalse{LWR@amsmultline}
\LWR@addmathjax{multline*}{\the\@envbody}
}
{\endlateximage}

}

```

Env `gather`

```
\BeforeBeginEnvironment{gather}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\boolfalse{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{gather}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\LWR@addmathjax{gather}{\the\@envbody}
}
{\endlateximage}
```

```
}
```

Env gather*

```
\BeforeBeginEnvironment{gather*}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\boolfalse{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{gather*}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\LWR@addmathjax{gather*}{\the\@envbody}
}
{\endlateximage}

}
```

Env align

```
\BeforeBeginEnvironment{align}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\boolfalse{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{align}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\LWR@addmathjax{align}{\the\@envbody}
}
```

```
{\endlateximage}

}
```

Env `align*`

```
\BeforeBeginEnvironment{align*}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\boolfalse{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{align*}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\LWR@addmathjax{align*}{\the\@envbody}
}
{\endlateximage}

}
```

Env `flalign`

```
\BeforeBeginEnvironment{flalign}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\boolfalse{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{flalign}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\LWR@addmathjax{flalign}{\the\@envbody}
```

```
}
{\endlateximage}

}
```

Env `flalign*`

```
\BeforeBeginEnvironment{flalign*}{
\ifbool{mathjax}
{
\LWR@syncmathjax
\boolfalse{LWR@amsmultline}
\LWR@beginhideamsmath
}
{
\lateximage[LWRAMSMATHBODY]
}
}

\AfterEndEnvironment{flalign*}{

\ifbool{mathjax}
{
\LWR@endhideamsmath
\LWR@addmathjax{flalign*}{\the\@envbody}
}
{\endlateximage}

}
```

```
\end{warpHTML}
```

61 Lateximages

A `\lateximage` is typeset on its own PDF page inside an HTML comment which starts on the preceeding page and ends on following page, and instructions are written to `lateximage.txt` for `lwarpmk` to extract the `\lateximage` from the page of the PDF file then generate an accompanying `.svg` file image file. Meanwhile, instructions to show this image are placed into the HTML file after the comment.

An HTML span is created to hold both the HTML comment, which will have the `pdftotext` conversion, and also the link to the final `.svg` image.

A LaTeX label is used to remember which PDF page has the image. A label is used because footnotes, endnotes, and pagenotes may cause the image to appear

at a later time. The label is declared along with the image, and so it correctly remembers where the image finally ended up.

SVG image font size

The size of the math and text used in the SVG image may be adjusted by setting `\LateximageFontSizeName` to a font size name — *without the backslash*, for ex:

```
\renewcommand{\LateximageFontSizeName}{large}
```

for HTML output:

```
\begin{warpHTML}
```

Ctr `LWR@lateximagenumber`

Sequence the images.

```
\newcounter{LWR@lateximagenumber}
\setcounter{LWR@lateximagenumber}{0}
```

Ctr `LWR@lateximagedepth`

Do not create `\lateximage` inside of `\lateximage`.

```
\newcounter{LWR@lateximagedepth}
\setcounter{LWR@lateximagedepth}{0}
```

Declare the `\LWR@file` for writing to generate file `lateximages.txt`:

```
\ifcsdef{LWR@file}{}{\newwrite{\LWR@file}}
```

A few utility macros to write special characters:

```
\edef\LWR@hashmark{\string#} % for use in \write
\edef\LWR@percent{\@percentchar} % for use in \write
```

Ctr `LWR@LIpage`

Used to reference the PDF page number of a lateximage to be written into `lateximages.txt`.

```
\newcounter{LWR@LIpage}
```

```
\end{warpHTML}
```

for HTML & PRINT:

```
\begin{warpall}
```

`\LateximageFontSizeName`

Declares how large to write text in the `\lateximage`. The `.svg` file text size should blend well with the surrounding HTML text size.

Do not include the leading backslash in the name.

```
\newcommand*{\LateximageFontSizeName}{large}
```

```
\end{warpall}
```

for HTML output:

```
\begin{warpHTML}
```

`\LWR@HTMLsanitize` {⟨*text*⟩}

Math expressions are converted to `lateximage`s, and some math environments may contain "&", "<", or ">", which should not be allowed inside an HTML ALT tag, so must convert them to HTML entities.

Two versions follow, depending on expansion needs. There may be a better way...

```
\newcommand{\LWR@HTMLsanitize}[1]{%
\protect\StrSubstitute{\detokenize{#1}}%
{\detokenize{&}}%
{\detokenize{&amp;}}[\LWR@strresult]%
\protect\StrSubstitute{\detokenize\expandafter{\LWR@strresult}}%
{\detokenize{<}}%
{\detokenize{&lt;}}%
[\LWR@strresult]%
\protect\StrSubstitute{\detokenize\expandafter{\LWR@strresult}}%
{\detokenize{>}}%
{\detokenize{&gt;}}%
[\LWR@strresult]%
\protect\StrSubstitute{\detokenize\expandafter{\LWR@strresult}}%
{\detokenize{##}}%
{\#}%
[\LWR@strresult]%
\LWR@strresult%
}
```

`\LWR@HTMLsanitizeexpand` {⟨*text*⟩}

This version expands the argument before sanitizing it.

```
\newcommand{\LWR@HTMLsanitizeexpand}[1]{%
\protect\StrSubstitute{\detokenize\expandafter{#1}}%
{\detokenize{&}}%
{\detokenize{&amp;}}%
[\LWR@strresult]%
\protect\StrSubstitute{\detokenize\expandafter{\LWR@strresult}}%
{\detokenize{<}}%
{\detokenize{&lt;}}%
[\LWR@strresult]%
\protect\StrSubstitute{\detokenize\expandafter{\LWR@strresult}}%
{\detokenize{>}}%
{\detokenize{&gt;}}%
[\LWR@strresult]%
\LWR@strresult%
}
```

Env `lateximage` [⟨*alttag*⟩]

```
\NewDocumentEnvironment{lateximage}{O{image}}{%
\LWR@traceinfo{lateximage: starting on page \arabic{page}}%
\ifthenelse{\cnttest{\value{LWR@lateximagedepth}}{>}{0}}%
```

If nesting inside an already-existing lateximage, simply record one more level:

```
{%
\addtocounter{LWR@lateximagedepth}{1}%
}%
```

Otherwise, this is the outer-most lateximage:

```
{% start of outer-most lateximage
```

Starting a new lateximage:

```
\addtocounter{LWR@lateximagenumber}{1}%
\LWR@traceinfo{lateximage: LWR@lateximagenumber is \arabic{LWR@lateximagenumber}}%
```

While inside a lateximage, do not use mathjax:

```
\boolfalse{mathjax}
```

Be sure that are doing a paragraph:

```
\LWR@ensuredoingapar%
```

Next file:

```
\addtocounter{LWR@externalfilecnt}{1}%
\LWR@traceinfo{lateximage: LWR@externalfilecnt is \arabic{LWR@externalfilecnt}}%
```

Figure out what the next page number will be:

```
\setcounterpageref{LWR@LIpage}{LWRlateximage\theLWR@lateximagenumber}%
\LWR@traceinfo{lateximage: LWR@LIpage is \arabic{LWR@LIpage}}%
```

Create an HTML span which will hold the comment which contains the `pdftotext` translation of the image's page, and also will hold the link to the `.svg` file:

```
\LWR@htmltag{span id="lateximage\arabic{LWR@lateximagenumber}" %
class="lateximagesource"{}} \LWR@orignewline
```

Write instructions to the `lateximages.txt` file:

```
\immediate\write\LWR@file{|\theLWR@LIpage|\theLWR@externalfilecnt|}%
```

Place an open comment tag at the bottom of page; footnotes will be above this tag. This will hide any traces of the lateximage PDF page which were picked up by `pdftotext`.

```
\LWR@htmlopencomment%
\addtocounter{LWR@lateximagedepth}{1}%
```

Start the new PDF page:

```
\LWR@orignewpage%
```

Typeset the image in a "standard" width page and font size:

```
\LWR@origminipage{6in}%
\csuse{LWR@orig\LateximageFontSizeName}%
```

Temporarily restore formatting to its PDF definitions: Do not produce HTML tags for `\hspace`, etc. inside a lateximage.

```
\let\hspace\LWR@orighspace%
\let\rule\LWR@origrule%
\let\,\LWR@origcomma% disable HTML short unbreakable space
\let\textit\LWR@origtextit%
\let\textbf\LWR@origtextbf%
\let\texttt\LWR@origtexttt%
\let\textsc\LWR@origtextsc%
\let\textsf\LWR@origtextsf%
\let\textrm\LWR@origtextrm%
\renewcommand*{\thefootnote}{\fnsymbol{footnote}}%
\let\textsuperscript\LWR@origtextsuperscript%
\let\textsubscript\LWR@origtextsubscript%
\let~\LWR@origtilde%
\let\enskip\LWR@origenskip%
\let\quad\LWR@origquad%
\let\qquad\LWR@origqquad%
\let\tabular\LWR@origtabular%
\let\endtabular\LWR@origendtabular%
\let\newline\LWR@orignewline%
\LWR@origlabel{LWRlateximage\arabic{LWR@lateximagenumber}}%
}% end of outer-most lateximage
}% end of \begin{lateximage}
{% start of \end{lateximage}
\ifthenelse{\cnttest{\value{LWR@lateximagedepth}}{>}{1}}%
```

If nesting inside an already-existing lateximage, simply record one more level:

```
{%
\addtocounter{LWR@lateximagedepth}{-1}%
}%
```

if this is the outer-most lateximage:

```
{% end of outer-most lateximage
```

Finish the lateximage minipage and start a new PDF page:

```
\LWR@origendminipage%
\LWR@orignewpage%
\LWR@origscriptsize%
```

Close the HTML comment which encapsulated any traces of the lateximage picked up by `pdftotext`:

```
\LWR@htmlclosecomment{}\LWR@orignewline%
\LWR@traceinfo{lateximage: The page after the image is \arabic{page}}%
```

Create a link to the lateximage, allowing its natural height:

If the alt tag is given as "LWRAMSMATHBODY", then use the text collected by the amsmath multline, gather, or align environments.

```
\ifthenelse{\equal{#1}{LWRAMSMATHBODY}}%
{%
\LWR@subinlineimage[%
\LWR@HTMLsanitizeexpand{\detokenize\expandafter{\the\@envbody}}%
]%
{lateximage}%
{lateximages\OSPathSymbol{}lateximage-\theLWR@externalfilecnt}%
{svg}%
{}%
}%
{%
\LWR@subinlineimage[#1]{lateximage}%
{lateximages\OSPathSymbol{}lateximage-\theLWR@externalfilecnt}{svg}{}%
}%
% \LWR@orignewline% Removed to prevent extra space.
```

Be sure that are doing a paragraph:

```
\LWR@ensuredoingapar%
```

Close the HTML span which has the `pdftotext` comment and also the link to the `.svg` image:

```
\LWR@htmltag{/span}%
\ifbool{HTMLDebugComments}{%
\LWR@htmlcomment{End of lateximage}%
}{}%
% \LWR@orignewline% Removed to prevent extra space.
```

Undo one lateximage level:

```
\addtocounter{LWR@lateximagedepth}{-1}%
}% end of outer-most lateximage
\LWR@traceinfo{lateximage: done}
}%
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\newenvironment{lateximage}[1][]{\minipage{\linewidth}}{\endminipage}
\end{warpprint}
```

62 center, flushleft, flushright

for HTML output:

```
\begin{warpHTML}
```

Env `center` Replace `center` functionality with CSS tags:

```
\renewenvironment*{center}
{
\LWR@forcenewpage
\BlockClass{center}
}
{\endBlockClass}
```

Env `flushright`

```
\renewenvironment*{flushright}
{
\LWR@forcenewpage
\BlockClass{flushright}
}
{\endBlockClass}
```

Env `flushleft`

```
\renewenvironment*{flushleft}
{
\LWR@forcenewpage
\BlockClass{flushleft}
}
{\endBlockClass}

\end{warpHTML}
```

63 Siunitx

Pkg siunitx

⚠ per-mode Do not use `per-mode=fraction`, which cannot be seen by the final `pdftotext` conversion.

for HTML output:
```
\begin{warpHTML}
```

Options for siunitx:

```
\PassOptionsToPackage{
detect-mode=true,
per-mode=symbol,% fraction is not seen by pdftotext
text-celsius = {\HTMLentity{deg}C},
text-degree = {\HTMLentity{deg}},
}{siunitx}

\end{warpHTML}
```

64 Graphics

Pkg graphics

Pkg graphicx

⚠ graphics vs. graphicx If using the older graphics syntax, use both optional arguments for `\includegraphics`. A single optional parameter is interpreted as the newer graphicx syntax. Note that viewports are not supported by warp; the entire image will be shown.

⚠ viewports

⚠ `\graphicspath` `\graphicspath` only works for a single directory; all graphics must be in this directory.

units For `\includegraphics`, avoid `px` and % units for width and height, or enclose them inside `warpHTML` environments. For font-proportional image sizes, use `ex` or `em`. For fixed-sized images, use `cm`, `mm`, `in`, `pt`, or `pc`. Using the keys `width=.5\linewidth`, or similar for `\textwidth` or `\textheight` to give fixed-sized images proportional to a 6 by 9 inch text area.

options `\includegraphics` accepts `width` and `height`, `origin`, `rotate` and `scale`, plus a new `class` key.

HTML class With HTML output, `\includegraphics` accepts an optional `class=xyz` keyval combination, and if this is given then the HTML output will include that class for the image. The class is ignored for print output.

⚠ image file types

For `\includegraphics` the user should provide both `.pdf` and `.svg` images, but always refer to `.pdf` images in the document source. All `\includegraphics` references to `.pdf` will automatically be changed to `.svg` for HTML output, and will be left as `.pdf` for print output. Images may also be `.jpg` and `.png`, and will be used as-is for either output.

`\rotatebox`

`\rotatebox` accepts the optional `origin` key.

⚠ browser support

`\rotatebox`, `\scalebox`, and `\reflectbox` depend on modern browser support. The CSS3 standard declares that when an object is transformed the whitespace which they occupied is preserved, unlike LaTeX, so expect some ugly results for scaling and rotating.

for HTML output:

```
\begin{warpHTML}
```

64.1 \graphicspath

`\graphicspath` {⟨*path*⟩}

```
\newcommand*{\thisgraphicspath}{{}}
\renewcommand*{\graphicspath}[1]{\renewcommand*{\thisgraphicspath}{#1}}
```

`\DeclareGraphicsExtensions` {⟨*list*⟩}

`\DeclareGraphicsRule` {⟨⟩} {⟨⟩} {⟨⟩} {⟨⟩}

```
\renewcommand*{\DeclareGraphicsExtensions}[1]{}
\renewcommand*{\DeclareGraphicsRule}[4]{}
```

64.2 Length conversions and graphics options

⚠ whitespace

A scaled image in LaTeX by default takes only as much space on the page as it requires, but HTML browsers use as much space as the original unscaled image would have taken, with the scaled image over- or under-flowing the area.

```
\renewcommand*{\unitspace}{}
```

Used to store the user's selected dimensions and HTML class.

The class defaults to "inlineimage" unless changed by a `class=xyx` option.

```
\newlength{\LWR@igwidth}
\newlength{\LWR@igheight}
```

```
\newcommand*{\LWR@igwidthstyle}{}
\newcommand*{\LWR@igheightstyle}{}
\newcommand*{\LWR@igorigin}{}
\newcommand*{\LWR@igangle}{}
\newcommand*{\LWR@igxscale}{1}
\newcommand*{\LWR@igyscale}{1}
\newcommand*{\LWR@igclass}{inlineimage}
```

Set the actions of each of the key/value combinations for `\includegraphics`. Many are ignored.

If an optional width was given, set an HTML style:

```
\define@key{igraph}{width}{%
\setlength{\LWR@igwidth}{#1}%
\ifthenelse{\lengthtest{\LWR@igwidth > 0pt}}%
{%
```

Default to use the converted fixed length given:

```
\uselengthunit{PT}%
\renewcommand*{\LWR@igwidthstyle}{width:\rndprintlength{\LWR@igwidth}}%
```

If ex or em dimensions were given, use those instead:

```
\IfEndWith{#1}{ex}%
{\renewcommand*{\LWR@igwidthstyle}{width:#1}}% yes ex
{}% not ex
\IfEndWith{#1}{em}%
{\renewcommand*{\LWR@igwidthstyle}{width:#1}}% yes em
{}% not em
\IfEndWith{#1}{\%}%
{\renewcommand*{\LWR@igwidthstyle}{width:#1}}% yes percent
{}% not percent
\IfEndWith{#1}{px}%
{\renewcommand*{\LWR@igwidthstyle}{width:#1}}% yes px
{}% not px
}{}% end of length > 0pt
}
```

If an optional height was given, set an HTML style:

```
\define@key{igraph}{height}{%
\setlength{\LWR@igheight}{#1}%
\ifthenelse{\lengthtest{\LWR@igheight > 0pt}}%
{%
```

Default to use the converted fixed length given:

```
\uselengthunit{PT}%
\renewcommand*{\LWR@igheightstyle}{%
height:\rndprintlength{\LWR@igheight} %
}%
```

If ex or em dimensions were given, use those instead:

```
\IfEndWith{#1}{ex}%
{\renewcommand*{\LWR@igheightstyle}{height:#1}}% yes ex
{}% not ex
\IfEndWith{#1}{em}%
{\renewcommand*{\LWR@igheightstyle}{height:#1}}% yes em
{}% not em
\IfEndWith{#1}{\%}%
{\renewcommand*{\LWR@igheightstyle}{height:#1}}% yes percent
{}% not percent
\IfEndWith{#1}{px}%
{\renewcommand*{\LWR@igheightstyle}{height:#1}}% yes px
{}% not px
}{}% end of length > 0pt
}
```

Handle origin key:

```
\define@key{igraph}{origin}{%
\renewcommand*{\LWR@igorigin}{#1}%
}
```

Handle angle key:

```
\define@key{igraph}{angle}{\renewcommand*{\LWR@igangle}{#1}}
```

Handle class key:

```
\define@key{igraph}{class}{\renewcommand*{\LWR@igclass}{#1}}

```

It appears that graphicx does not have separate keys for `xscale` and `yscale`. `scale` adjusts both at the same time.

```
\define@key{igraph}{scale}{%
\renewcommand*{\LWR@igxscale}{#1}%
\renewcommand*{\LWR@igyscale}{#1}}
```

Numerous ignored keys:

```
\define@key{igraph}{bb}{}
\define@key{igraph}{bbllx}{}
\define@key{igraph}{bblly}{}
```

```
\define@key{igraph}{bburx}{}
\define@key{igraph}{bbury}{}
\define@key{igraph}{natwidth}{}
\define@key{igraph}{natheight}{}
\define@key{igraph}{hiresbb}{}
\define@key{igraph}{viewport}{}
\define@key{igraph}{trim}{}
\define@key{igraph}{totalheight}{}
\define@key{igraph}{keepaspectratio}{}
\define@key{igraph}{clip}{}
\define@key{igraph}{draft}{}
\define@key{igraph}{type}{}
\define@key{igraph}{ext}{}
\define@key{igraph}{read}{}
\define@key{igraph}{command}{}
```

`\LWR@rotstyle` {⟨*prefix*⟩} {⟨*degrees*⟩}

Prints the rotate style with the given prefix.

`prefix` is `-ms-` or `-webkit-` or nothing, and is used to generate three versions of the `transform:rotate` style.

```
\newcommand*{\LWR@rotstyle}[2]{%
 #1transform:rotate(-#2deg);
}
```

`\LWR@scalestyle` {⟨*prefix*⟩} {⟨*xscale*⟩} {⟨*yscale*⟩}

Prints the scale style with the given prefix.

`prefix` is `-ms-` or `-webkit-` or nothing, and is used to generate three versions of the `transform:scale` style.

```
\newcommand*{\LWR@scalestyle}[3]{%
 #1transform:scale(#2,#3);
}
```

64.3 \includegraphics

Bool `LWR@infloatrow` Used to compute `\linewidth`.

```
\newbool{LWR@infloatrow}
\boolfalse{LWR@infloatrow}
```

```
\newcommand*{\LWR@imageextension}{}
\newcommand*{\LWR@expgraphicsfilename}{}
```

`\LWR@includegraphicsb` * [⟨*2: options*⟩] [⟨*3: options*⟩] {⟨*4: filename*⟩}

graphics syntax is `\includegraphics` * [⟨*llx,lly*⟩] [⟨*urx,ury*⟩] {⟨*file*⟩}

graphicx syntax is `\includegraphics` [⟨*key values*⟩] {⟨*file*⟩}

If `#3` is empty, only one optional argument was given, thus graphicx syntax.

```
\NewDocumentCommand{\LWR@includegraphicsb}{s o o m}
{%
```

Start the image tag on a new line, allow PDF output word wrap:

```
\LWR@origtilde \LWR@orignewline%
```

Temporarily compute `\linewidth`, `\textwidth`, `\textheight` arguments with a 6x9 inch size until the next `\endgroup`.

```
\ifthenelse{\cnttest{\value{LWR@minipagedepth}}{=}{0}}{%
\ifbool{LWR@infloatrow}%
{}
{% not in a minipage or a floatrow:
\setlength{\linewidth}{6in}%
\setlength{\textwidth}{6in}%
\setlength{\textheight}{9in}%
}%
}{}%
```

See if can find the image by adding an extension:

Preference is `svgz`, then `svg`, `gif`, `png`, and `jpg`.

`\detokenize\expandafter` allows underscore characters in filenames.

```
\edef\LWR@expgraphicsfilename{#4}
\renewcommand*{\LWR@imageextension}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.jpg}%
{\renewcommand*{\LWR@imageextension}{.jpg}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.JPG}%
{\renewcommand*{\LWR@imageextension}{.JPG}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.png}%
{\renewcommand*{\LWR@imageextension}{.png}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.PNG}%
{\renewcommand*{\LWR@imageextension}{.PNG}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.gif}%
{\renewcommand*{\LWR@imageextension}{.gif}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.GIF}%
{\renewcommand*{\LWR@imageextension}{.GIF}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.svg}%
```

```
{\renewcommand*{\LWR@imageextension}{.svg}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.SVG}%
{\renewcommand*{\LWR@imageextension}{.SVG}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.svgz}%
{\renewcommand*{\LWR@imageextension}{.svgz}}{}%
\IfFileExists{\detokenize\expandafter\thisgraphicspath\LWR@expgraphicsfilename.SVGZ}%
{\renewcommand*{\LWR@imageextension}{.SVGZ}}{}%
```

Convert a PDF extension to SVG, leave the result in `\LWR@strresult`:

Must also `\detokenize .pdf` and `.svg` comparison strings.

```
\StrSubstitute{\detokenize\expandafter{\LWR@expgraphicsfilename}}%
{\detokenize{.pdf}}{\detokenize{.svg}}[\LWR@strresult]%
%
\StrSubstitute{\LWR@strresult}%
{\detokenize{.PDF}}{\detokenize{.SVG}}[\LWR@strresult]%
```

For correct em sizing during the width and height conversions:

```
\large%
```

Reset some defaults, possibly will be changed below if options were given:

```
\setlength{\LWR@igwidth}{0pt}%
\setlength{\LWR@igheight}{0pt}%
\renewcommand*{\LWR@igwidthstyle}{}%
\renewcommand*{\LWR@igheightstyle}{}%
\renewcommand*{\LWR@igorigin}{}%
\renewcommand*{\LWR@igangle}{}%
\renewcommand*{\LWR@igxscale}{1}%
\renewcommand*{\LWR@igyscale}{1}%
\renewcommand*{\LWR@igclass}{inlineimage}%
```

If `#3` is empty, only one optional argument was given, thus graphicx syntax:

```
\IfValueTF{#3}{}{%
\IfValueTF{#2}%
{\setkeys{igraph}{#2}}%
{\setkeys{igraph}{}}%
}%
```

Create the HTML reference with the graphicspath, filename, extension, alt tag, style, and class.

The `\LWR@origtilde` adds space between tags in case this is being done inside a `\savebox` where `\newline` has no effect.

```
\href{\thisgraphicspath\LWR@strresult\LWR@imageextension}%
```

```
{% start of href
\LWR@htmltag{% start of image tags
img src="\thisgraphicspath\LWR@strresult\LWR@imageextension" \LWR@orignewline
\LWR@origtilde{} alt="\LWR@strresult" \LWR@orignewline
```

Only include a style tag if a width, height, angle, or scale was given:

```
\ifthenelse{
\NOT\equal{\LWR@igwidthstyle}{} \OR
\NOT\equal{\LWR@igheightstyle}{} \OR
\NOT\equal{\LWR@igorigin}{} \OR
\NOT\equal{\LWR@igangle}{} \OR
\NOT\equal{\LWR@igxscale}{1} \OR
\NOT\equal{\LWR@igyscale}{1}
}%
{\LWR@origtilde{} style="%
\ifthenelse{\NOT\equal{\LWR@igwidthstyle}{}}%
{\LWR@igwidthstyle;}{}%
\ifthenelse{\NOT\equal{\LWR@igheightstyle}{}}%
{\LWR@igheightstyle;}{}%
\ifthenelse{\NOT\equal{\LWR@igorigin}{}}%
{\LWR@origtilde{} transform-origin: \LWR@originnames{\LWR@igorigin}; \LWR@orignewline}{}%
\ifthenelse{\NOT\equal{\LWR@igangle}{}}%
{%
\LWR@rotstyle{-ms-}{\LWR@igangle}%
\LWR@rotstyle{-webkit-}{\LWR@igangle}%
\LWR@rotstyle{}{\LWR@igangle%
}}{}%
\ifthenelse{\NOT\equal{\LWR@igxscale}{1}\OR%
\NOT\equal{\LWR@igyscale}{1}}%
{\LWR@scalestyle{-ms-}{\LWR@igxscale}{\LWR@igyscale}%
\LWR@scalestyle{-webkit-}{\LWR@igxscale}{\LWR@igyscale}%
\LWR@scalestyle{}{\LWR@igxscale}{\LWR@igyscale}}{}%
" \LWR@orignewline}{}%
```

Set the class:

```
\LWR@origtilde{} class="\LWR@igclass" \LWR@orignewline%
}% end of image tags
}% end of href
\endgroup
```

Return to small-sized output:

```
\LWR@origscriptsize
}
```

`\includegraphics` [⟨*key=val*⟩] {⟨*filename*⟩}

Handles width and height, converted to fixed width and heights.

Converts any `.pdf` references to `.svg` for HTML

The user should always refer to `.pdf` in the document source.

```
\renewcommand*{\includegraphics}
{%
```

This graphic should trigger an HTML paragraph even if alone, so ensure that are doing paragraph handling:

```
\LWR@ensuredoingapar%
\begingroup%
\LWR@includegraphicsb%
}
```

```
\end{warpHTML}
```

for PRINT output: For print output, accept and then discard the new `class` key:

```
\begin{warpprint}
\define@key{Gin}{class}{}
\end{warpprint}
```

64.4 \rotatebox, \scalebox, \reflectbox

for HTML output:

```
\begin{warpHTML}
```

`\LWR@rotboxorigin` Holds the origin key letters.

```
\newcommand*{\LWR@rotboxorigin}{}
```

`\LWR@originname` {⟨*letter*⟩}

Given one LaTeX origin key value, translate into an HTML origin word:

```
\newcommand*{\LWR@originname}[1]{%
\ifthenelse{\equal{#1}{t}}{top}{}%
\ifthenelse{\equal{#1}{b}}{bottom}{}%
\ifthenelse{\equal{#1}{c}}{center}{}%
\ifthenelse{\equal{#1}{l}}{left}{}%
\ifthenelse{\equal{#1}{r}}{right}{}%
}
```

`\LWR@originnames` {⟨*letters*⟩}

Given one- or two-letter LaTeX origin key values, translate into HTML origin words:

```
\newcommand*{\LWR@originnames}[1]{%
\StrChar{#1}{1}[\LWR@strresult]%
\LWR@originname{\LWR@strresult}
\StrChar{#1}{2}[\LWR@strresult]%
\LWR@originname{\LWR@strresult}
}
```

Handle the origin key for `\rotatebox`:

```
\define@key{krotbox}{origin}{%
\renewcommand*{\LWR@rotboxorigin}{#1}%
}
```

These keys are ignored:

```
\define@key{krotbox}{x}{}
\define@key{krotbox}{y}{}
\define@key{krotbox}{units}{}
```

`\rotatebox` [⟨*keyval list*⟩] {⟨*angle*⟩} {⟨*text*⟩}

Will `\let\rotatebox\LWR@rotatebox` at `\LWR@LwarpStart`, in case `\rotatebox` was over-written by a later package load.

```
\NewDocumentCommand{\LWR@rotatebox}{O{} m +m}{%
```

Reset the origin to "none-given":

```
\renewcommand*{\LWR@rotboxorigin}{}
```

Process the optional keys, which may set `\LWR@rotateboxorigin`:

```
\setkeys{krotbox}{#1}%
```

Select `inline-block` so that HTML will transform this span:

```
\LWR@htmltagc{span style="display: inline-block; %
```

If an origin was given, translate and print the origin information:

```
\ifthenelse{\NOT\equal{\LWR@rotboxorigin}{}}%
{transform-origin: \LWR@originnames{\LWR@rotboxorigin};\LWR@origtilde}{}%
```

Print the rotation information:

```
\LWR@rotstyle{-ms-}{#2} %
\LWR@rotstyle{-webkit-}{#2} %
\LWR@rotstyle{}{#2} %
"{}}\LWR@orignewline%
```

Print the text to be rotated:

```
\begin{LWR@nestspan}%
#3%
```

Close the span:

```
\LWR@htmltagc{/span}%
\end{LWR@nestspan}%
}
```

`\scalebox` {⟨*h-scale*⟩} [⟨*v-scale*⟩] {⟨*text*⟩}

Will `\let\scalebox\LWR@scalebox` at `\LWR@LwarpStart`, in case `\scalebox` was over-written by a later package load.

```
\NewDocumentCommand{\LWR@scalebox}{m o m}{%
```

Select `inline-block` so that HTML will transform this span:

```
\LWR@htmltagc{span style="display: inline-block; %
```

Print the scaling information:

```
\LWR@scalestyle{-ms-}{#1}{\IfNoValueTF{#2}{#1}{#2}} %
\LWR@scalestyle{-webkit-}{#1}{\IfNoValueTF{#2}{#1}{#2}} %
\LWR@scalestyle{}{#1}{\IfNoValueTF{#2}{#1}{#2}} %
"{}}%
```

Print the text to be scaled:

```
\begin{LWR@nestspan}%
#3%
```

Close the span:

```
\LWR@htmltagc{/span}%
\end{LWR@nestspan}%
}
```

`\reflectbox` {⟨*text*⟩}

Will \let\reflectbox\LWR@reflectbox at \LWR@LwarpStart, in case \reflectbox was over-written by a later package load.

```
\newcommand{\LWR@reflectbox}[1]{\LWR@scalebox{-1}[1]{#1}}
```

```
\end{warpHTML}
```

64.5 Null functions

These functions are not supported by lwarp's HTML conversion.

for HTML output:

```
\begin{warpHTML}
```

\resizebox {⟨*h-length*⟩} {⟨*v-length*⟩} {⟨*text*⟩}

Simply prints its text argument.

```
\renewcommand{\resizebox}[3]{#3}
```

```
\end{warpHTML}
```

65 Cleverref

Pkg cleveref

cleveref package is used as-is with minor patches.

loading order

cleveref and the following associated macro patches are automatically preloaded at the end of the preamble via \AtEndPreamble and \AfterEndPreamble. This is done because the HTML conversion requires cleveref. The user's document may not require cleveref, thus the user may never explicitly load it, so during HTML output lwarp loads it last. If the user's document preamble uses cleveref options, or functions such as \crefname, then cleveref may be loaded in the user's preamble near the end, and lwarp's additional loading of cleveref will have no effect.

Table 9 on 264 shows the data structure of the label/reference system as revised by lwarp and cleveref.

A few patches allow cleveref to work as-is:

for HTML output:

```
\begin{warpHTML}
```

\AtEndPreable forces cleveref to be loaded last:

The following patches are applied after cleveref has loaded, and after \AtBeginDocument:

```
\AfterEndPreamble{
```

\@@setcref {⟨*kindofref*⟩} {⟨*label*⟩}

```
\renewcommand*{\@@setcref}[2]{#1{\ref{#2}}{}{}}
```

\@@setcrefrange {⟨*text*⟩} {⟨*label*⟩} {⟨*label*⟩}

```
\renewcommand{\@@setcrefrange}[3]{%
#1{\ref{#2}}{\ref{#3}}{}{}{}{}}
```

\cpagerefFor Redefinable word between "page(s)" and the page numbers.

```
\newcommand*{\cpagerefFor}{for}
```

\@@setcpageref {⟨*typeofref*⟩} {⟨*label*⟩}, where typeofref is "page" or "pages"

```
\renewcommand*{\@@setcpageref}[2]{%
#1{\cpagerefFor\ \cref{#2}}{}{}%
}

\renewcommand{\@@setcpagerefrange}[3]{%
#1{\cpagerefFor\ \cref{#2}}{\cref{#3}}{}{}{}{}}
}% AfterEndPreamble
```

Remember and patch some label-related defintions. These will be further encased and patched by other packages later.

```
\let\LWR@origlabel\label
\let\label\LWR@newlabel
\let\LWR@origref\ref
\let\ref\LWR@newref% \end{ syntax highlighting
\let\LWR@origpageref\pageref
\let\pageref\LWR@newpageref


\end{warpHTML}
```

66 Picture

Env picture The picture environment is enclosed inside a \lateximage.

for HTML output:

```
\begin{warpHTML}
```

Env `picture`

```
\BeforeBeginEnvironment{picture}{%
\lateximage%
\let\makebox\LWR@origmakebox%
}

\AfterEndEnvironment{picture}{\endlateximage}
```

```
\end{warpHTML}
```

67 Boxes and Minipages

A CSS flexbox is used for minipages and parboxes, allowing external and internal vertical positioning.

placement — Minipages and parboxes will be placed side-by-side in HTMLunless you place a `\newline` between them.

⚠ inline — A line of text with an inline minipage or parbox will have the minipage or parbox placed onto its own line, because a paragraph is a block element and cannot be made `inline-block`.

side-by-side — Side-by-side minipages may be separated by `\quad`, `\qquad`, `\enskip`, `\hspace`, `\hfill`, or a `\rule`. When inside a `center` environment, the result is similar in print and HTML. Paragraph tags are surpressed between side-by-side minipages and these spacing commands, but not at the start or end of the paragraph.

in a span — There is limited support for minipages inside an HTML `<span>`. An HTML `<div>` cannot appear inside a `<span>`. While in a `<span>`, minipages and parboxes are ignored. Use `\newline` or `\par` for an HTML break.

size — When using `\linewidth`, `\textwidth`, and `\textheight`, widths and heights are scaled proportionally to a 6×9 inch text area.

no-width minipages — A minipage of width exactly `\linewidth` is automatically given no HTML width.

full-width minipages — A new macro `\minipagefullwidth` requests that the next minipage be generated without an HTML `width` tag, allowing it to be the full width of the display rather than the fixed width given.

⚠ text alignment — Nested minipages adopt their parent's text alignment in HTML, whereas in regular LaTeX PDF output they do not. Use a `flushleft` or similar environment in the child minipage to force a text alignment.

for HTML output:

```
\begin{warpHTML}
```

67.1 Counters and lengths

Ctr `LWR@minipagedepth` Used to only reset the line width at the outermost minipage.

```
\newcounter{LWR@minipagedepth}
\setcounter{LWR@minipagedepth}{0}
```

Len `\WR@minipagewidth` Used to convert the width into printable units.

```
\newlength{\LWR@minipagewidth}
```

Len `\WR@minipageheight` Used to convert the height into printable units.

```
\newlength{\LWR@minipageheight}
```

Remember the original definitions:

```
\let\LWR@origminipage\minipage
\let\LWR@origendminipage\endminipage
```

67.2 Footnote handling

Also see section 41 for other forms of footnotes.

67.3 Minipage handling

`\LWR@endminipage` Used to close a minipage.

Copied the LaTeX definition and modified to create a `mpfootnotes` div class:

```
\def\LWR@endminipage{%
    \par
    \unskip
    \ifvoid\@mpfootins\else
      \vskip\skip\@mpfootins
      \normalcolor
\LWR@htmldivclass{mpfootnotes}
\LWR@origmedskip
      \unvbox\@mpfootins
\LWR@htmldivclassend{mpfootnotes}
    \fi
    \@minipagefalse
  \color@endgroup
  \egroup
  \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
```

`\LWR@subminipage` Used to create a PDF minipage without creating an HTML minipage. This allows footnotes to appear at the bottom of the minipage instead of the bottom of the HTML page.

```
\newcommand*{\LWR@subminipage}{%
\LWR@stoppars
\LWR@origminipage{6in}
```

`\raggedright` cancels hyphenation, which will be done by HTML instead.

```
\LWR@origraggedright%
```

Resume paragraph tag handling for the contents of the minipage:

```
\LWR@startpars%
}
```

`\LWR@endsubminipage` Closes the subminipage.

```
\newcommand*{\LWR@endsubminipage}{%
\LWR@stoppars%
\LWR@endminipage% The following empty line is required:

}
```

Bool `LWR@minipagefullwidth` Should the next minipage have no HTML `width`?

```
\newbool{LWR@minipagefullwidth}
\boolfalse{LWR@minipagefullwidth}
```

`\minipagefullwidth` Requests that the next minipage have no `width` tag in HTML:

for HTML output:

```
\newcommand*{\minipagefullwidth}{\booltrue{LWR@minipagefullwidth}}
\end{warpHTML}
```

for PRINT output:

```
\begin{warpprint}
\newcommand*{\minipagefullwidth}{}
\end{warpprint}
```

for HTML output:

```
\begin{warpHTML}
```

Bool `LWR@minipagethispar` Has a minipage been seen this paragraph? If true, prevents paragraph tags around horizontal space between minipages.

```
\newbool{LWR@minipagethispar}
\boolfalse{LWR@minipagethispar}
```

Env **minipage** [⟨*vert position*⟩] [⟨*height*⟩] [⟨*inner vert position*⟩] {⟨*width*⟩}

The vertical positions may be 'c', 't', or 'b'. The inner position may also be 's'.

When using `\linewidth`, `\textwidth`, or `\textheight`, these are scaled proportionally to a 6×9 inch text area.

```
\RenewDocumentEnvironment{minipage}{O{t} o O{t} m}
{%
```

Pre-compute the given width and height:

Reset the text area if are starting the outer-most minipage:

```
\LWR@traceinfo{starting minipage of width #4}%
\uselengthunit{in}%
\setlength{\LWR@minipagewidth}{#4}%
\ifthenelse{\cnttest{\value{LWR@minipagedepth}}{=}{0}}{%
\addtolength{\LWR@minipagewidth}{3em}% room for frames
\setlength{\linewidth}{6in}%
\setlength{\textwidth}{6in}%
\setlength{\textheight}{9in}%
}{}%
\LWR@traceinfo{computed width is \rndprintlength{\LWR@minipagewidth}}
\addtocounter{LWR@minipagedepth}{1}%
\setlength{\LWR@minipageheight}{\textheight}% default unless specified
\IfValueTF{#2}{\setlength{\LWR@minipageheight}{#2}}{}%
```

LaTeX wants to start a paragraph for the new minipage, then start a paragraph again for the contents of the minipage, so cancel the paragraph tag handling until the minipage has begun.

```
\LWR@stoppars%
```

Create the `<div>` tag with optional alignment style:

```
\LWR@traceinfo{minipage: creating div class}%
\LWR@orignewpage%
\LWR@htmltag{div class="minipage" style="%
\ifthenelse{\equal{#1}{t}}{vertical-align: bottom ; }{}%
\ifthenelse{\equal{#1}{c}}{vertical-align: middle ; }{}%
\ifthenelse{\equal{#1}{b}}{vertical-align: top ; }{}%
\ifthenelse{\equal{#3}{t}}{justify-content: flex-start ; }{}%
\ifthenelse{\equal{#3}{c}}{justify-content: center ; }{}%
\ifthenelse{\equal{#3}{b}}{justify-content: flex-end ; }{}%
\ifthenelse{\equal{#3}{s}}{justify-content: space-between ; }{}%
```

Print the width and optional height styles:

```
\LWR@traceinfo{minipage: about to print the width of \rndprintlength{\LWR@minipagewidth}}%
\uselengthunit{PT}%
\ifbool{LWR@minipagefullwidth}%
{\boolfalse{LWR@minipagefullwidth}}%
{%
\ifthenelse{\lengthtest{#4}=\linewidth}%
{}%
{width:\rndprintlength{\LWR@minipagewidth} ; }%
}%
\LWR@traceinfo{minipage: about to print the height}%
\IfValueTF{#2}{height:\rndprintlength{\LWR@minipageheight} ; }{}%
"{}}%
```

Finish with an empty line to start LaTeX minipage processing on a new line. Use a large minipage area to avoid the unnecessary wrapping of tags.

```

\LWR@origminipage{6in}% The preceding empty line is required.
```

Set the user-accessible minipage and text width and height values inside the minipage. These do not affect the actual size of the large minipage created by `\LWR@origminipage` above, but are used by any reference to `\linewidth`, etc. inside the PDF minipage being created here.

```
\setlength{\linewidth}{#4}% the original width
\setlength{\textwidth}{6in}%
\setlength{\textheight}{9in}%
```

`\raggedright` cancels hyphenation, which will be done by HTML instead.

```
\LWR@origraggedright%
```

Resume paragraph tag handling for the contents of the minipage:

```
\LWR@startpars%
\LWR@traceinfo{minipage: finished starting the minipage}%
}
```

End the environment with LaTeX processing and closing tag:

```
{%
\LWR@stoppars%
\LWR@endminipage% The following empty line is required:

\LWR@htmldivclassend{minipage}%
\vspace{1\baselineskip}% required for subcaption
\addtocounter{LWR@minipagedepth}{-1}%
```

```
\LWR@startpars%
```

Prevent paragraph tags around horizontal white space until the start of the next paragraph:

```
\global\booltrue{LWR@minipagethispar}%
}
```

67.4 Parbox, makebox, framebox, fbox, raisebox

\parbox [⟨*pos*⟩] [⟨*height*⟩] [⟨*inner-pos*⟩] {⟨*width*⟩} {⟨*text*⟩}

A parbox uses the minipage code:

```
\RenewDocumentCommand{\parbox}{O{t} o O{t} m +m}
{
\LWR@traceinfo{parbox of width #4}%
\begin{minipage}[#1][#2][#3]{#4}
#5
\end{minipage}
}
```

\makebox [⟨*width*⟩] [⟨*pos*⟩] {⟨*text*⟩}

Width and position are ignored.

```
\let\LWR@origmakebox\makebox

\RenewDocumentCommand{\makebox}{o o m}{%
\mbox{#3}
}
```

\framebox [⟨*width*⟩] [⟨*pos*⟩] {⟨*text*⟩}

Width and position are ignored.

```
\RenewDocumentCommand{\framebox}{o o m}{%
\fbox{#3}
}
```

\fbox {⟨*text*⟩}

```
\let\LWR@origfbox\fbox
%
\renewcommand*{\fbox}[1]{%
```

```
\InlineClass{framebox}{#1}%
}
```

\raisebox {⟨*raiselen*⟩} [⟨*height*⟩] [⟨*depth*⟩] {⟨*text*⟩}

```
\RenewDocumentCommand{\raisebox}{m o o m}{%
#4%
}
```

```
\end{warpHTML}
```

68 Direct formatting

\textbf, etc. are supported, but \bfseries, etc. are not yet supported.

For high-level block and inline custom CSS classes, see section 35.7.

for HTML output:

```
\begin{warpHTML}
```

\emph {⟨*text*⟩}

```
\renewcommand{\emph}[1]{\LWR@htmlspan{em}{#1}}
```

\textmd {⟨*text*⟩}

```
\renewcommand{\textmd}[1]{\LWR@htmlspan{textmd}{#1}}
```

\textbf {⟨*text*⟩}

```
\renewcommand{\textbf}[1]{\LWR@htmlspan{b}{#1}}
```

\textrm {⟨*text*⟩}

```
\renewcommand{\textrm}[1]{\InlineClass{textrm}{#1}}
```

\textsf {⟨*text*⟩}

```
\renewcommand{\textsf}[1]{\InlineClass{textsf}{#1}}
```

\texttt {⟨*text*⟩}

```
\renewcommand{\texttt}[1]{\LWR@htmlspan{kbd}{#1}}
```

\textup {⟨*text*⟩}

```
\renewcommand{\textup}[1]{\InlineClass{textup}{#1}}
```

\textit {⟨*text*⟩}

```
\renewcommand{\textit}[1]{\LWR@htmlspan{i}{#1}}
```

\textsc {⟨*text*⟩}

```
\renewcommand{\textsc}[1]{\InlineClass{textsc}{#1}}
```

\textnormal {⟨*text*⟩}

```
\renewcommand{\textnormal}[1]{\textmd{\textrm{\textup{#1}}}}
```

\mdseries

```
\renewcommand*{\mdseries}{}
```

\bfseries

```
\renewcommand*{\bfseries}{}
```

\rmfamily

```
\renewcommand*{\rmfamily}{}
```

\sffamily

```
\renewcommand*{\sffamily}{}
```

\ttfamily

```
\renewcommand*{\ttfamily}{}
```

\upshape

```
\renewcommand*{\upshape}{}
```

\itshape

```
\renewcommand*{\itshape}{}
```

`\scshape`

```
\renewcommand*{\scshape}{}
```

`\scshape`

```
\renewcommand*{\normalfont}{}
```

`\sp` {⟨*text*⟩}

For siunitx. Must work in math mode.

```
\renewcommand{\sp}[1]{\text{<sup>#1</sup>}{}}
```

`\sb` {⟨*text*⟩}

For siunitx. Must work in math mode.

```
\renewcommand{\sb}[1]{\text{<sub>#1</sub>}{}}
```

`\textsuperscript` {⟨*text*⟩}

```
\renewcommand{\textsuperscript}[1]{\LWR@htmlspan{sup}{#1}}
```

`\textsubscript` {⟨*text*⟩}

```
\renewcommand{\textsubscript}[1]{\LWR@htmlspan{sub}{#1}}
```

`\up` {⟨*text*⟩} Prints superscript.

This is `\let` at the beginning of the document in case some other package has changed the definition.

```
\AtBeginDocument{\let\up\textsuperscript}
```

`\fup` {⟨*text*⟩} Prints superscript.

Supports fmtcount package.

This is `\let` at the beginning of the document in case some other package has changed the definition.

```
\AtBeginDocument{\let\fup\textsuperscript}
```

\hfill

```
\renewcommand*{\hfill}{\qquad}
```

\hrulefill

```
\renewcommand*{\hrulefill}{\rule{1in}{1pt}}
```

\dotfill

```
\renewcommand*{\dotfill}{\dots}
```

```
\end{warpHTML}
```

69 Skips, spaces, font sizes

for HTML output:

```
\begin{warpHTML}
```

\, must be redefined after \RequirePackage{printlen}

```
\let\LWR@origcomma\,
\let\LWR@origtilde~
\let\LWR@origenskip\enskip
\let\LWR@origquad\quad
\let\LWR@origqquad\qquad
\let\LWR@orighspace\hspace
\let\LWR@origrule\rule
\let\LWR@origmedskip\medskip
```

Direct-formatting space commands become HTML entities:

```
\renewcommand*{\,}{\HTMLunicode{202f}} % HTML thin non-breakable space
```

```
\renewcommand*{~}{\HTMLentity{nbsp}}
\renewcommand*{\textellipsis}{\HTMLunicode{2026}}
```

Direct-formatting font sizes are ignored:

```
\let\LWR@orignormalsize\normalsize
\let\LWR@origsmall\small
\let\LWR@origfootnotesize\footnotesize
\let\LWR@origscriptsize\scriptsize
\let\LWR@origtiny\tiny
\let\LWR@origlarge\large
```

```
\let\LWR@origLarge\Large
\let\LWR@origLARGE\LARGE
\let\LWR@orighuge\huge
\let\LWR@origHuge\Huge
\renewcommand*{\normalsize}{}
\renewcommand*{\small}{}
\renewcommand*{\footnotesize}{}
\renewcommand*{\scriptsize}{}
\renewcommand*{\tiny}{}
\renewcommand*{\large}{}
\renewcommand*{\Large}{}
\renewcommand*{\LARGE}{}
\renewcommand*{\huge}{}
\renewcommand*{\Huge}{}

\renewcommand*{\onecolumn}{}

\renewcommand{\twocolumn}[1][]{

#1

}
```

`\newline` Uses HTML `<br />` tag

```
\newcommand*{\LWR@newlinebr}{\unskip\LWR@htmltag{br /}\LWR@orignewline}%
\let\newline\LWR@newlinebr
```

`\\` Redefined to `\LWR@endofline` or `\LWR@tabularendofline`.

`\LWR@endofline` * [⟨*len*⟩]

`\\` is assigned to `\LWR@endofline` at `\LWR@LwarpStart`.

Inside `tabular`, `\\` is temporarily changed to `\LWR@tabularendofline`.

```
\let\LWR@origendofline\\
\NewDocumentCommand{\LWR@endofline}{s o}
{%
\newline%
}
```

`\LWR@minipagestartpars` Minipages are often placed side-by-side inside figures, with a bit of horizontal space to separate them. Since HTML does not allow a `<div>` to be inside a `p`, paragraphs must be turned off during the generation of the minipage, then turned on after the minipage is complete. When this occurs between side-by-side minipages, `lwarp` correctly surpresses the paragraph tags between the minipages, unless some other

\hspace
\enskip
\quad
\qquad

text is between the minipages. Such text forms its own paragraph, resulting in text after a minipage to be on its own line. Since people often place small horizontal space between minipages, it is desirable to maintain this space if possible. lwarp tries to do this by remembering that a minipage has been seen, in which case paragraph tags are surpressed around \hspace, \enskip, \quad, and \qquad until the end of the paragraph, when the closing p tag is created.

When a minipage is seen, the boolean LWR@minipagethispar is set, telling the following horizontal whitespace commands to try to surpress their surrounding paragraph tags. LWR@minipagethispar is cleared at the next end of paragraph, when the HTML paragraph closing tag is generated.

Placed just before \hspace, \quad, or \qquad's HTML output.

```
\newcommand*{\LWR@minipagestartpars}{%
\ifbool{LWR@minipagethispar}%
{%
\LWR@startpars%
}{}%
}
```

\LWR@minipagestoppars

Placed just after \hspace, \quad, or \qquad's HTML output.

```
\newcommand*{\LWR@minipagestoppars}{%
\ifbool{LWR@minipagethispar}%
{%
\LWR@stoppars%
}{}%
}
```

\quad

Handles special minipage & horizontal space interactions.

```
\renewcommand*{\quad}{%
\LWR@minipagestoppars%
\HTMLunicode{2001}%
\LWR@minipagestartpars%
}
```

\qquad

Handles special minipage & horizontal space interactions.

```
\renewcommand*{\qquad}{\quad\quad}
```

\enskip

Handles special minipage & horizontal space interactions.

```
\renewcommand*{\enskip}{%
\LWR@minipagestoppars%
\HTMLunicode{2000}%
```

```
\LWR@minipagestartpars%
}
```

Len \WR@tempwidth
Len \WR@tempheight
Len \WR@tempraise

Used to compute span width, height, raise for \hspace and \rule:

```
\newlength{\LWR@tempwidth}
\newlength{\LWR@tempheight}
\newlength{\LWR@tempraise}
```

\LWR@hspace * {⟨*length*⟩}

Handles special minipage & horizontal space interactions.

Prints a span of a given width. Ignores the optional star.

\hspace{\fill} is converted to \hspace{2em}, equal to \qquad.

```
\NewDocumentCommand{\LWR@hspace}{s m}{%
\setlength{\LWR@tempwidth}{#2}%
```

If \fill, change to \qquad:

```
\ifnum\gluestretchorder\LWR@tempwidth>0%
\setlength{\LWR@tempwidth}{2em}%
\fi%
```

Only if the width is not zero:

```
\ifthenelse{\dimtest{\LWR@tempwidth}{=}{0pt}}{}{%
```

If had a minipage this paragraph, try to inline the white space without generating paragraph tags:

```
\LWR@minipagestoppars%
```

Support the HTML thin wrappable space:

```
\ifthenelse{\dimtest{\LWR@tempwidth}{=}{.16667em}}
{%
\HTMLunicode{2009}% thin breakable space
}%
```

Print the span with the converted width. Not rounded.

```
{%
\uselengthunit{PT}%
\LWR@htmltagc{%
span style="width:\printlength{\LWR@tempwidth}; display:inline-block;"%
```

```
}%
\LWR@htmltagc{/span}%
}%
```

If had a minipage this paragraph, try to inline the white space without generating paragraph tags:

```
\LWR@minipagestartpars%
}%
}
```

`\hspace` * {⟨*length*⟩}

Handles special minipage & horizontal space interactions.

```
\let\hspace\LWR@hspace
```

`\linebreak` [⟨*num*⟩] Inserts an HTML `br` tag.

```
\renewcommand*{\linebreak}[1][]{\newline}
```

`\nolinebreak` [⟨*num*⟩]

```
\renewcommand*{\nolinebreak}[1][]{}
```

`\pagebreak` [⟨*num*⟩] Starts a new paragraph.

```
\renewcommand*{\pagebreak}[1][]{

}
```

`\nopagebreak` [⟨*num*⟩]

```
\renewcommand*{\nopagebreak}[1][]{}
```

`\enlargethispage` * {⟨*len*⟩}

```
\RenewDocumentCommand{\enlargethispage}{s m}{}
```

`\LWR@rule` [⟨*raise*⟩] {⟨*width*⟩} {⟨*height*⟩}

Handles special minipage & horizontal space interactions.

Creates a span of a given width and height. Ignores the optional star.

`\fill` is zero-width, so `\hspace{\fill}` is ignored.

```
\NewDocumentCommand{\LWR@rule}{o m m}{%
```

The width is copied into a temporary LaTeX length, from which comparisons and conversions may be made:

```
\setlength{\LWR@tempwidth}{#2}%
```

If it's zero-width then skip the entire rule:

```
\ifthenelse{\lengthtest{\LWR@tempwidth=0pt}}
{}% zero- width
{% non-zero width
```

If it's non-zero width, set a minimal thickness so that it more reliably shows in the browser:

```
\ifthenelse{\lengthtest{\LWR@tempwidth>0pt}\AND%
\lengthtest{\LWR@tempwidth<1pt}}%
{\setlength{\LWR@tempwidth}{1pt}}{}%
```

Likewise with height:

```
\setlength{\LWR@tempheight}{#3}%
\ifthenelse{\lengthtest{\LWR@tempheight>0pt}\AND%
\lengthtest{\LWR@tempheight<1pt}}%
{\setlength{\LWR@tempheight}{1pt}}{}%
```

If had a minipage this paragraph, try to inline the rule without generating paragraph tags:

```
\LWR@minipagestoppars%
```

Print the span with the converted width and height. The width and height are NOT rounded, since a height of less than 1pt is quite common in LaTeX code.

```
\uselengthunit{PT}%
\LWR@htmltagc{%
span
style=" %
```

The background color is used to draw the filled rule. The color may be changed by `\textcolor`.

```
background:\LWR@currenttextcolor; %
```

The width and height are printed, converted to PT:

```
width:\printlength{\LWR@tempwidth}; %
height:\printlength{\LWR@tempheight}; %
```

The raise height is converted to a CSS transform. The `*2` raise multiplier is to approximately match HTML output's X height. Conversion to a LaTeX length allows a typical LaTeX expression to be used as an argument for the raise, whereas printing the raise argument directly to HTML output without conversion to a LaTeX length limits the allowable syntax. To do: A superior method would compute a ratio of LaTeX ex height, then print that to HTML with an ex unit.

```
\IfValueTF{#1}%
{%
\setlength{\LWR@tempraise}{0pt-#1}%
\setlength{\LWR@tempraise}{\LWR@tempraise*2}%
\LWR@orignewline%
-ms-transform: translate(0pt,\printlength{\LWR@tempraise}); %
\LWR@orignewline%
-webkit-transform: translate(0pt,\printlength{\LWR@tempraise}); %
\LWR@orignewline%
transform: translate(0pt,\printlength{\LWR@tempraise}); %
\LWR@orignewline%
}{}%
```

Display inline-block to place the span inline with the text:

```
display:inline-block;"%
}%
\LWR@htmltagc{/span}%
```

If had a minipage this paragraph, try to inline the white space without generating paragraph tags:

```
\LWR@minipagestartpars%
}% non-zero width
}
```

`\rule` [⟨*raise*⟩] {⟨*width*⟩} {⟨*height*⟩}

Handles special minipage & horizontal space interactions.

```
\let\rule\LWR@rule
```

```
\end{warpHTML}
```

70 \phantomsection

for HTML output:
```
\begin{warpHTML}
```

`\phantomsection` Emulate the hyperref `\phantomsection` command, often used to insert the bibliography into table of contents:

```
\newcommand*{\phantomsection}{\section*{}}
```

```
\end{warpHTML}
```

71 \LaTeX and other logos

Logos for HTML and print modes:

Some of these logos may be redefined in a later package, so after loading other packages, and at the beginning of the document, their definitions are finally `\let` in `\LWR@LwarpStart`.

For CSS conversions, see:
http://edward.oconnor.cx/2007/08/tex-poshlet
http://nitens.org/taraborelli/texlogo

71.1 HTML logos

for HTML output:
```
\begin{warpHTML}
```

`\TeX` TEX

`latexlogo` is a CSS class used to properly typeset the E and A in LATEX and friends.

`latexlogofont` is a CSS class used to select the font for the rest of the logo in LATEX, LuaTEX, ConTEXt, etc.

```
\newcommand*{\LWR@TeX}
{\InlineClass{latexlogofont}%
{\InlineClass{latexlogo}{T\textsubscript{e}X}}}
```

`\LaTeX` LATEX, LATEX 2_ε
`\LaTeXe`

```
\newcommand*{\LWR@LaTeX}
{\InlineClass{latexlogofont}%
```

```
{\InlineClass{latexlogo}%
{L\textsuperscript{a}T\textsubscript{e}X}}}

\renewcommand*{\LaTeXe}
{\LaTeX\InlineClass{latexlogofont}%
{\,2\textsubscript{\textit{\HTMLunicode{3B5}}}}}}
```

\LuaTeX
\LuaLaTeX

LuaTEX, LuaLATEX

```
\newcommand*{\LWR@LuaTeX}{\InlineClass{latexlogofont}{Lua}\TeX}
\newcommand*{\LWR@LuaLaTeX}{\InlineClass{latexlogofont}{Lua}\LaTeX}
```

\XeTeX
\XeLaTeX

XƎTEX, XƎLATEX

`xetexlogo` is a CSS class which aligns the backwards E in XƎTEX and spaces TEX appropriately.

`xelatexlogo` is a CSS class which aligns the backwards E in XƎLATEX and spaces LATEX appropriately.

```
\newcommand*{\Xe}
  {X\textsubscript{\HTMLunicode{18e}}}
\newcommand*{\LWR@XeTeX}{\InlineClass{xetexlogo}{\Xe}\TeX}
\newcommand*{\LWR@XeLaTeX}{\InlineClass{xelatexlogo}{\Xe}\LaTeX}
```

\ConTeXt

ConTEXt

```
\newcommand*{\LWR@ConTeXt}
{\InlineClass{latexlogofont}{Con}\TeX{}%
\InlineClass{latexlogofont}{t}}
```

\BibTeX
\MakeIndex

BIBTEX, *MakeIndex*

```
\providecommand*{\BibTeX}
{\InlineClass{latexlogofont}{B\textsc{ib}}\TeX}

\newcommand*{\MakeIndex}
{\InlineClass{latexlogofont}{\textit{MakeIndex}}}
```

\AmS

$\mathcal{AMS}$

`amslogo` is a CSS class used for the $\mathcal{AMS}$logo.

```
\AtBeginDocument{\DeclareDocumentCommand{\AmS}{}
{\InlineClass{amslogo}{\textit{A\textsubscript{M}S}}}}
```

\MiKTeX MiKTEX

```
\newcommand*{\MiKTeX}{\InlineClass{latexlogofont}{MiK}\TeX}
```

\LyX LyX

lyxlogo is a CSS class used for the LyXlogo.

```
\newcommand*{\LyX}{\InlineClass{lyxlogo}{LyX}}
```

```
\end{warpHTML}
```

71.2 Print logos

for PRINT output:

```
\begin{warpprint}
\newcommand*{\XeTeXrevE}
  {\hspace{-.1667em}\raisebox{-.5ex}{\reflectbox{E}}\hspace{-.125em}}
\providecommand*{\XeTeX}{\mbox{X\XeTeXrevE\TeX}}
\providecommand*{\XeLaTeX}{\mbox{X\XeTeXrevE\LaTeX}}
\providecommand*{\AmS}{%
\leavevmode\hbox{$\mathcal A\kern-.2em\lower.376ex%
\hbox{$\mathcal M$}\kern-.2em\mathcal S$}}
\newcommand*{\LyX}{\textsf{LyX}}
\providecommand*{\LuaTeX}{\mbox{Lua\TeX}}
\providecommand*{\LuaLaTeX}{\mbox{Lua\LaTeX}}
\providecommand*{\BibTeX}{\mbox{B\textsc{ib}\TeX}}
\providecommand*{\MakeIndex}{\mbox{\textit{MakeIndex}}}
\providecommand*{\ConTeXt}{\mbox{Con\TeX{}t}}
\providecommand*{\MiKTeX}{\mbox{MiK\TeX}}
\end{warpprint}
```

72 \AtBeginDocument, \AtEndDocument

for HTML output:

```
\begin{warpHTML}
```

\LWR@LwarpStart
\LWR@LwarpEnd

Automatically sets up the HTML-related actions for the start and end of the document.

```
\AfterEndPreamble{\LWR@LwarpStart}
\AtEndDocument{\LWR@LwarpEnd}
```

```
\end{warpHTML}
```

73 Trademarks

- TEX is a trademark of American Mathematical Society.
- Adobe® and Adobe Framemaker® are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.
- Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Mac OS® is a trademark of Apple Inc.
- MadCap Flare™ is the property of MadCap Software, Inc.
- MathJax is copyright 2009 and later. The MathJax Consortium is a joint venture of the American Mathematical Society (AMS) and the Society for Industrial and Applied Mathematics (SIAM) to advance mathematical and scientific content on the web.
- Microsoft®, Encarta, MSN, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- UNIX® is a registered trademark of The Open Group.

Package 2

lwarp-abstract.sty

74 Abstract

(*Based on original code by* Peter Wilson.)

Pkg `abstract` abstract is supported and patched by lwarp.

abstract is supported. If using the `number` option with file splits, be sure to place the table of contents before the abstract. The number option causes a section break which may cause a file split, which would put a table of contents out of the home page if it is after the abstract.

for HTML output: Accept all options for lwarp-abstract:

```
\LWR@ProvidesPackagePass{abstract}

\AtBeginDocument{
\BeforeBeginEnvironment{abstract}{
\LWR@forcenewpage
\BlockClass{abstract}
}
\AfterEndEnvironment{abstract}{\endBlockClass}
}

\renewcommand{\@bsrunintitle}{%
\hspace*{\abstitleskip}%
{\abstractnamefont%
\InlineClass{abstractrunintitle}{\abstractname}%
\@bslabeldelim}%
}

\if@titlepage
  \renewenvironment{abstract}{%
%     \titlepage
    \null\vfil
    \@beginparpenalty\@lowpenalty
    \if@bsrunin
    \else
      \if@bsstyle
        \abstitlestyle{\BlockClassSingle{abstracttitle}{\abstractname}}
      \else
        \ifnumber@bs
          \num@bs
```

```
        \else
          \begin{\absnamepos}%
  \abstractnamefont \BlockClassSingle{abstracttitle}{\abstractname}
            \@endparpenalty\@M
          \end\absnamepos%
%%        \vspace{\abstitleskip}%
        \fi
      \fi
      \vspace{\abstitleskip}%
    \fi
    \put@bsintoc%
    \begin{@bstr@ctlist}\if@bsrunin\@bsrunintitle\fi\abstracttextfont}%
    {\par\end{@bstr@ctlist}\vfil\null%\endtitlepage
}
\else
  \renewenvironment{abstract}{%
    \if@bsrunin
    \else
      \if@bsstyle
        \abstitlestyle{\BlockClassSingle{abstracttitle}{\abstractname}}
      \else
        \ifnumber@bs
          \num@bs
        \else
\begin{\absnamepos}%
\abstractnamefont\BlockClassSingle{abstracttitle}{\abstractname}%
\end\absnamepos%
%%        \vspace{\abstitleskip}%
        \fi
      \fi
      \vspace{\abstitleskip}%
    \fi
    \put@bsintoc%
    \begin{@bstr@ctlist}\if@bsrunin\@bsrunintitle\fi\abstracttextfont}%
    {\par\end{@bstr@ctlist}}
\fi

```

Package 3

lwarp-afterpage.sty

75 Afterpage

Pkg `afterpage` Not used.

for HTML output: Discard all options for lwarp-afterpage:

```
\LWR@ProvidesPackageDrop{afterpage}
```

```
\newcommand{\afterpage}[1]{#1}
```

Package 4

lwarp-algorithmicx.sty

76 Algorithmicx

Pkg `algorithmicx` algorithmicx is supported with minor adjustments.

for HTML output:

```
\LWR@ProvidesPackagePass{algorithmicx}
```

Inside the `algorithmic` environment, level indenting is converted to a `<span>` of the required length, and comments are placed inside a `<span>` which is floated right.

⚠ package conflicts If using `\newfloat`, trivfloat, and/or algorithmicx together, see section 159.1.

for HTML output:

```
\begin{warpHTML}
```

```
\AtBeginEnvironment{algorithmic}{%
%
\let\origALG@doentity\ALG@doentity%
%
\renewcommand*{\ALG@doentity}{%
\origALG@doentity%
\uselengthunit{PT}%
\LWR@htmltagc{%
span style="width:\rndprintlength{\ALG@thistlm}; display:inline-block;"%
}%
\LWR@htmltagc{/span}%
}%
%
\let\origComment\Comment%
%
\renewcommand{\Comment}[1]{\InlineClass{floatright}{\origComment{#1}}}%
}
```

```
\end{warpHTML}
```

Package 5

lwarp-alltt.sty

77 Alltt

Pkg `alltt` alltt is patched for use by lwarp.

for HTML output:

```
\LWR@ProvidesPackagePass{alltt}

\AfterEndPreamble{
\AtBeginEnvironment{alltt}{%
\LWR@forcenewpage
\LWR@atbeginverbatim{alltt}\unskip\vspace*{-\baselineskip}%
}
\AfterEndEnvironment{alltt}{\unskip\vspace*{-\baselineskip}\LWR@afterendverbatim}
}
```

Package 6

lwarp-amsthm.sty

78 AMSthm

(*Based on original code by* Publications Technical Group — American Mathematical Society.)

Pkg `amsthm` amsthm is patched for use by lwarp.

CSS styling of theorems and proofs:

Theorem: `<div>` of class `amsthmbody<theoremstyle>`

Theorem Name: `<span>` of class `amsthmname<theoremtyle>`

Theorem Number: `<span>` of class `amsthmnumber<theoremstyle>`

Theorem Note: `<span>` of class `amsthmnote<theoremstyle>`

Proof: `<div>` of class `amsthmproof`

Proof Name: `<span>` of class `amsthmproofname`

where `<theoremstyle>` is `plain`, `definition`, etc.

for HTML output:

```
\LWR@ProvidesPackagePass{amsthm}
```

Storage for the style being used for new theorems:

```
\newcommand{\LWR@newtheoremstyle}{plain}
```

Patched to remember the style being used for new theorems:

```
\renewcommand{\theoremstyle}[1]{%
  \@ifundefined{th@#1}{%
    \PackageWarning{amsthm}{Unknown theoremstyle `#1'}%
    \thm@style{plain}%
\renewcommand{\LWR@newtheoremstyle}{plain}% new
  }{%
    \thm@style{#1}%
\renewcommand{\LWR@newtheoremstyle}{#1}% new
  }%
}
```

Patched to remember the style for this theorem type:

```
\def\@xnthm#1#2{%
  \csedef{LWR@thmstyle#2}{\LWR@newtheoremstyle}% new
  \let\@tempa\relax
  \@xp\@ifdefinable\csname #2\endcsname{%
    \global\@xp\let\csname end#2\endcsname\@endtheorem
    \ifx *#1% unnumbered, need to get one more mandatory arg
      \edef\@tempa##1{%
        \gdef\@xp\@nx\csname#2\endcsname{%
          \@nx\@thm{\@xp\@nx\csname th@\the\thm@style\endcsname}%
            {}{##1}}}%
    \else % numbered theorem, need to check for optional arg
      \def\@tempa{\@oparg{\@ynthm{#2}}[]}%
    \fi
\AtBeginEnvironment{#2}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#2}}}% new
  }%
  \@tempa
}
```

Patched to enclose with CSS:

```
\newcommand{\LWR@haveamsthmname}{
\renewcommand{\thmname}[1]{\InlineClass{amsthmname\LWR@thisthmstyle}{##1}}
}

\newcommand{\LWR@haveamsthmnumber}{
\renewcommand{\thmnumber}[1]{\InlineClass{amsthmnumber\LWR@thisthmstyle}{##1}}
}

\newcommand{\LWR@haveamsthmnote}{
\renewcommand{\thmnote}[1]{\InlineClass{amsthmnote\LWR@thisthmstyle}{##1}}
}

\LWR@haveamsthmname
\LWR@haveamsthmnumber
\LWR@haveamsthmnote
```

Patches for CSS:

```
\def\@begintheorem#1#2[#3]{%
\LWR@forcenewpage% new
  \BlockClass{amsthmbody\LWR@thisthmstyle}% new
  \deferred@thm@head{
\the\thm@headfont \thm@indent
    \@ifempty{#1}{\let\thmname\@gobble}{\LWR@haveamsthmname}% new
    \@ifempty{#2}{\let\thmnumber\@gobble}{\LWR@haveamsthmnumber}% new
    \@ifempty{#3}{\let\thmnote\@gobble}{\LWR@haveamsthmnote}% new
    \thm@swap\swappedhead\thmhead{#1}{#2}{#3}%
    \the\thm@headpunct~
    \thmheadnl % possibly a newline.
    \hskip\thm@headsep
```

```
  }%
  \ignorespaces}
```

Patched for CSS:

```
\def\@endtheorem{\endBlockClass\endtrivlist\@endpefalse }
```

Proof QED symbol:

```
\AtBeginDocument{
\def\openbox{\text{\HTMLunicode{25A1}}}% UTF-8 white box
\def\blacksquare{\text{\HTMLunicode{220E}}}% UTF-8 end-of-proof
\def\Box{\text{\HTMLunicode{25A1}}}% UTF-8 white box
}
```

Patched for CSS:

```
\renewenvironment{proof}[1][\proofname]{\par
\LWR@forcenewpage% new
\BlockClass{amsthmproof}% new
  \pushQED{\qed}%
  \normalfont \topsep6\p@\@plus6\p@\relax
  \trivlist
  \item[\hskip\labelsep
        \InlineClass{amsthmproofname}{#1\@addpunct{.}}]\ignorespaces% changes
}{%
  \InlineClass{theoremendmark}{\popQED}\endtrivlist%
  \endBlockClass% new
  \@endpefalse
}
```

Package 7

lwarp-bookmark.sty

79 Bookmark

Pkg `bookmark` bookmark is emulated during HTML output, and the bookmark package is ignored.

for HTML output: Discard all options for lwarp-bookmark:

```
\LWR@ProvidesPackageDrop{bookmark}
```

```
\newcommand*{\bookmarksetup}[1]{}
```

```
\newcommand*{\bookmarksetupnext}[1]{}
\newcommand*{\bookmark}[2][]{}
\newcommand*{\bookmarkdefinestyle}[2]{}
\newcommand*{\bookmarkget}[1]{}
\newcommand{\BookmarkAtEnd}[1]{}
```

Package 8

lwarp-booktabs.sty

80 Booktabs

Pkg `booktabs` booktabs is emulated during HTML output, and the booktabs package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{booktabs}
```

Booktabs emulation is spread among the tabular code.

Emulated for source compatibility.

```
\newcommand*{\addlinespace}[1]{}
\newcommand*{\morecmidrules}{}
\newcommand*{\specialrule}[3]{}
```

Package 9

lwarp-ccaption.sty

81 Ccaption

Pkg `ccaption` ccaption is not used. The user is recommended to use caption instead.

for HTML output:

```
\LWR@loadnever{ccaption}{caption}
```

Package 10

lwarp-changepage.sty

82 Changepage

Pkg changepage changepage is ignored.

for HTML output: Discard all options for lwarp-changepage:

```
\LWR@ProvidesPackageDrop{changepage}
```

```
\newif\ifoddpage
\DeclareRobustCommand{\checkoddpage}{\oddpagetrue}
\DeclareRobustCommand{\changetext}[5]{}
\DeclareRobustCommand{\changepage}[9]{}
\newenvironment{adjustwidth}[2]{}{}
\newenvironment{adjustwidth*}[2]{}{}
```

Package 11

lwarp-cutwin.sty

83 Cutwin

Pkg `cutwin` Emulated.

for HTML output: Discard all options for lwarp-cutwin:

```
\LWR@ProvidesPackageDrop{cutwin}
```

```
\newcommand*{\opencutleft}{}
\newcommand*{\opencutright}{}
\newcommand*{\opencutcenter}{}
\newcommand*{\cutfuzz}{}

\newenvironment{cutout}[4]
{\marginpar{\windowpagestuff}}
{}

\newcommand*{\windowpagestuff}{}

\newcommand*{\pageinwindow}{%
% \begin{minipage}{.3\linewidth}
\windowpagestuff
% \end{minipage}
}

\newenvironment{shapedcutout}[3]
{\marginpar{\picinwindow}}
{}

\newcommand*{\putstuffinpic}{}

\newcommand*{\picinwindow}{%
\begin{picture}(0,0)
\putstuffinpic
\end{picture}}
```

Package 12

lwarp-dcolumn.sty

84 Dcolumn

Pkg `dcolumn` dcolumn is emulated during HTML output, and the dcolumn package is ignored.

```
\LWR@ProvidesPackageDrop{dcolumn}
```

Package 13

lwarp-draftwatermark.sty

85 Draftwatermark

Pkg `draftwatermark` draftwatermark is emulated during HTML output, and the draftwatermark package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{draftwatermark}
```

```
\newcommand{\SetWatermarkAngle}[1]{}
\newcommand{\SetWatermarkColor}[1]{}
\newcommand{\SetWatermarkLightness}[1]{}
\newcommand{\SetWatermarkFontSize}[1]{}
\newcommand{\SetWatermarkScale}[1]{}
\newcommand{\SetWatermarkHorCenter}[1]{}
\newcommand{\SetWatermarkVertCenter}[1]{}
\newcommand{\SetWatermarkText}[1]{}
```

Package 14

lwarp-ellipsis.sty

86 Ellipsis

Pkg `ellipsis` ellipsis is emulated during HTML output, and the ellipsis package is ignored.

```
\LWR@ProvidesPackageDrop{ellipsis}

\newcommand{\ellipsisgap}{0.1em}
```

Package 15

lwarp-emptypage.sty

87 Emptypage

Pkg `emptypage` emptypage is ignored.

for HTML output: Discard all options for lwarp-emptypage:

```
\LWR@ProvidesPackageDrop{emptypage}
```

Package 16

lwarp-endnotes.sty

88 Endnotes

(*Based on original code by* John Lavagnino.)

Pkg `endnotes`

Discard all options for lwarp-endnotes:

for HTML output:

```
\LWR@ProvidesPackagePass{endnotes}

\def\enoteformat{%
% \rightskip\z@ \leftskip\z@ \parindent=1.8em
\leavevmode
% \llap{
\makeenmark
% }
}

\def\@makeenmark{\hbox{\textsuperscript{\normalfont\theenmark}}}
\def\makeenmark{\@makeenmark}
```

Package 17

lwarp-enumerate.sty

89 Enumerate

Pkg `enumerate` enumerate is ignored. enumitem is then modified per the `shortlabels` option.

enumerate conflicts with enumitem if both are loaded at the same time, but lwarp does not actually load enumerate. While generating HTML, lwarp only loads enumitem, and enumerate is simulated by enumitem using the functionality of the `shortlabels` option.

A problem may occur during print output if enumitem is loaded, either manually or by some other package such as siunitx. If these are used, enumerate will conflict with enumitem during print output.

for HTML output: Discard all options for lwarp-enumerate:

```
\LWR@ProvidesPackageDrop{enumerate}
```

```
% \DeclareOption{shortlabels}
% {
\def\enit@shl#1{%
     \ifnum\enit@type=\tw@
       \enit@toks{#1}%
     \else
       \def\enit@c{#1}%
       \enit@first#1,\@nil\@@nil % Returns enit@toks
    \fi}
% }
```

Package 18

lwarp-epigraph.sty

90 Epigraph

Pkg epigraph epigraph is emulated during HTML output, and the epigraph package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{epigraph}

\newcommand{\qitem}[2]
{
\begin{BlockClass}{qitem}
#1
\begin{BlockClass}{epigraphsource}
#2
\end{BlockClass}
\end{BlockClass}
}

\newcommand{\epigraph}[2]
{
\begin{BlockClass}{epigraph}
\qitem{#1}{#2}
\end{BlockClass}
}

\newenvironment*{epigraphs}
{\BlockClass{epigraph}}
{\endBlockClass}
```

Use CSS to format epigraphs.

The following are null commands for source compatibility:

```
\newlength{\epigraphwidth}
\setlength{\epigraphwidth}{.5\linewidth}
\newenvironment*{flushepinormal}{}{}
\newcommand{\textflush}[1]{flushepinormal}
\newcommand{\epigraphflush}[1]{flushright}
\newcommand{\sourceflush}[1]{flushright}
\newcommand*{\epigraphsize}{\small}
\newlength{\epigraphrule}
\newlength{\beforeepigraphskip}
\newlength{\afterepigraphskip}
\newcommand{\epigraphhead}[2][0]{#2}
```

```
\newcommand{\dropchapter}[1]{}
\newcommand*{\undodrop}{}
\newcommand{\cleartoevenpage}[1][]{}
```

Package 19

lwarp-eso-pic.sty

91 Eso-pic

Pkg `eso-pic` eso-pic is emulated during HTML output, and the eso-pic package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{eso-pic}
```

```
\newcommand*{\LenToUnit}{}
\newcommand{\AtPageUpperLeft}[1]{}
\newcommand{\AtPageLowerLeft}[1]{}
\newcommand{\AtPageCenter}[1]{}
\newcommand{\AtStockLowerLeft}[1]{}
\newcommand{\AtStockUpperLeft}[1]{}
\newcommand{\AtStockCenter}[1]{}
\newcommand{\AtTextUpperLeft}[1]{}
\newcommand{\AtTextLowerLeft}[1]{}
\newcommand{\AtTextCenter}[1]{}
\NewDocumentCommand{\AddToShipoutPictureBG}{s +m}{}
\let\AddToShipoutPicture\AddToShipoutPictureBG
\NewDocumentCommand{\AddToShipoutPictureFG}{s +m}{}
\newcommand*{\ClearShipoutPictureBG}{}
\newcommand*{\ClearShipoutPicture}{}
\newcommand*{\ClearShipoutPictureFG}{}
\newcommand{\gridSetup}[6][]{}
```

Package 20

lwarp-everypage.sty

92 Everypage

Pkg `everypage` everypage is emulated during HTML output, and the everypage package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{everypage}
```

```
\newcommand*{\AddEverypageHook}[1]{}
\newcommand*{\AddThispageHook}[1]{}
```

Package 21

lwarp-extramarks.sty

93 Extramarks

Pkg `extramarks` extramarks is not used.

for HTML output: Discard all options for lwarp-extramarks:

```
\LWR@ProvidesPackageDrop{extramarks}
```

```
\newcommand*{\extramarks}[2]{}
\newcommand*{\firstleftxmark}{}
\newcommand*{\lastleftxmark}{}
\newcommand*{\firstrightxmark}{}
\newcommand*{\lastrightxmark}{}
\newcommand*{\firstxmark}{}
\newcommand*{\lastxmark}{}
\newcommand*{\topxmark}{}
\newcommand*{\topleftxmark}{}
\newcommand*{\firstleftmark}{}
\newcommand*{\lastrightmark}{}
```

Package 22

lwarp-fancyhdr.sty

94 Fancyhdr

Pkg `fancyhdr` fancyhdr is nullified.

for HTML output: Discard all options for lwarp-fancyhdr:

```
\LWR@ProvidesPackageDrop{fancyhdr}
```

```
\newcommand*{\fancyhead}[2][]{}
\newcommand*{\fancyfoot}[2][]{}
\newcommand*{\fancyhf}[2][]{}
\newcommand*{\fancypagestyle}[2]{}
\newcommand*{\lhead}[1]{}
\newcommand*{\chead}[1]{}
\newcommand*{\rhead}[1]{}
\newcommand*{\lfoot}[1]{}
\newcommand*{\cfoot}[1]{}
\newcommand*{\rfoot}[1]{}
\newcommand*{\headrulewidth}{}
\newcommand*{\footrulewidth}{}
\newcommand*{\fancyheadoffset}[2][]{}
\newcommand*{\fancyfootoffset}[2][]{}
\newcommand*{\fancyhfoffset}[2][]{}
\newcommand*{\iffloatpage}[2]{#2}
\newcommand*{\ifftopfloat}[2]{#2}
\newcommand*{\iffbotfloat}[2]{#2}
```

Package 23

lwarp-float.sty

95 Float and \newfloat

Pkg `float` float is emulated during HTML output, and the float package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{float}[2016/03/04]
```

See section 58.2 for the `\listof` command.

`\newfloat` {⟨*1: type*⟩} {⟨*2: placement*⟩} {⟨*3: ext*⟩} [⟨*4: within*⟩]

Emulates the `\newfloat` command from the float package.

"placement" is ignored.

```
\NewDocumentCommand{\newfloat}{m m m o}{%
\IfValueTF{#4}
{
\DeclareFloatingEnvironment[fileext=#3,within=#4]{#1}
}
{\DeclareFloatingEnvironment[fileext=#3]{#1}}
```

newfloat package automatically creates the `\listof` command for new floats, but float does not, so remove `\listof` here in case it is manually created later.

```
\cslet{listof#1s}\relax
\cslet{listof#1es}\relax
}
```

`\floatname` {⟨*type*⟩} {⟨*name*⟩}

Sets the text name of the float, such as "Figure".

```
\NewDocumentCommand{\floatname}{m +m}{%
\SetupFloatingEnvironment{#1}{name=#2}%
}
```

`\floatplacement` {⟨*type*⟩} {⟨*placement*⟩}

Float placement is ignored.

```
\newcommand*{\floatplacement}[2]{%
\SetupFloatingEnvironment{#1}{placement=#2}%
}
```

`\floatstyle` {⟨*style*⟩}

Float styles are ignored.

```
\newcommand{\floatstyle}[1]{%
}
```

`\restylefloat` * {⟨*style*⟩}

Float styles are ignored.

```
\NewDocumentCommand{\restylefloat}{s m}{%
}
```

Package 24

lwarp-floatflt.sty

96 Floatflt

Pkg `floatflt` Emulated.

for HTML output: Discard all options for lwarp-floatflt:

```
\LWR@ProvidesPackageDrop{floatflt}
```

Borrowed from the lwarp version of keyfloat:

```
\NewDocumentEnvironment{KFLTfloatflt@marginfloat}{O{-1.2ex} m}
{% start
\LWR@maybeincthisfloat%
\LWR@forcenewpage
\LWR@stoppars%
\LWR@htmltag{div class="marginblock" id="autofloat-\arabic{LWR@thisfloat}"}
\LWR@startpars%
\captionsetup{type=#2}%
}
{
\LWR@htmldivclassend{div}
}

\DeclareDocumentEnvironment{floatingfigure}{o m}
  {\begin{KFLTfloatflt@marginfloat}{figure}}
  {\end{KFLTfloatflt@marginfloat}}

\DeclareDocumentEnvironment{floatingtable}{o +m}
  {\begin{KFLTfloatflt@marginfloat}{table}#2}
  {\end{KFLTfloatflt@marginfloat}}
```

Package 25

lwarp-floatrow.sty

97 Floatrow

Pkg `floatrow` floatrow is emulated during HTML output, and the floatrow package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{floatrow}
```

⚠ subfig package

When combined with the subfig package, while inside a `subfloatrow` `\ffigbox` and `\ttabbox` must have the caption in the first of the two of the mandatory arguments.

⚠ `\FBwidth`, `\FBheight`

The emulation of floatrow does not support `\FBwidth` or `\FBheight`. These values are pre-set to `.3\linewidth` and `2in`. Possible solutions include:

- Use fixed lengths. lwarp will scale the HTML lengths appropriately.
- Use `warpprint` and `warpHTML` environments to select appropriate values for each case.
- Inside a `warpHTML` environment, manually change `\FBwidth` or `\FBheight` before the `\ffigbox` or `\ttabbox`. Use `\FBwidth` or `\FBheight` normally afterwards; it will be used as expected in print output, and will use your custom-selected value in HTML output. This custom value will be used repeatedly, until it is manually changed to a new value.

After everything has loaded, remember whether subcaption was loaded. If not, it is assumed that subfig is used instead:

```
\newbool{LWR@subcaptionloaded}

\AtBeginDocument{
\@ifpackageloaded{subcaption}
{\booltrue{LWR@subcaptionloaded}}
{\boolfalse{LWR@subcaptionloaded}}
}
```

`\floatbox` [⟨*1 preamble*⟩] {⟨*2 captype*⟩} [⟨*3 width*⟩] [⟨*4 height*⟩] [⟨*5 vert pos*⟩] {⟨*6 caption*⟩} {⟨*7 object*⟩}

Only parameters for captype, width, caption, and object are used.

`LWR@insubfloatrow` is true if inside a `subfloatrow` environment.

There are two actions, depending on the use of subcaption or subfig.

```
\NewDocumentCommand{\floatbox}{o m o o o +m +m}{%
\ifbool{LWR@subcaptionloaded}%
{% subcaption
```

For subcaption:

```
\ifbool{LWR@insubfloatrow}%
{% subcaption in a subfloatrow
```

`subfigure` and `subtable` environments take width as an argument.

```
\IfValueTF{#3}%
{\@nameuse{sub#2}{#3}}%
{\@nameuse{sub#2}{\linewidth}}%
}% subcaption in a subfloatrow
{% subcaption not in subfloatrow
```

`figure` and `table` environments do not take a width argument.

```
\@nameuse{#2}%
}% subcaption not in subfloatrow
#6

#7
```

End the environments:

```
\ifbool{LWR@insubfloatrow}%
{\@nameuse{endsub#2}}%
{\@nameuse{end#2}}%
}% subcaption
{% assume subfig
```

For subfig:

```
\ifbool{LWR@insubfloatrow}%
{% subfig in a subfloatrow
```

`\subfloat` is a macro, not an environment.

Package subfig's `\subfloat` command takes an optional argument which is the caption, but `\floatbox` argument #6 contains commands to create the caption and label, not the caption itself. Thus, `\caption` is temporarily disabled to return its own argument without braces.

```
\begingroup
\let\caption\@firstofone
```

```
\subfloat[#6]{#7}
\endgroup
}% subfig in a subfloatrow
{% subfig package, but not a subfig
```

`figure` and `table` are environments:

```
\@nameuse{#2}
#6

#7
\@nameuse{end#2}
}% subfig package, but not a subfig
}% assume subfig
}
```

Not used:

```
\newcommand*{\nocapbeside}{}
\newcommand*{\capbeside}{}
\newcommand*{\captop}{}
\newlength{\FBwidth}
\setlength{\FBwidth}{.3\linewidth}
\newlength{\FBheight}
\setlength{\FBheight}{2in}
\newcommand*{\useFCwidth}{}
\newcommand{\floatsetup}[2][]{}
\newcommand{\thisfloatsetup}[1]{}
\newcommand{\clearfloatsetup}[1]{}
\newcommand*{\killfloatstyle}{}
```

Preamble and default width are ignored.

```
\NewDocumentCommand{\newfloatcommand}{m m o o}{%
\@namedef{#1}{
\floatbox{#2}
}
}
```

Preamble and default width are ignored.

```
\NewDocumentCommand{\renewfloatcommand}{m m o o}{%
\@namedef{#1}{%
\floatbox{#2}
}
}
```

```
\newfloatcommand{ffigbox}{figure}[\nocapbeside][]
```

```
\newfloatcommand{ttabbox}{table}[\captop][\FBwidth]
```

```
\newfloatcommand{fcapside}{figure}[\capbeside][]
```

The row of floats is placed into a `<div>` of class `floatrow`.

```
\newenvironment*{floatrow}[1][2]
{
\LWR@forcenewpage
\BlockClass{floatrow}
```

While inside the floatrow, divide the `\linewidth` by the number of floats.

```
\booltrue{LWR@infloatrow}
\setlength{\linewidth}{6in/#1}
}
{
\boolfalse{LWR@infloatrow}
\endBlockClass
}
```

Keys for `\DeclareNewFloatType`:

```
\newcommand*{\LWR@frowkeyplacement}{}
\newcommand*{\LWR@frowkeyname}{}
\newcommand*{\LWR@frowkeyfileext}{}
\newcommand*{\LWR@frowkeywithin}{}
\newcommand*{\LWR@frowkeycapstyle}{}

\define@key{frowkeys}{placement}{}%
\define@key{frowkeys}{name}{\renewcommand{\LWR@frowkeyname}{#1}}%
\define@key{frowkeys}{fileext}{\renewcommand{\LWR@frowkeyfileext}{#1}}%
\define@key{frowkeys}{within}{\renewcommand{\LWR@frowkeywithin}{#1}}%
\define@key{frowkeys}{relatedcapstyle}{}%
```

Use `\listof{type}{Title}` to print a list of the floats.

```
\newcommand*{\DeclareNewFloatType}[2]{%
```

Reset key values:

```
\renewcommand*{\LWR@frowkeyplacement}{}
\renewcommand*{\LWR@frowkeyname}{}
\renewcommand*{\LWR@frowkeyfileext}{}
\renewcommand*{\LWR@frowkeywithin}{}
\renewcommand*{\LWR@frowkeycapstyle}{}
```

Read new key values:

```
\LWR@traceinfo{about to setkeys frowkeys}%
\setkeys{frowkeys}{#2}%
\LWR@traceinfo{finished setkeys frowkeys}%
```

Create a new float with optional [within]:

```
\ifthenelse{\equal{\LWR@frowkeywithin}{}}
{
\LWR@traceinfo{about to newfloat #1 \LWR@frowkeyplacement\
\LWR@frowkeyfileext}%
\newfloat{#1}{\LWR@frowkeyplacement}{\LWR@frowkeyfileext}
}%
{%
\LWR@traceinfo{about to newfloat #1\ \LWR@frowkeyplacement\
\LWR@frowkeyfileext\ \LWR@frowkeywithin}%
\newfloat{#1}{\LWR@frowkeyplacement}%
{\LWR@frowkeyfileext}[\LWR@frowkeywithin]%
\LWR@traceinfo{finished newfloat #1}
}%
```

Rename the float if a name was given:

```
\ifthenelse{\equal{\LWR@frowkeyname}{}}
{}
{\floatname{#1}{\LWR@frowkeyname}}%
}
```

Not used:

```
\newcommand{\buildFBBOX}[2]{}
\newcommand*{\CenterFloatBoxes}{}
\newcommand*{\TopFloatBoxes}{}
\newcommand*{\BottomFloatBoxes}{}
\newcommand*{\PlainFloatBoxes}{}

\newcommand{\capsubrowsettings}{}

\NewDocumentCommand{\RawFloats}{o o}{}
```

To be used inside a minipage or parbox.

```
\newcommand{\RawCaption}[1]{#1}
```

Places additional text inside a float, inside a CSS `<div>` of class `floatfoot`.

```
\NewDocumentCommand{\floatfoot}{s +m}{%
\begin{BlockClass}{floatfoot}
#2
\end{BlockClass}
```

```
}
```

Used to compute \linewidth.

```
\newbool{LWR@insubfloatrow}
\boolfalse{LWR@insubfloatrow}
```

```
\newenvironment*{subfloatrow}[1][2]
{
```

The row of floats is placed into a of class floatrow:

```
\LWR@forcenewpage
\BlockClass{floatrow}
```

While inside the floatrow, LWR@insubfloatrow is set true, which tells \floatbox to use \subfigure or \subtable.

```
\begingroup
\booltrue{LWR@insubfloatrow}
}
{
\endgroup
\endBlockClass
\boolfalse{LWR@insubfloatrow}
}
```

Package 26

lwarp-fontenc.sty

98 Fontenc

Pkg `fontenc` Error if fontenc is loaded after lwarp.

Discard all options for lwarp-fontenc:

for HTML output:

```
\LWR@ProvidesPackageDrop{fontenc}
\LWR@loadbefore{fontenc}
```

Package 27

lwarp-fontspec.sty

99 Fontspec

Pkg `fontspec` Error if fontspec is loaded after lwarp.

Discard all options for lwarp-fontspec:

for HTML output:

```
\LWR@ProvidesPackageDrop{fontspec}
\LWR@loadbefore{fontspec}
```

Package 28

lwarp-footmisc.sty

100 Footmisc

(*Based on original code by* Robin Fairbairns.)

Pkg `footmisc` footmisc is emulated during HTML output, and the footmisc package is ignored.

```
\LWR@ProvidesPackageDrop{footmisc}
```

Some nullified commands:

```
\newcommand{\footnotelayout}{}
\newcommand{\setfnsymbol}[1]{}
\NewDocumentCommand{\DefineFNsymbols}{s m o m}{}

\newdimen\footnotemargin
\footnotemargin1.8em\relax

\newcommand*\hangfootparskip{0.5\baselineskip}
\newcommand*\hangfootparindent{0em}%

\let\pagefootnoterule\footnoterule
\let\mpfootnoterule\footnoterule
\def\splitfootnoterule{\kern-3\p@ \hrule \kern2.6\p@}

\providecommand*{\multiplefootnotemarker}{3sp}
\providecommand*{\multfootsep}{,}
```

Using cleveref:

```
\providecommand*{\footref}[1]{\labelcref{#1}}
```

The following work as-is:

```
\newcommand\mpfootnotemark{%
  \@ifnextchar[%
    \@xmpfootnotemark
    {%
      \stepcounter\@mpfn
      \protected@xdef\@thefnmark{\thempfn}%
      \@footnotemark
    }%
}
\def\@xmpfootnotemark[#1]{%
  \begingroup
    \csname c@\@mpfn\endcsname #1\relax
    \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotemark
}
```

Package 29

lwarp-footnote.sty

101 Footnote

Pkg `footnote` footnote is used with minor patches.

for HTML output:

```
\LWR@ProvidesPackagePass{footnote}
```

Removed print-version formatting:

```
\def\fn@startnote{%
%   \@parboxrestore%
  \protected@edef\@currentlabel{\csname p@\@mpfn\endcsname\@thefnmark}%
%   \color@begingroup% *** conflicts with lwarp
}

% \let\fn@endnote\color@endgroup% *** conflicts with lwarp
\newcommand*{\fn@endnote}{\LWR@htmltagc{/\LWR@tagregularparagraph}}
```

Removed print-version formatting:

```
\def\fn@startfntext{%
  \setbox\z@\vbox\bgroup%
    \fn@startnote%
    \fn@prefntext%
    \ignorespaces%
}
```

Removed print-version formatting, added closing paragraph tag:

```
\def\fn@endfntext{%
\LWR@htmltagc{/\LWR@tagregularparagraph}%
    \fn@postfntext%
  \egroup%
  \begingroup%
    \let\@makefntext\@empty%
    \let\@finalstrut\@gobble%
    \let\rule\@gobbletwo%
    \@footnotetext{\unvbox\z@}%
  \endgroup%
}
```

These have been redefined, so re-`\let` them again:

```
\let\endfootnote\fn@endfntext
\let\endfootnotetext\endfootnote
```

Package 30

lwarp-footnotehyper.sty

102 Footnotehyper

Pkg `footnotehyper` footnotehyper is a hyperref-safe version of footnote. For lwarp, footnotehyper is emulated.

for HTML output: Discard all options for lwarp-footnotehyper:

```
\RequirePackage{footnote}
\LWR@ProvidesPackageDrop{footnotehyper}
```

Package 31

lwarp-framed.sty

103 Framed

(*Based on original code by* Donald Arseneau.)

Pkg `framed` framed is supported and patched by lwarp.

for HTML output: Accept all options for lwarp-framed:

```
\LWR@ProvidesPackagePass{framed}

\renewenvironment{framed}{%
\LWR@forcenewpage
\BlockClass{framed}%
}
{\endBlockClass}

\renewenvironment{oframed}{%
\LWR@forcenewpage
\BlockClass{framed}%
}
{\endBlockClass}


\renewenvironment{shaded}{%
\convertcolorspec{named}{shadecolor}{HTML}\LWR@tempcolor%
\LWR@forcenewpage
\BlockClass{framed}[background: \#\LWR@tempcolor]%
}
{\endBlockClass}

\renewenvironment{shaded*}{%
\convertcolorspec{named}{shadecolor}{HTML}\LWR@tempcolor%
\LWR@forcenewpage
\BlockClass{framed}[background: \#\LWR@tempcolor]%
}
{\endBlockClass}


\renewenvironment{leftbar}{%
\LWR@forcenewpage
\BlockClass{framedleftbar}
  \def\FrameCommand{}%
```

```
  \MakeFramed {}
}%
 {\endMakeFramed\endBlockClass}


\renewenvironment{snugshade}{%
\convertcolorspec{named}{shadecolor}{HTML}\LWR@tempcolor%
\LWR@forcenewpage
\BlockClass{snugframed}[background: \#\LWR@tempcolor]%
}
{\endBlockClass}

\renewenvironment{snugshade*}{%
\convertcolorspec{named}{shadecolor}{HTML}\LWR@tempcolor%
\LWR@forcenewpage
\BlockClass{snugframed}[background: \#\LWR@tempcolor]%
}
{\endBlockClass}

\let\oframed\framed
\let\endoframed\endframed


\RenewEnviron{titled-frame}[1]{%
\CustomFBox{#1}{}{0pt}{0pt}{0pt}{0pt}{\BODY}
}
```

\CustomFBox {⟨*toptitle*⟩} {⟨*bottitle*⟩} {⟨*thicknesstop*⟩} {⟨*bottom*⟩} {⟨*left*⟩} {⟨*right*⟩} {⟨*text contents*⟩}

```
\renewcommand{\CustomFBox}[7]{%
\convertcolorspec{named}{TFFrameColor}{HTML}\LWR@tempcolor%
\LWR@forcenewpage
\begin{BlockClass}{framed}[border: 3px solid \#\LWR@tempcolor]%
\ifthenelse{\isempty{#1}}{}{% not empty
\begin{BlockClass}{framedtitle}[background: \#\LWR@tempcolor]%
\textcolor{TFTitleColor}{\textbf{#1}}%
\end{BlockClass}
}% not empty

#7

\ifthenelse{\isempty{#2}}{}{% not empty
\convertcolorspec{named}{TFFrameColor}{HTML}\LWR@tempcolor%
\begin{BlockClass}{framedtitle}[background: \#\LWR@tempcolor]%
\textcolor{TFTitleColor}{\textbf{#2}}%
\end{BlockClass}
}% not empty
\end{BlockClass}
```

```
}
```

\TitleBarFrame [⟨*marker*⟩] {⟨*title*⟩} {⟨*contents*⟩}

```
\renewcommand\TitleBarFrame[3][]{
\CustomFBox
    {#2}{}%
    \fboxrule\fboxrule\fboxrule\fboxrule
    {#3}%
}
```

```
\renewcommand{\TF@Title}[1]{#1}
```

MakeFramed {⟨*settings*⟩}

```
\let\MakeFramed\relax
\let\endMakeFramed\relax

\NewEnviron{MakeFramed}[1]{%
\FrameCommand{\begin{minipage}{\linewidth}\BODY\end{minipage}}%
}
```

\fb@put@frame {⟨*frame cmd no split*⟩} {⟨*frame cmd split*⟩}

```
\renewcommand*{\fb@put@frame}[2]{%
\relax%
\@tempboxa%
}
```

Package 32

lwarp-ftnright.sty

104 Ftnright

Pkg `ftnright` ftnright is ignored.

for HTML output: Discard all options for lwarp-ftnright:

```
\LWR@ProvidesPackageDrop{ftnright}
```

Package 33

lwarp-geometry.sty

105 Geometry

Pkg `geometry` geometry is preloaded by lwarp, but must be nullified as seen by the user's source code.

for HTML output: Discard all options for lwarp-geometry:

```
\LWR@ProvidesPackageDrop{geometry}
```

```
\renewcommand*{\geometry}[1]{}
\renewcommand*{\newgeometry}[1]{}
\renewcommand*{\restoregeometry}{}
\renewcommand*{\savegeometry}[1]{}
\renewcommand*{\loadgeometry}[1]{}
```

Package 34

lwarp-glossaries.sty

106 Glossaries

Pkg `glossaries` `xindy` is required for glossaries.

The default `style=item` option for glossaries conflicts with lwarp, so the style is forced to `index` instead.

The page number list in the printed form would become `\namerefs` in HTML, which could become a very long string if many items are referenced. For now, the number list is simply turned off.

`lwarpmk` has the commands `printglossary` and `htmlglossary` to process the glossaries created by glossaries using `xindy`.

Opt `IndexLanguage` The package lwarp takes an option `IndexLanguage=english` to set the language used by `xindy`. This is passed to `xindy` using its `-L` option, and is used for both index and glossary generation.

for HTML output:

```
\PassOptionsToPackage{xindy}{glossaries}
\LWR@ProvidesPackagePass{glossaries}
\setupglossaries{nonumberlist}
\setglossarystyle{index}
```

Package 35

lwarp-graphics.sty

107 Graphics

Pkg `graphics` graphics is emulated.

for HTML output:

```
\LWR@ProvidesPackageDrop{graphics}
```

Package 36

lwarp-graphicx.sty

108 Graphicx

Pkg `graphicx` graphicx is emulated.

for HTML output:

```
\LWR@ProvidesPackageDrop{graphicx}
```

Package 37

lwarp-hyperref.sty

109 Hyperref

Pkg hyperref hyperref is emulated during HTML output, and the hyperref package is ignored.

for HTML output:

```
% \LWR@ProvidesPackageDrop{hyperref}
\typeout{Using the lwarp html version of package ‘hyperref’ -- discarding options.}
\typeout{    Are not using ProvidesPackage, so that other packages}
\typeout{    do not attempt to patch lwarp’s version of ‘hyperref’.}
% \ProvidesPackage{lwarp-#1-#2}
\DeclareOption*{}
\ProcessOptions\relax

\newcommand*{\hypersetup}[1]{}
\newcommand*{\hyperbaseurl}[1]{}
```

Insert an image with alt text:

```
\NewDocumentCommand{\LWR@hyperimageb}{m +m}{%
\LWR@htmltag{img src="#1" alt="#2" class="hyperimage"{}}%
\endgroup%
\LWR@ensuredoingapar%
}

\newcommand{\hyperimage}{%
\LWR@ensuredoingapar%
\begingroup\catcode‘\_=12
\LWR@hyperimageb%
}
```

Creates an HTML anchor to `category.name` with the given text.

```
\NewDocumentCommand{\hyperdef}{m m +m}{%
\LWR@ensuredoingapar%
\LWR@subsublabel{#1.#2}%
#3%
}
```

Creates an HTML link to `URL#category.name` with the given text.

```
\NewDocumentCommand{\LWR@hyperrefb}{m m m +m}{%
\LWR@htmltag{a href="#1\LWR@hashmark#2.#3"}%
```

```
#4%
\LWR@htmltag{/a}%
\endgroup%
}
```

Creates text as an HTML link to the LaTeX label.

```
\NewDocumentCommand{\LWR@hyperrefc}{O{label} +m}{
\LWR@startref{#1}%
#2%
\LWR@htmltag{/a}%
\endgroup%
}

\newcommand{\hyperref}{%
\LWR@ensuredoingapar%
\begingroup\catcode`\_=12
\@ifnextchar[\LWR@hyperrefc\LWR@hyperrefb%
}
```

Creates an anchor to `name` with the given text.

```
\NewDocumentCommand{\hypertarget}{m +m}{%
\label{#1}%
#2%
}
```

Creates a link to the anchor created by hypertarget, with the given link text.

```
\NewDocumentCommand{\hyperlink}{m +m}{%
\hyperref[#1]{#2}%
}
```

For HTML, `\cleverref` is used instead.

```
\NewDocumentCommand{\autoref}{s m}{%
\IfBooleanTF{#1}{\ref{#2}}{\cref{#2}}%
}
```

For HTML, `\cleverref` is used instead.

```
\NewDocumentCommand{\autopageref}{s m}{%
\IfBooleanTF{#1}{\cpageref{#2}}{\cref{#2}}%
}

\newcommand{\pdfstringdef}[2]{}

\newcommand{\pdfbookmark}[3][]{}
```

```
\newcommand{\currentpdfbookmark}[2]{}
\newcommand{\subpdfbookmark}[2]{}
\newcommand{\belowpdfbookmark}[2]{}
\newcommand{\texorpdfstring}[2]{#2}
```

From hyperref.

```
\def\hypercalcbp#1{%
\strip@pt\dimexpr 0.99626401\dimexpr(#1)\relax\relax
}%
\newcommand{\Acrobatmenu}[2]{}
\newcommand*{\TextField}[2][]{}
\newcommand*{\CheckBox}[2][]{}
\newcommand{\ChoiceMenu}[3][]{}
\newcommand*{\PushButton}[2][]{}
\newcommand*{\Submit}[2][]{}
\newcommand*{\Reset}[2][]{}
\newcommand*{\LayoutTextField}[2]{}
\newcommand*{\LayoutChoiceField}[2]{}
\newcommand*{\LayoutCheckField}[2]{}
\newcommand*{\MakeRadioField}[2]{}
\newcommand*{\MakeCheckField}[2]{}
\newcommand*{\MakeTextField}[2]{}
\newcommand*{\MakeChoiceField}[2]{}
\newcommand{\MakeFieldButton}[1]{}
```

Package 38

lwarp-indentfirst.sty

110 Indentfirst

Pkg `indentfirst` indentfirst is ignored.

Discard all options for lwarp-indentfirst:

for HTML output:

```
\LWR@ProvidesPackageDrop{indentfirst}
```

Package 39

lwarp-inputenc.sty

111 Inputenc

Pkg `inputenc` Error if inputenc is loaded after lwarp.

Discard all options for lwarp-inputenc:

for HTML output:

```
\LWR@ProvidesPackageDrop{inputenc}
```

```
\LWR@loadbefore{inputenc}
```

Package 40

lwarp-keyfloat.sty

112 Keyfloat

Pkg `keyfloat` keyfloat is supported with minor adjustments.

for HTML output:

```
\LWR@ProvidesPackagePass{keyfloat}
```

After keyfloat has loaded:

```
\AtBeginDocument{

\let\KFLT@boxinner\relax
\let\endKFLT@boxinner\relax

\NewEnviron{KFLT@boxinner}
{%
\LWR@traceinfo{kflt@boxinner}%
\LWR@stoppars%
\KFLT@frame{\BODY}%
\LWR@startpars%
\LWR@traceinfo{ended kflt@boxinner}%
}

\DeclareDocumentEnvironment{KFLT@marginfloat}{O{-1.2ex} m}
{% start
\LWR@maybeincthisfloat%
\LWR@forcenewpage
\LWR@stoppars%
\LWR@htmltag{div class="marginblock" id="autofloat-\arabic{LWR@thisfloat}"}
\LWR@startpars%
\captionsetup{type=#2}%
}
{
\LWR@htmldivclassend{div}
}

\DeclareDocumentEnvironment{marginfigure}{o}
  {\begin{KFLT@marginfloat}{figure}}
  {\end{KFLT@marginfloat}}

\DeclareDocumentEnvironment{margintable}{o}
  {\begin{KFLT@marginfloat}{table}}
  {\end{KFLT@marginfloat}}
```

```
\DeclareDocumentEnvironment{keywrap}{m +m}
{%
\begin{BlockClass}{marginblock}
\setlength{\linewidth}{#1}
#2%
\end{BlockClass}
}
{%
}

}% AtBeginDocument
```

Package 41

lwarp-layout.sty

113 Layout

Pkg `layout` layout is ignored.

for HTML output: Discard all options for lwarp-layout:

```
\LWR@ProvidesPackageDrop{layout}
```

```
\NewDocumentCommand{\layout}{s}{}
```

Package 42

lwarp-letterspace.sty

114 Letterspace

Pkg `letterspace` letterspace is a subset of microtype, which is pre-loaded by lwarp. All user options and macros are ignored and disabled.

for HTML output: Discard all options for lwarp-letterspace:

```
\LWR@ProvidesPackageDrop{letterspace}
```

```
\newcommand*\lsstyle{}
\newcommand\textls[2][]{}
\def\textls#1#{}
\newcommand*\lslig[1]{#1}
```

Package 43

lwarp-lettrine.sty

115 Lettrine

(*Based on original code by* Daniel Flipo.)

Pkg `lettrine` Emulated.

for HTML output: Discard all options for lwarp-lettrine:

```
\LWR@ProvidesPackageDrop{lettrine}
```

The initial letter is in a `<span>` of class `lettrine`, and the following text is in a `<span>` of class `lettrinetext`. `\lettrine [`⟨*keys*⟩`] {`⟨*letter*⟩`} {`⟨*additional text*⟩`}`

```
\DeclareDocumentCommand{\lettrine}{o m m}{%
\InlineClass{lettrine}{#2}\InlineClass{lettrinetext}{#3} %
}

\newcounter{DefaultLines}
\setcounter{DefaultLines}{2}
\newcounter{DefaultDepth}
\newcommand*{\DefaultOptionsFile}{\relax}
\newcommand*{\DefaultLoversize}{0}
\newcommand*{\DefaultLraise}{0}
\newcommand*{\DefaultLhang}{0}
\newdimen\DefaultFindent
\setlength{\DefaultFindent}{\z@}
\newdimen\DefaultNindent
\setlength{\DefaultNindent}{0.5em}
\newdimen\DefaultSlope
\setlength{\DefaultSlope}{\z@}
\newdimen\DiscardVskip
\setlength{\DiscardVskip}{0.2\p@}
\newif\ifLettrineImage
\newif\ifLettrineOnGrid
\newif\ifLettrineRealHeight

\newcommand*{\LettrineTextFont}{\scshape}

\newcommand*{\LettrineFontHook}{}

\newcommand*{\LettrineFont}[1]{\InlineClass{lettrine}{#1}}
\newcommand*{\LettrineFontEPS}[1]{\includegraphics[height=1.5ex]{#1}}
```

Package 44

lwarp-lips.sty

116 Lips

Pkg `lips` lips is emulated during HTML output, and the lips package is ignored.

```
% \LWR@ProvidesPackageDrop{lips}
\PackageInfo{lwarp}{Using the lwarp version of package ‘lips’.}%
\ProvidesPackage{lwarp-lips}

\NewDocumentCommand{\Lips}{}{\textellipsis}

\NewDocumentCommand{\BracketedLips}{}{[\textellipsis]}

\let\lips\Lips
\let\olips\lips

\DeclareOption*{}
\DeclareOption{mla}{
\let\lips\BracketedLips
}
\ProcessOptions\relax

\newcommand \LPNobreakList {}
```

Package 45

lwarp-listings.sty

117 Listings

(*Based on original code by* Carsten Heinz, Brooks Moses, Jobst Hoffmann.)

Pkg `listings` listings is supported with some limitations. Text formatting is not yet supported.

for HTML output:

```
\begin{warpHTML}

\LWR@ProvidesPackagePass{listings}
```

Patches to embed listings inside `pre` tags:

```
\let\LWR@origlst@Init\lst@Init
\let\LWR@origlst@DeInit\lst@DeInit

\let\LWR@origlsthkEveryPar\lsthk@EveryPar

\renewcommand{\l@lstlisting}[2]{\hypertocfloat{1}{lstlisting}{lol}{#1}{#2}}
```

Done at the start of a listing.

```
\renewcommand{\lst@Init}[1]{%
```

First, perform the listings initialization:

```
\LWR@traceinfo{lst@Init}%
\renewcommand*{\@captype}{lstlisting}%
\LWR@origlst@Init{#1}%
\LWR@traceinfo{finished origlst@Init}%
\lst@ifdisplaystyle%
```

Creating a display.

Disable line numbers, produce the `<pre>`, then reenable line numbers.

```
\LWR@traceinfo{About to create verbatim.}%
\let\lsthk@EveryPar\relax%
\LWR@forcenewpage
\LWR@atbeginverbatim{programlisting}%

\let\lsthk@EveryPar\LWR@origlsthkEveryPar%
\else%
```

Inline, so open a

```
\ifbool{LWR@verbtags}{\LWR@htmltag{span class="inlineprogramlisting"}}{}%
\fi%
}
```

```
\renewcommand*{\lst@DeInit}{%
\lst@ifdisplaystyle%
```

Creating a display.

Disable line numbers, produce the </pre>, then reenable line numbers:

```
\let\lsthk@EveryPar\relax%

\LWR@afterendverbatim%
\let\lsthk@EveryPar\LWR@origlsthkEveryPar%
\else%
```

Inline, so create the closing :

```
\ifbool{LWR@verbtags}{\noindent\LWR@htmltag{/span}}{}%
\fi%
```

Final listings deinit:

```
\LWR@origlst@DeInit%
}
```

This is called BOTH at the top and at the bottom of each listing.

Patched for lwarp.

```
\def\lst@MakeCaption#1{%
\LWR@traceinfo{MAKING CAPTION at #1}%
  \lst@ifdisplaystyle
\LWR@traceinfo{making a listings display caption}%
    \ifx #1t%
         \ifx\lst@@caption\@empty\expandafter\lst@HRefStepCounter \else
                                 \expandafter\refstepcounter
         \fi {lstlisting}%
\LWR@traceinfo{About to assign label: !\lst@label!}%
%         \ifx\lst@label\@empty\else
% \label{\lst@label}\fi
\LWR@traceinfo{Finished assigning the label.}%
        \let\lst@arg\lst@intname \lst@ReplaceIn\lst@arg\lst@filenamerpl
        \global\let\lst@name\lst@arg \global\let\lstname\lst@name
        \lst@ifnolol\else
            \ifx\lst@@caption\@empty
```

```
                \ifx\lst@caption\@empty
                    \ifx\lst@intname\@empty \else \def\lst@temp{ }%
                    \ifx\lst@intname\lst@temp \else
```

This code places a contents entry for a non-float. This would have to be modified for lwarp:

```
\LWR@traceinfo{addcontents lst@name: -\lst@name-}%
%                            \addcontentsline{lol}{lstlisting}{\lst@name}
                    \fi\fi
                \fi
            \else
```

This would have to be modified for lwarp:

```
\LWR@traceinfo{addcontents lst@@caption: -\lst@@caption-}%
                 \addcontentsline{lol}{lstlisting}%
{\protect\numberline{\thelstlisting}%
{\protect\ignorespaces \lst@@caption \protect\relax}}%
            \fi
         \fi
     \fi
    \ifx\lst@caption\@empty\else
\LWR@traceinfo{lst@caption not empty-}%
        \lst@IfSubstring #1\lst@captionpos
            {\begingroup
\LWR@traceinfo{at the selected position}%
```

These space and box commands are not needed for HTML output:

```
%               \let\@@vskip\vskip
%               \def\vskip{\afterassignment\lst@vskip \@tempskipa}%
%               \def\lst@vskip{\nobreak\@@vskip\@tempskipa\nobreak}%
%               \par\@parboxrestore\normalsize\normalfont % \noindent (AS)
%               \ifx #1t\allowbreak \fi
              \ifx\lst@title\@empty
```

New lwarp code to create a caption:

```
                  \lst@makecaption\fnum@lstlisting{\ignorespaces \lst@caption}
              \else
```

New lwarp code to create a title:

```
%                  \lst@maketitle\lst@title % (AS)
\LWR@traceinfo{Making title: \lst@title}%
\begin{BlockClass}{lstlistingtitle}% lwarp
\lst@maketitle\lst@title% lwarp
\end{BlockClass}% lwarp
```

```
            \fi
\LWR@traceinfo{About to assign label: !\lst@label!}%
        \ifx\lst@label\@empty\else
\leavevmode% gets rid of bad space factor error
\GetTitleStringExpand{\lst@caption}%
\edef\LWR@lntemp{\GetTitleStringResult}%
\edef\@currentlabelname{\detokenize\expandafter{\LWR@lntemp}}%
\label{\lst@label}\fi
\LWR@traceinfo{Finished assigning the label.}%
```

Not needed for lwarp:

```
%              \ifx #1b\allowbreak \fi
            \endgroup}{}%
    \fi
\LWR@traceinfo{end of making a listings display caption}%
  \else
\LWR@traceinfo{INLINE}%
  \fi
\LWR@traceinfo{DONE WITH CAPTION at #1}%
}
```

Patched to keep left line numbers outside of the left margin, and place right line numbers in a field `\VerbatimHTMLWidth` wide.

```
\lst@Key{numbers}{none}{%
    \let\lst@PlaceNumber\@empty
    \lstKV@SwitchCases{#1}%
    {none&\\%
     left&\def\lst@PlaceNumber{%
% \llap{
\LWR@orignormalfont%
\lst@numberstyle{\thelstnumber}\kern\lst@numbersep%
% }
}
\\%
     right&\def\lst@PlaceNumber{\rlap{\LWR@orignormalfont
                \kern\VerbatimHTMLWidth \kern\lst@numbersep
                \lst@numberstyle{\thelstnumber}}}%
    }{\PackageError{Listings}{Numbers #1 unknown}\@ehc}}
```

```
\end{warpHTML}
```

Package 46

lwarp-longtable.sty

118 Longtable

Pkg `longtable` longtable is emulated during HTML output, and the longtable package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{longtable}
```

⚠ Longtable `\endhead`, `\endfoot`, and `\endlastfoot` rows are not used for HTML, and these rows should be disabled. Use

`\warpprintonly{row contents}`

instead of

`\begin{warpprint} ... \end{warpprint}`

Doing so helps avoid "Misplaced `\noalign`." when using `\begin`{warpprint}.

Keep the `\endfirsthead` row, which is still relevent to HTML output.

⚠ `\kill` is ignored, place a `\kill` line inside

`\begin{warpprint} ... \end{warpprint}`

or place it inside `\warpingprintonly`.

See:
http://tex.stackexchange.com/questions/43006/why-is-input-not-expandable

Env `longtable` * [⟨*horizalignment*⟩] {⟨*colspec*⟩} Emulates the `longtable` environment.

Per the caption package, the starred version steps the counter per caption. The unstarred version steps the counter once at the beginning, but not at each caption.

Options [c], [l], and [r] are thrown away.

```
\newenvironment{longtable*}[2][]{%
\LWR@floatbegin{table}%
\setcaptiontype{\LTcaptype}%
\caption@setoptions{longtable}%
\caption@setoptions{@longtable}%
\caption@LT@setup%
\booltrue{LWR@starredlongtable}%
\let\captionlistentry\LWR@LTcaptionlistentry%
\LWR@tabular{#2}
}
```

```
{\endLWR@tabular\LWR@floatend}

\newenvironment{longtable}[2][]{%
\LWR@floatbegin{table}%
\setcaptiontype{\LTcaptype}%
\caption@setoptions{longtable}%
\caption@setoptions{@longtable}%
\caption@LT@setup%
\refstepcounter{\LTcaptype}%
\let\captionlistentry\LWR@LTcaptionlistentry%
\LWR@tabular{#2}
}
{\endLWR@tabular\LWR@floatend}

```

Provided for compatibility, but ignored:

```
\newcounter{LTchunksize}
\def\endhead{\LWR@tabularendofline}% throws away options //[dim] and //*
\def\endfirsthead{\LWR@tabularendofline}
\def\endfoot{\LWR@tabularendofline}
\def\endlastfoot{\LWR@tabularendofline}
\newcommand\tabularnewline{\LWR@tabularendofline}
\newcommand{\setlongtables}{}% Obsolete command, does nothing.
\newlength{\LTleft}
\newlength{\LTright}
\newlength{\LTpre}
\newlength{\LTpost}
\newlength{\LTcapwidth}

\renewcommand*{\kill}{\LWR@tabularendofline}
```

Package 47

lwarp-lscape.sty

119 Lscape

Pkg `lscape` lscape is nullified.

for HTML output: Discard all options for lwarp-lscape.

```
\LWR@ProvidesPackageDrop{lscape}
```

```
\newenvironment*{landscape}{}{}
```

Package 48

lwarp-ltcaption.sty

120 Ltcaption

Pkg `ltcaption` ltcaption is emulated during HTML output, and the ltcaption package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{ltcaption}
```

`\LTcaptype` is already defined by lwarp.

`longtable*` is already defined by lwarp-longtable.

```
\newlength{\LTcapskip}
\newlength{\LTcapleft}
\newlength{\LTcapright}
\newcommand*{\LTcapmarginsfalse}{}
```

Package 49

lwarp-marginfix.sty

121 Marginfix

Pkg `marginfix` Not used.

for HTML output: Discard all options for lwarp-marginfix:

```
\LWR@ProvidesPackageDrop{marginfix}
```

```
\newcommand*{\marginskip}[1]{}
\newcommand*{\clearmargin}{}
\newcommand*{\softclearmargin}{}
\newcommand*{\extendmargin}[1]{}
\newcommand*{\mparshift}[1]{}
\newdimen\marginheightadjustment
\newdimen\marginposadjustment
\newcommand*{\blockmargin}[1][]{}
\newcommand*{\unblockmargin}[1][]{}
\newcommand*{\marginphantom}[2][]{}
```

Package 50

lwarp-marginnote.sty

122 Marginnote

Pkg `marginnote` Emulated.

for HTML output: Discard all options for lwarp-marginnote:

```
\LWR@ProvidesPackageDrop{marginnote}
```

```
\NewDocumentCommand{\marginnote}{o +m o}{\marginpar{#2}}
\newcommand*{\marginnoteleftadjust}{}
\newcommand*{\marginnoterightadjust}{}
\newcommand*{\marginnotetextwidth}{}
\let\marginnotetextwidth\textwidth
\newcommand*{\marginnotevadjust}{}
\newcommand*{\marginfont}{}
\newcommand*{\raggedleftmarginnote}{}
\newcommand*{\raggedrightmarginnote}{}
```

Package 51

lwarp-mcaption.sty

123 Mcaption

Pkg `mcaption` mcaption is nullified.

for HTML output: Discard all options for lwarp-mcaption:

```
\LWR@ProvidesPackageDrop{mcaption}
```

```
\newenvironment{margincap}{}{}
\newcommand*{\margincapalign}{}
\newlength{\margincapsep}
```

Package 52

lwarp-mdframed.sty

124 Mdframed

Pkg `mdframed` mdframed is loaded with options forced to `framemethod=none`.

for HTML output:

```
1 \LWR@ProvidesPackageDrop{mdframed}
```

support Most basic functionality is supported, including frame background colors and single-border colors and thickness, title and subtitle background colors and borders and thickness, border radius, and shadow. CSS classes are created for mdframed environments and frame titles.

⚠ loading When used, lwarp loads mdframed in HTML with `framemethod=none`.

font For title font, use

```
frametitlefont=\textbf,
```

instead of

```
frametitlefont=\bfseries,
```

where `\textbf` must appear just before the comma and will receive the following text as its argument (since the text happens to be between braces in the mdframed source). Since lwarp does not support `\bfseries` and friends, only one font selection may be made at a time.

theoremtitlefont `theoremtitlefont` is not supported, since the following text is not in braces in the mdframed source.

footnotes Footnotes are currently placed at the bottom of the HTML page.

ignored options `userdefinedwidth` and `align` are currently ignored.

CSS classes Environments created or encapsulated by mdframed are enclosed in a `<div>` of class `md<environmentname>`, or `mdframed` otherwise.

Frame titles are placed into a `<span>` of class `mdframedtitle`. Subtitles are in a `<span>` of class `mdframedsubtitle`, and likewise for subsubtitles.

Pre-existing hooks are used to patch extra functions before and after the frames.

amsthm must be loaded before mdframed

```
2 \LWR@origRequirePackage{amsthm}
```

Do not require Ti*k*z or pstricks:

```
\LWR@origRequirePackage[framemethod=none]{mdframed}
```

To handle CSS and paragraphs, patch code at start and end of environment and contents. `\LWR@origraggedright` helps avoid hyphenation.

```
\mdfsetup{
startcode={\LWR@mdframedstart\LWR@origraggedright},
endcode={\LWR@mdframedend},
startinnercode={\LWR@startpars\LWR@origraggedright},
endinnercode={\LWR@stoppars},
}
```

Given the mdframed key, print the color.

```
\newcommand*{\LWR@mdfprintcolor}[1]{%
\convertcolorspec{named}{\csuse{mdf@#1}}{HTML}\LWR@tempcolor%
\#\LWR@tempcolor
}
```

Given the mdframed key, print the length.

```
\newcommand*{\LWR@mdfprintlength}[1]{%
\rndprintlength{\csuse{mdf@#1@length}}
}
```

Actions before an mdframe starts.

Encapsulate a frame inside a `<div>` of the desired `class`.

```
\newcommand*{\LWR@mdframedstart}{%
```

Turn off paragraph handling during the generation of the encapsulating tags:

```
\LWR@stoppars%
```

Below, print HTML pt units:

```
\uselengthunit{PT}%
```

Open a `<div>` and with custom `class` and custom `style`:

```
\LWR@htmltagc{div class="\LWR@mdthisenv" \LWR@orignewline
style=" \LWR@orignewline
```

Convert and print the background color:

```
background: \LWR@mdfprintcolor{backgroundcolor} ; \LWR@orignewline
```

Convert and print the border color and width:

```
border: \LWR@mdfprintlength{linewidth} solid
\LWR@mdfprintcolor{linecolor} ; \LWR@orignewline
```

Convert and print the border radius:

```
border-radius: \LWR@mdfprintlength{roundcorner} ; \LWR@orignewline
```

Convert and print the shadow:

```
\ifbool{mdf@shadow}{%
box-shadow:
\LWR@mdfprintlength{shadowsize}
\LWR@mdfprintlength{shadowsize}
\LWR@mdfprintlength{shadowsize}
\LWR@mdfprintcolor{shadowcolor} ;
}
{box-shadow: none ;}
\LWR@orignewline

"}
% \LWR@htmldivclass{\LWR@mdthisenv}
```

`mdframed` environment may not work with the modified `\hspace` and `\rule`, so restore them to their originals while inside `mdframed`:

```
\let\hspace\LWR@orighspace%
\let\rule\LWR@origrule%
}
```

Actions after an mdframe ends.

After closing the `<div>`, globally restore to the default environment type:

```
\newcommand*{\LWR@mdframedend}{
```

Close the custom `<div>`:

```
\LWR@htmldivclassend{\LWR@mdthisenv}
```

Reset future custom class to the default:

```
\gdef\LWR@mdthisenv{mdframed}
```

Resume paragraph handling:

```
\LWR@startpars%
}
```

Encapsulation of the original which places the title inside a `<span>` of class `mdframedtitle`:

```
\let\LWR@origmdframedtitleenv\mdframedtitleenv

\newlength{\LWR@titleroundcorner}

\renewrobustcmd\mdframedtitleenv[1]{%
\LWR@origmdframedtitleenv{%
```

Below, print HTML pt lengths:

```
\uselengthunit{PT}%
```

Open a `<span>` with a custom `class` and custom `style`:

```
\LWR@htmltagc{span class="mdframedtitle" \LWR@orignewline
style=" \LWR@orignewline
```

Convert and print the title background color:

```
background:
\LWR@mdfprintcolor{frametitlebackgroundcolor}
; \LWR@orignewline
```

Convert and print the title rule:

```
\ifbool{mdf@frametitlerule}{%
border-bottom:
\LWR@mdfprintlength{frametitlerulewidth}
solid
\LWR@mdfprintcolor{frametitlerulecolor}
; \LWR@orignewline
}{}%
```

The title's top border radius is adjusted for the line width:

```
border-radius:
\setlength{\LWR@titleroundcorner}
{\maxof{\mdf@roundcorner@length-\mdf@linewidth@length}{0pt}}
\rndprintlength{\LWR@titleroundcorner}
\rndprintlength{\LWR@titleroundcorner}
0pt 0pt
\LWR@orignewline
```

Finish the custom style and the opening span tag:

```
" \LWR@orignewline
}% span
```

Restrict paragraph tags inside a span:

```
\begin{LWR@nestspan}%
```

Print the title inside the span:

```
#1%
```

Closee the span and unnest the paragraph tag restriction:

```
\LWR@htmltagc{/span}%
\end{LWR@nestspan}%
}
}
```

Common code for `\LWR@mdfsubtitle` and `\LWR@mdfsubsubtitle`.

Encapsulate the subtitle inside a `<span>` of class `mdframedsubtitle`:

```
\NewDocumentCommand{\LWR@mdfsubtitlecommon}{m o m}
{% the following empty line is required

```

Special handling for mdframed: Subtitles have `\par`s around them, so temporarily disable them here.

```
\let\par\LWR@origpar%
```

Open a `<span>` with a custom `class` and custom `style`:

```
\LWR@htmltagc{span class="mdframed#1title"
style=" \LWR@orignewline
```

Convert and print the background color:

```
background:
\LWR@mdfprintcolor{#1titlebackgroundcolor}
; \LWR@orignewline
```

Convert and print the above line:

```
\ifbool{mdf@#1titleaboveline}{%
border-top:
\LWR@mdfprintlength{#1titleabovelinewidth}
solid
\LWR@mdfprintcolor{#1titleabovelinecolor}
; \LWR@orignewline
}{}%
```

Convert and print the below line:

```
\ifbool{mdf@#1titlebelowline}{%
border-bottom:
\LWR@mdfprintlength{#1titlebelowlinewidth}
solid
\LWR@mdfprintcolor{#1titlebelowlinecolor}
; \LWR@orignewline
}{}%
```

Finish the custom style and the opening span tag:

```
"}% span
```

Restrict paragraph tags inside a span:

```
\begin{LWR@nestspan}%
```

Perform the original subtitle action:

```
\IfNoValueTF{#2}
{\csuse{LWR@origmdf#1title}{#3}}%
{\csuse{LWR@origmdf#1title}[#2]{#3}}%
```

Close the span and unnest the paragraph tag restriction:

```
\LWR@htmltagc{/span}% the following empty line is required
\end{LWR@nestspan}% must follow the /span or an extra <p> appears

}
```

```
\let\LWR@origmdfsubtitle\mdfsubtitle

\newcommand*{\LWR@mdfsubtitle}{%
\LWR@mdfsubtitlecommon{sub}%
}
\let\mdfsubtitle\LWR@mdfsubtitle
```

```
\let\LWR@origmdfsubsubtitle\mdfsubsubtitle

\newcommand*{\LWR@mdfsubsubtitle}{%
\LWR@mdfsubtitlecommon{subsub}%
}
\let\mdfsubsubtitle\LWR@mdfsubsubtitle
```

Stores the environment of the frame about to be created:

```
\newcommand*{\LWR@mdthisenv}{mdframed}
```

Modified from the original to remember the environment.

```
\renewrobustcmd*\newmdenv[2][]{%
\newenvironment{#2}%
{%
\mdfsetup{#1}%
\renewcommand*{\LWR@mdthisenv}{md#2}%
\begin{mdframed}%
}
{\end{mdframed}}%
}
```

Modified from the original to remember the environment.

```
\renewrobustcmd*{\surroundwithmdframed}[2][]{%
\BeforeBeginEnvironment{#2}{%
\renewcommand*{\LWR@mdthisenv}{md#2}%
\begin{mdframed}[#1]}%
\AfterEndEnvironment{#2}{\end{mdframed}}%
}
```

[⟨*numberedlike*⟩] {⟨*caption*⟩} [⟨*within*⟩]

Modified from the original to remember the environment.

```
\let\LWR@origmdtheorem\mdtheorem

\DeclareDocumentCommand{\LWR@mdtheorem}{O{} m o m o}{%
\LWR@origmdtheorem[#1]{#2}[#3]{#4}[#5]%
\BeforeBeginEnvironment{#2}{\renewcommand*{\LWR@mdthisenv}{md#2}}%
}

\let\mdtheorem\LWR@mdtheorem
```

[⟨*numberedlike*⟩] {⟨*caption*⟩} [⟨*within*⟩]

Modified from the original to remember the environment.

```
\DeclareDocumentCommand\newmdtheoremenv{O{} m o m o }{%
 \ifboolexpr{ test {\IfNoValueTF {#3}} and test {\IfNoValueTF {#5}} }%
    {\newtheorem{#2}{#4}}{%
     \IfValueTF{#3}{\newtheorem{#2}[#3]{#4}}{}%
     \IfValueTF{#5}{\newtheorem{#2}{#4}[#5]}{}%
    }%
\BeforeBeginEnvironment{#2}{%
\renewcommand*{\LWR@mdthisenv}{md#2}%
\begin{mdframed}[#1]}%
\AfterEndEnvironment{#2}{%
\end{mdframed}}%
}
```

Package 53

lwarp-microtype.sty

125 Microtype

Pkg `microtype` microtype is pre-loaded by lwarp. All user options and macros are ignored and disabled.

for HTML output: Discard all options for lwarp-microtype:

```
\LWR@ProvidesPackageDrop{microtype}
```

```
\DeclareDocumentCommand{\DeclareMicrotypeSet}{o m m}{}
\DeclareDocumentCommand{\UseMicrotypeSet}{o m}{}
\DeclareDocumentCommand{\DeclareMicrotypeSetDefault}{o m}{}
\DeclareDocumentCommand{\SetProtrusion}{o m m}{}
\DeclareDocumentCommand{\SetExpansion}{o m m}{}
\DeclareDocumentCommand{\SetTracking}{o m m}{}
\DeclareDocumentCommand{\SetExtraKerning}{o m m}{}
\DeclareDocumentCommand{\SetExtraSpacing}{o m m}{}
\DeclareDocumentCommand{\DisableLigatures}{o m}{}
\DeclareDocumentCommand{\DeclareCharacterInheritance}{o m m}{}
\DeclareDocumentCommand{\DeclareMicrotypeVariants}{m}{}
\DeclareDocumentCommand{\DeclareMicrotypeAlias}{m m}{}
\DeclareDocumentCommand{\LoadMicrotypeFile}{m}{}
\DeclareDocumentCommand{\DeclareMicrotypeBabelHook}{m m}{}
\DeclareDocumentCommand{\microtypesetup}{m}{}
\DeclareDocumentCommand{\microtypecontext}{m}{}
\DeclareDocumentCommand{\textmicrotypecontext}{m m}{#2}
\@ifpackageloaded{letterspace}{\let\MT@textls\relax}{%
\DeclareDocumentCommand{\lsstyle}{}{}
\DeclareDocumentCommand{\textls}{o +m}{}
\DeclareDocumentCommand{\lslig}{m}{#1}
}
\def\DeclareMicrotypeSet#1#{\@gobbletwo}
\def\DeclareMicrotypeVariants#1#{\@gobble}
\@onlypreamble\DeclareMicrotypeSet
\@onlypreamble\UseMicrotypeSet
\@onlypreamble\DeclareMicrotypeSetDefault
\@onlypreamble\DisableLigatures
\@onlypreamble\DeclareMicrotypeVariants
\@onlypreamble\DeclareMicrotypeBabelHook
```

Package 54

lwarp-mparhack.sty

126 Mparhack

Pkg `mparhack` Not used.

for HTML output: Discard all options for lwarp-mparhack:

```
\LWR@ProvidesPackageDrop{mparhack}
```

Package 55

lwarp-multicol.sty

127 Multicol

Pkg `multicol` multicol is emulated during HTML output, and the multicol package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{multicol}[2015/09/13]
```

Multicols are converted into a 1–3 column display, browser-supported.

The optional multicols heading is placed inside a `<div>` of class `multicolsheading`.

The content is placed inside a `<div>` of class `multicols`.

```
\begin{warpHTML}
```

```
\NewDocumentEnvironment{multicols}{s m o}
```

HTML div class to contain everything:

```
{
\LWR@forcenewpage
\BlockClass{multicols}
```

Optional HTML div class for the heading:

```
\IfValueTF{#3}{\begin{BlockClass}{multicolsheading}#3\end{BlockClass}}{}}
```

When done with the environment, close the div:

```
{\endBlockClass}
```

Emulated null functions which are not used in HTML:

```
\newcommand*{\columnbreak}{}
\newcommand*{\RLmulticolcolumns}{}
\newcommand*{\LRmulticolcolumns}{}

\newlength{\premulticols}
\newlength{\postmulticols}
\newlength{\multicolsep}
\newlength{\multicolbaselineskip}
\newlength{\multicoltolerance}
```

```
\newlength{\multicolpretolerance}
\newcommand*{\columnseprulecolor}{\normalcolor}
\newcounter{columnbadness}
\newcounter{finalcolumnbadness}
\newcounter{collectmore}
\newcounter{unbalance}
\newlength{\multicolovershoot}
\newlength{\multicolundershoot}

\end{warpHTML}
```

Package 56

lwarp-multirow.sty

128 Multirow

Pkg `multirow` multirow is emulated during HTML output, and the multirow package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{multirow}
```

Package 57

lwarp-nameref.sty

129 Nameref

Pkg `nameref` nameref is emulated by lwarp.

for HTML output: Discard all options for lwarp-nameref:

```
\typeout{Using the lwarp html version of package ‘nameref’ -- discarding options.}
\typeout{    Are not using ProvidesPackage, so that other packages}
\typeout{    do not attempt to patch lwarp’s version of ‘nameref’.}
\DeclareOption*{}
\ProcessOptions\relax
```

Package 58

lwarp-needspace.sty

130 Needspace

Pkg `needspace` needspace is not used during HTML conversion.

for HTML output: Discard all options for lwarp-needspace:

```
\LWR@ProvidesPackageDrop{needspace}

\newcommand*{\needspace}[1]{}
\DeclareDocumentCommand{\Needspace}{s m}{}
```

Package 59

lwarp-newclude.sty

131 Newclude

Pkg `newclude` Error if newclude is loaded after lwarp.

Discard all options for lwarp-newclude:

for HTML output:

```
\LWR@ProvidesPackageDrop{newclude}
```

```
\LWR@loadbefore{newclude}
```

Package 60

lwarp-newunicodechar.sty

132 Newunicodechar

Pkg `newunicodechar` Error if newunicodechar is loaded after lwarp.

Discard all options for lwarp-newunicodechar:

for HTML output:

```
\LWR@ProvidesPackageDrop{newunicodechar}
```

```
\LWR@loadbefore{newunicodechar}
```

Package 61

lwarp-nextpage.sty

133 Nextpage

Pkg `nextpage` nextpage is nullified.

for HTML output: Discard all options for lwarp-nextpage.

```
\LWR@ProvidesPackageDrop{nextpage}
```

```
\newcommand{\cleartoevenpage}[1][]{}
\newcommand{\movetoevenpage}[1][]{}
\newcommand{\cleartooddpage}[1][]{}
\newcommand{\movetooddpage}[1][]{}
```

Package 62

lwarp-nowidow.sty

134 Nowidow

Pkg `nowidow` nowidow is not used during HTML conversion.

Discard all options for lwarp-nowidow:

for HTML output:

```
\LWR@ProvidesPackageDrop{nowidow}

\newcommand*{\nowidow}[1][]{}
\newcommand*{\setnowidow}[1][]{}

\newcommand*{\noclub}[1][]{}
\newcommand*{\setnoclub}[1][]{}
```

Package 63

lwarp-ntheorem.sty

135 Ntheorem

(*Based on original code by* Wolfgang May, Andreas Schedler.)

Pkg `ntheorem` ntheorem is patched for use by lwarp.

CSS styling of theorems and proofs:

Theorem: `<div>` of class `theorembody<theoremstyle>`

Theorem Header: `<span>` of class `theoremheader<style>`

where `<theoremstyle>` is `plain`, `break`, etc.

⚠ Font control This conversion is not total. Font control is via CSS, and the custom LaTeX font settings are ignored.

⚠ Equation numbering ntheorem has a bug with equation numbering in AMS environments when the option `thref` is used. lwarp does not share this bug, so equations with `\split`, etc, are numbered correctly with lwarp's HTML output, but not with the print output. It is recommended to use cleveref instead of ntheorem's `thref` option.

Options `amsthm` or `standard` choose which set of theorems and proofs to initialize.

⚠ Disabled options The options `thmmarks` and `amsmath` are disabled, since they heavily modify the underlying math code. Theorem marks are emulated. The AMS-math modifications are not done.

Option `thref` is disabled because cleveref functions are used instead. `\thref` is emulated.

Option `hyperref` is disabled because lwarp emulated hyperref.

for HTML output: Some disabled options:

```
\DeclareOption{thref}{}


\newbool{LWR@ntheoremmarks}
\boolfalse{LWR@ntheoremmarks}

\DeclareOption{thmmarks}{
```

```
\booltrue{LWR@ntheoremmarks}
\newif\ifsetendmark\setendmarktrue
}


\newbool{LWR@ntheoremamsthm}
\boolfalse{LWR@ntheoremamsthm}

\DeclareOption{amsthm}{\booltrue{LWR@ntheoremamsthm}}


\DeclareOption{amsmath}{}
\DeclareOption{hyperref}{}


\LWR@ProvidesPackagePass{ntheorem}
```

Storage for the style being used for new theorems.

```
\newcommand{\LWR@newtheoremstyle}{plain}
```

Patched to remember the style being used for new theorems:

```
\gdef\theoremstyle#1{%
   \@ifundefined{th@#1}{\@warning
          {Unknown theoremstyle ‘#1’. Using ‘plain’}%
          \theorem@style{plain}
            \renewcommand{\LWR@newtheoremstyle}{plain}% new
}%
      {
\theorem@style{#1}
        \renewcommand{\LWR@newtheoremstyle}{#1}% new
}
}
```

Patched to remember the style for this theorem type, and set it later when the environment is started.

```

\gdef\@xnthm#1#2[#3]{%
  \ifthm@tempif
     \csedef{LWR@thmstyle#1}{\LWR@newtheoremstyle}% new
     \expandafter\@ifundefined{c@#1}%
        {\@definecounter{#1}}{}%
     \@newctr{#1}[#3]%
     \expandafter\xdef\csname the#1\endcsname{%
       \expandafter\noexpand\csname the#3\endcsname \@thmcountersep
          {\noexpand\csname\the\theoremnumbering\endcsname{#1}}}%
     \expandafter\gdef\csname mkheader@#1\endcsname
```

```
       {\csname setparms@#1\endcsname
        \@thm{#1}{#1}{#2}
}%
     \global\@namedef{end#1}{\@endtheorem}
     \AtBeginEnvironment{#1}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#1}}}% new
  \fi
}

\gdef\@ynthm#1#2{%
  \ifthm@tempif
     \csedef{LWR@thmstyle#1}{\LWR@newtheoremstyle}% new
     \expandafter\@ifundefined{c@#1}%
        {\@definecounter{#1}}{}%
     \expandafter\xdef\csname the#1\endcsname
        {\noexpand\csname\the\theoremnumbering\endcsname{#1}}%
     \expandafter\gdef\csname mkheader@#1\endcsname
       {\csname setparms@#1\endcsname
        \@thm{#1}{#1}{#2}
}%
     \global\@namedef{end#1}{\@endtheorem}
     \AtBeginEnvironment{#1}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#1}}}% new
  \fi
}

\gdef\@othm#1[#2]#3{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}%
   {\ifthm@tempif
     \csedef{LWR@thmstyle#1}{\LWR@newtheoremstyle}% new
     \global\@namedef{the#1}{\@nameuse{the#2}}%
     \expandafter\protected@xdef\csname num@addtheoremline#1\endcsname{%
              \noexpand\@num@addtheoremline{#1}{#3}}%
     \expandafter\protected@xdef\csname nonum@addtheoremline#1\endcsname{%
              \noexpand\@nonum@addtheoremline{#1}{#3}}%
    \theoremkeyword{#3}%
    \expandafter\protected@xdef\csname #1Keyword\endcsname
             {\the\theoremkeyword}%
     \expandafter\gdef\csname mkheader@#1\endcsname
       {\csname setparms@#1\endcsname
                \@thm{#1}{#2}{#3}
}%
     \global\@namedef{end#1}{\@endtheorem}
     \AtBeginEnvironment{#1}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#1}}}% new
  \fi}
}
```

Mimics a float by incrementing the float counter and generating an HTML anchor. These are used for list-of-theorem cross-references.

```
\newcommand{\LWR@inctheorem}{%
```

```
\addtocounter{LWR@thisfloat}{1}%
\LWR@stoppars%
\LWR@htmltag{a id="autofloat-\arabic{LWR@thisfloat}"{}}\LWR@htmltag{/a}%
\LWR@startpars%
}
```

The following are patched for CSS.

These were in individual files `thp.sty` for plain, `thmb.sty` for margin break, etc. They are gathered together here.

Each theorem is encased in a `BlockClass` environment of class `theorembody<style>`.

Each header is encased in an `\InlineClass` of class `theoremheader<style>`.

```
\gdef\newtheoremstyle#1#2#3{%
  \expandafter\@ifundefined{th@#1}%
   {\expandafter\gdef\csname th@#1\endcsname{%
    \def\@begintheorem####1####2{%
\LWR@forcenewpage% new
\BlockClass{theorembody#1}%\LWR@thisthmstyle% new
\LWR@inctheorem% new
#2}%
    \def\@opargbegintheorem####1####2####3{%
\LWR@forcenewpage% new
\BlockClass{theorembody#1}%\LWR@thisthmstyle% new
\LWR@inctheorem% new
#3}%
}%
}%
{\PackageError{\basename}{Theorem style #1 already defined}\@eha}
}

\renewtheoremstyle{plain}%
  {\item[\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderplain}{##1\ ##2\theorem@separator}]}%
  {\item[\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderplain}{##1\ ##2\ (##3)\theorem@separator}]}

\renewtheoremstyle{break}%
  {\item[
%  \rlap{\vbox{\hbox{
\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderbreak}{##1\ ##2\theorem@separator}\newline
%  }\hbox{\strut}}}
]}%
  {\item[
%  \rlap{\vbox{\hbox{
\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderbreak}{##1\ ##2\ (##3)\theorem@separator}\newline
```

```
%  }\hbox{\strut}}}
]}

\renewtheoremstyle{change}%
  {\item[\hskip\labelsep
\theorem@headerfont
\InlineClass{theoremheaderchange}{##2\ ##1\theorem@separator}]}%
  {\item[\hskip\labelsep
\theorem@headerfont
\InlineClass{theoremheaderchange}{##2\ ##1\ (##3)\theorem@separator}]}

\renewtheoremstyle{changebreak}%
  {\item[
%  \rlap{\vbox{\hbox{
\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderchangebreak}{##2\ ##1\theorem@separator}\newline
%  }\hbox{\strut}}}
]}%
  {\item[
%  \rlap{\vbox{\hbox{
\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderchangebreak}{##2\ ##1\ (##3)\theorem@separator}\newline
%  }\hbox{\strut}}}
]}

\renewtheoremstyle{margin}%
  {\item[\hskip\labelsep\theorem@headerfont
\InlineClass{theoremheadermargin}{##2 \qquad ##1\theorem@separator}
]}%
  {\item[\hskip\labelsep\theorem@headerfont
\InlineClass{theoremheadermargin}{##2 \qquad ##1\ (##3)\theorem@separator}
]}

\renewtheoremstyle{marginbreak}%
  {\item[\hskip\labelsep\theorem@headerfont
\InlineClass{theoremheadermarginbreak}{##2 \qquad ##1\theorem@separator}\newline
]}%
  {\item[\hskip\labelsep\theorem@headerfont
\InlineClass{theoremheadermarginbreak}{##2 \qquad ##1\ (##3)\theorem@separator}\newline
]}

\renewtheoremstyle{nonumberplain}%
  {\item[\theorem@headerfont\hskip\labelsep
\InlineClass{theoremheaderplain}{##1\theorem@separator}]}%
  {\item[\theorem@headerfont\hskip \labelsep
\InlineClass{theoremheaderplain}{##1\ (##3)\theorem@separator}]}

\renewtheoremstyle{nonumberbreak}%
  {\item[
%  \rlap{\vbox{\hbox{
```

```
\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderbreak}{##1\theorem@separator}\newline
%  }\hbox{\strut}}}
]}%
  {\item[
%  \rlap{\vbox{\hbox{
\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderbreak}{##1\ (##3)\theorem@separator}\newline
%  }\hbox{\strut}}}
]}

\renewtheoremstyle{empty}%
  {\item[]}%
  {\item[\theorem@headerfont \hskip\labelsep\relax
\InlineClass{theoremheaderplain}{##3}]}

\renewtheoremstyle{emptybreak}%
  {\item[]}%
  {\item[\theorem@headerfont \hskip\labelsep\relax
\InlineClass{theoremheaderplain}{##3}] \ \newline}
```

The following manually adjust the CSS for the standard configuration objects which are not a purely plain style:

```
\ifbool{LWR@ntheoremamsthm}{}{
% upright text via CSS
\newtheoremstyle{plainupright}%
  {\item[\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderplain}{##1\ ##2\theorem@separator}]}%
  {\item[\hskip\labelsep \theorem@headerfont
\InlineClass{theoremheaderplain}{##1\ ##2\ (##3)\theorem@separator}]}

% upright text and small caps header via CSS
\newtheoremstyle{nonumberplainuprightsc}%
  {\item[\theorem@headerfont\hskip\labelsep
\InlineClass{theoremheadersc}{##1\theorem@separator}]}%
  {\item[\theorem@headerfont\hskip \labelsep
\InlineClass{theoremheadersc}{##1\ (##3)\theorem@separator}]}
```

The following standard configuration is renewed using the new CSS:

```
\theoremstyle{plainupright}
\theorembodyfont{\upshape}
\theoremsymbol{\ensuremath{_\Box}}
\renewtheorem{Example}{Example}
\renewtheorem{example}{Example}
\renewtheorem{Beispiel}{Beispiel}
\renewtheorem{beispiel}{Beispiel}
\renewtheorem{Bemerkung}{Bemerkung}
```

```
\renewtheorem{bemerkung}{Bemerkung}
\renewtheorem{Anmerkung}{Anmerkung}
\renewtheorem{anmerkung}{Anmerkung}
\renewtheorem{Remark}{Remark}
\renewtheorem{remark}{Remark}
\renewtheorem{Definition}{Definition}
\renewtheorem{definition}{Definition}

\theoremstyle{nonumberplainuprightsc}
\theoremsymbol{\ensuremath{_\blacksquare}}
\renewtheorem{Proof}{Proof}
\renewtheorem{proof}{Proof}
\renewtheorem{Beweis}{Beweis}
\renewtheorem{beweis}{Beweis}
\qedsymbol{\ensuremath{_\blacksquare}}

\theoremsymbol{}
}% not amsthm
```

Only if the amsthm option was given:

```
\ifbool{LWR@ntheoremamsthm}{

\gdef\th@plain{%
  \def\theorem@headerfont{\normalfont\bfseries}\itshape%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodyplain}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderplain}{##1\ ##2.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodyplain}% new
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderplain}{##1\ ##2\ (##3).}
]}}

\gdef\th@nonumberplain{%
  \def\theorem@headerfont{\normalfont\bfseries}\itshape%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodyplain}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
```

```
  \InlineClass{theoremheaderplain}{##1.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodyplain}% new
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderplain}{##1\ (##3).}
]}}

\gdef\th@definition{%
  \def\theorem@headerfont{\normalfont\bfseries}\normalfont%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodydefinition}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderdefinition}{##1\ ##2.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodydefinition}% new
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderdefinition}{##1\ ##2\ (##3).}
]}}

\gdef\th@nonumberdefinition{%
  \def\theorem@headerfont{\normalfont\bfseries}\normalfont%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodydefinition}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderdefinition}{##1.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodydefinition}% new
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderdefinition}{##1\ (##3).}
]}}

\gdef\th@remark{%
```

```
  \def\theorem@headerfont{\itshape}\normalfont%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodyremark}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderremark}{##1\ ##2.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodyremark}% new
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderremark}{##1\ ##2\ (##3).}
]}}

\gdef\th@nonumberremark{%
  \def\theorem@headerfont{\itshape}\normalfont%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodyremark}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderremark}{##1.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodyremark}% new
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderremark}{##1\ (##3).}
]}}

\gdef\th@proof{%
  \def\theorem@headerfont{\normalfont\bfseries}\itshape%
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembodyproof}% new
\LWR@inctheorem% new
      \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderproof}{##1.}
]}%
  \def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembodyroof}% new
```

```
\LWR@inctheorem% new
     \item[\hskip\labelsep
%  \theorem@headerfont
  \InlineClass{theoremheaderproof}{##1\ (##3).}
]}}



\newcounter{proof}%
\if@thmmarks
\newcounter{currproofctr}%
\newcounter{endproofctr}%
\fi

\gdef\proofSymbol{\openbox}

\newcommand{\proofname}{Proof}

\newenvironment{proof}[1][\proofname]{
\th@proof
\def\theorem@headerfont{\itshape}%
\normalfont
\theoremsymbol{\ensuremath{_\blacksquare}}
\@thm{proof}{proof}{#1}
}%
{\@endtheorem}

}{}% amsthm option
```

Patched for CSS:

```
\let\LWR@origendtheorem\@endtheorem
\renewcommand{\@endtheorem}{%
\ifbool{LWR@ntheoremmarks}{%
\ifsetendmark%
\InlineClass{theoremendmark}{\csname\InTheoType Symbol\endcsname}%
\setendmarkfalse%
\fi%
}{}%
\LWR@origendtheorem%
\ifbool{LWR@ntheoremmarks}{\global\setendmarktrue}{}%
\endBlockClass%
}
```

```
\gdef\NoEndMark{\global\setendmarkfalse}
```

Redefined to reuse the float mechanism to add list-of-theorem links:

\thm@thmline {⟨*1: printed type*⟩} {⟨*2: #*⟩} {⟨*3: optional*⟩} {⟨*4: page*⟩}

```
\renewcommand{\thm@@thmline@noname}[4]{%
\hypertocfloat{1}{theorem}{thm}{#2 #3}{}%
}

\renewcommand{\thm@@thmline@name}[4]{%
\hypertocfloat{1}{theorem}{thm}{#1 #2 #3}{}%
}
```

This was redefined by ntheorem when loaded, so it is now redefined for lwarp:

```
\def\thm@@thmline{\thm@@thmline@name}
```

Patch for CSS:

```
\def\listtheorems#1{
\LWR@htmlelementclass{nav}{lothm}%
\begingroup
\c@tocdepth=-2%
\def\thm@list{#1}\thm@processlist
\endgroup
\LWR@htmlelementclassend{nav}{lothm}%
}
```

Proof QED symbol:

```

\newcommand{\qed}{\qquad\the\qedsymbol}

\AtBeginDocument{
\def\openbox{\text{\HTMLunicode{25A1}}}% UTF-8 white box
\def\blacksquare{\text{\HTMLunicode{220E}}}% UTF-8 end-of-proof
\def\Box{\text{\HTMLunicode{25A1}}}% UTF-8 white box
}
```

\thref {⟨*label*⟩}

```
\newcommand*{\thref}[1]{\cref{#1}}
```

Package 64

lwarp-pagenote.sty

136 Pagenote

Pkg `pagenote` pagenote works as-is.

It is only included as an `lwarp-pagenote.sty` file because past versions of lwarp used pagenote to emulate footnotes, and so the file may exist on current installations, and should be over-written by this newer version.

for HTML output:

```
\LWR@ProvidesPackagePass{pagenote}
```

Package 65

lwarp-parskip.sty

137 Parskip

Pkg `parskip` parskip is ignored.

for HTML output: Discard all options for lwarp-parskip.

```
\LWR@ProvidesPackageDrop{parskip}
```

Package 66

lwarp-placeins.sty

138 Placeins

Pkg `placeins` placeins is not used during HTML conversion.

Discard all options for lwarp-placeins:

for HTML output:

```
\LWR@ProvidesPackageDrop{placeins}
\newcommand*{\FloatBarrier}{}
```

Package 67

lwarp-ragged2e.sty

139 Ragged2e

Pkg `ragged2e` ragged2e is not used during HTML conversion.

Discard all options for lwarp-ragged2e:

for HTML output:

```
\LWR@ProvidesPackageDrop{ragged2e}
```

```
\newcommand*{\Centering}{\centering}
\newcommand*{\RaggedLeft}{\raggedleft}
\newcommand*{\RaggedRight}{\raggedright}
\newcommand*{\justifying}{}
\newlength{\CenteringLeftskip}
\newlength{\RaggedLeftLeftskip}
\newlength{\RaggedRightLeftskip}
\newlength{\CenteringRightskip}
\newlength{\RaggedLeftRightskip}
\newlength{\RaggedRightRightskip}
\newlength{\CenteringParfillskip}
\newlength{\RaggedLeftParfillskip}
\newlength{\RaggedRightParfillskip}
\newlength{\JustifyingParfillskip}
\newlength{\CenteringParindent}
\newlength{\RaggedLeftParindent}
\newlength{\RaggedRightParindent}
\newlength{\JustifyingParindent}
\newenvironment*{Center}{\center}{\endcenter}
\newenvironment*{FlushLeft}{\flushleft}{\endflushleft}
\newenvironment*{FlushRight}{\flushright}{\endflushright}
\newenvironment*{justify}{\justifying}{\endjustifying}
```

Package 68

lwarp-rotating.sty

140 Rotating

Pkg `rotating` rotating is emulated during HTML output, and the rotating package is ignored.

All rotations are ignored in HTML output.

for HTML output:

```
\LWR@ProvidesPackageDrop{rotating}

\let\sidewaystable\table
\let\endsidewaystable\endtable

\let\sidewaysfigure\figure
\let\endsidewaysfigure\endfigure

\newenvironment*{sideways}{}{}
\newenvironment*{turn}[1]{}{}
\newenvironment*{rotate}[1]{}{}
\NewDocumentCommand{\turnbox}{m +m}{#2}
\let\rotcaption\caption
\let\@makerotcaption\@makecaption
```

Package 69

lwarp-setspace.sty

141 Setspace

Pkg `setspace` setspace is not used during HTML conversion.

Discard all options for lwarp-setspace:

for HTML output:

```
\LWR@ProvidesPackageDrop{setspace}

\newcommand*{\setstretch}[1]{}
\newcommand*{\SetSinglespace}[1]{}
\newcommand*{\singlespacing}{}
\newcommand*{\onehalfspacing}{}
\newcommand*{\doublespacing}{}

\newenvironment*{singlespace}
{
\LWR@forcenewpage
\BlockClass{singlespace}
}
{\endBlockClass}

\newenvironment*{singlespace*}
{
\LWR@forcenewpage
\BlockClass{singlespace}
}
{\endBlockClass}

\newenvironment*{spacing}[1]{

}{

}

\newenvironment*{onehalfspace}
{
\LWR@forcenewpage
\BlockClass{onehalfspace}
}
{\endBlockClass}

\newenvironment*{doublespace}
{
```

```
\LWR@forcenewpage
\BlockClass{doublespace}
}
{\endBlockClass}
```

clearpage

Package 70

lwarp-showidx.sty

142 Showidx

Pkg `showidx` showidx is ignored.

for HTML output: Discard all options for lwarp-showidx:

```
\LWR@ProvidesPackageDrop{showidx}
```

Package 71

lwarp-showkeys.sty

143 Showkeys

Pkg `showkeys` showkeys is ignored.

for HTML output: Discard all options for lwarp-showkeys:

```
\LWR@ProvidesPackageDrop{showkeys}
```

```
\NewDocumentCommand{\showkeys}{s}{}
```

Package 72

lwarp-sidecap.sty

144 Sidecap

Pkg `sidecap` sidecap is nullified.

for HTML output: Discard all options for lwarp-sidecap.

```
\LWR@ProvidesPackageDrop{sidecap}
```

See:
http://tex.stackexchange.com/questions/45401/use-the-s-star-argument-with-newdocumentenvironment
regarding the creation of starred environments with xparse.

```
\NewDocumentEnvironment{SCtable}{soo}
{\IfValueTF{#3}{\table[#3]}{\table}}
{\endtable}

\ExplSyntaxOn
\cs_new:cpn {SCtable*} {\SCtable*}
\cs_new_eq:cN {endSCtable*} \endSCtable
\ExplSyntaxOff


\NewDocumentEnvironment{SCfigure}{soo}
{\IfValueTF{#3}{\figure[#3]}{\figure}}
{\endfigure}

\ExplSyntaxOn
\cs_new:cpn {SCfigure*} {\SCfigure*}
\cs_new_eq:cN {endSCfigure*} \endSCfigure
\ExplSyntaxOff


\newenvironment*{wide}{}{}
```

Package 73

lwarp-sidenotes.sty

145 Sidenotes

(*Based on original code by* Andy Thomas, Oliver Schebaum.)

Pkg `sidenotes` Patched for lwarp.

for HTML output: Load the original package:

```
\LWR@ProvidesPackagePass{sidenotes}
```

The following patch sidenotes for use with lwarp:

Stop paragraph handling while creating the caption:

```
\RenewDocumentCommand \sidecaption {s o o m}
{
\LWR@stoppars
  \captionsetup{style=sidecaption}
  \IfBooleanTF{#1}
  { % starred
    \IfNoValueOrEmptyTF{#2}
    {\marginnote{\caption*{#4}}}
    {\marginnote{\caption*{#4}}[#2]}
  }
  { % unstarred
  \IfNoValueOrEmptyTF{#2}
    {\def\@sidenotes@sidecaption@tof{#4}}
    {\def\@sidenotes@sidecaption@tof{#2}}
  \IfNoValueOrEmptyTF{#3}
    {\marginnote{\caption[\@sidenotes@sidecaption@tof]{#4}}}
    {\marginnote{\caption[\@sidenotes@sidecaption@tof]{#4}}[#3]}
  }
\LWR@startpars
}
```

Borrowed from the lwarp version of keyfloat:

```
\NewDocumentEnvironment{KFLTsidenotes@marginfloat}{O{-1.2ex} m}
{% start
\LWR@maybeincthisfloat%
\LWR@forcenewpage
\LWR@stoppars%
\LWR@htmltag{div class="marginblock" id="autofloat-\arabic{LWR@thisfloat}"}
```

```
\LWR@startpars%
\captionsetup{type=#2}%
}
{
\LWR@htmldivclassend{div}
}

\RenewDocumentEnvironment{marginfigure}{o}
  {\begin{KFLTsidenotes@marginfloat}{figure}}
  {\end{KFLTsidenotes@marginfloat}}

\RenewDocumentEnvironment{margintable}{o}
  {\begin{KFLTsidenotes@marginfloat}{table}}
  {\end{KFLTsidenotes@marginfloat}}
```

The following were changed by sidenotes, and now are reset back to their lwarp-supported originals:

Restoring the definition from the LaTeX 2_ε `article.cls` source:

```
\renewenvironment{figure*}
               {\@dblfloat{figure}}
               {\end@dblfloat}

\renewenvironment{table*}
               {\@dblfloat{table}}
               {\end@dblfloat}
```

Package 74

lwarp-soul.sty

146 Soul

(*Based on original code by* Melchior FRANZ.)

Pkg `soul` Emulated.

for HTML output:

```
\LWR@ProvidesPackageDrop{soul}
```

Storage for the colors to use:

```
\newcommand*{\LWR@soululcolor}{}

\newcommand*{\LWR@soulstcolor}{}

% \definecolor{LWR@soulhlcolordefault}{HTML}{F8E800}
% \newcommand*{\LWR@soulhlcolor}{LWR@soulhlcolordefault}
\newcommand*{\LWR@soulhlcolor}{}
```

Basic markup with CSS:

```
\newcommand{\so}[1]{\InlineClass{letterspacing}{#1}}
\newcommand{\caps}[1]{\InlineClass{capsspacing}{#1}}
```

Add colors if not empty:

```
\newcommand{\LWR@soulcolor}[4]{%
\ifcsempty{#2}%
{%
\InlineClass{#3}{#1}}%
{%
\convertcolorspec{named}{\csuse{#2}}{HTML}\LWR@tempcolor%
\InlineClass{#3}[#4: \#\LWR@tempcolor]{#1}%
}%
}

\newcommand{\ul}[1]{%
\LWR@soulcolor{#1}{LWR@soululcolor}{uline}{text-decoration-color}%
}

\newcommand{\st}[1]{
\LWR@soulcolor{#1}{LWR@soulstcolor}{sout}{text-decoration-color}%
}
```

```

\newcommand{\hl}[1]{
\LWR@soulcolor{#1}{LWR@soulhlcolor}{highlight}{background-color}%
}
```

Nullified:

```
\newcommand*{\soulaccent}[1]{}
\newcommand*{\soulregister}[2]{}
\newcommand{\sloppyword}[1]{#1}
\newcommand*{\sodef}[5]{\DeclareRobustCommand*#1[1]{\so{##1}}}
\newcommand*{\resetso}{}
\newcommand*{\capsdef}[5]{}
\newcommand*{\capsreset}{}
\newcommand*{\capssave}[1]{}
\newcommand*{\capsselect}[1]{}
\newcommand*{\setul}[2]{}
\newcommand*{\resetul}{}
\newcommand*{\setuldepth}[1]{}
\newcommand*{\setuloverlap}[1]{}
```

Set colors:

```
\newcommand*{\setulcolor}[1]{\renewcommand{\LWR@soululcolor}{#1}}
\newcommand*{\setstcolor}[1]{\renewcommand{\LWR@soulstcolor}{#1}}
\newcommand*{\sethlcolor}[1]{\renewcommand{\LWR@soulhlcolor}{#1}}
```

Long versions of the user-level macros:

```
\let\textso\so
\let\textul\ul
\let\texthl\hl
\let\textcaps\caps
```

Package 75

lwarp-subfig.sty

147 Subfig

(*Based on original code by* Steven Douglas Cochran.)

Pkg `subfig` subfig is supported and patched by lwarp.

⚠ `lof/lotdepth` At present, the package options for `lofdepth` and `lotdepth` are not working. These counters must be set separately after the package has been loaded.

horizontal spacing In the document source, use `\hfill` and `\hspace*` between subfigures to spread them apart horizontally. The use of other forms of whitespace may cause paragraph tags to be generated, resulting in subfigures appearing on the following lines instead of all on a single line.

for HTML output: Accept all options for lwarp-subfig:

```
\LWR@ProvidesPackagePass{subfig}
```

`\sf@@@subfloat {`⟨*1 type*⟩`} [`⟨*2 lof entry*⟩`] [`⟨*3 caption*⟩`] {`⟨*4 contents*⟩`}`

The outer minipage allows side-by-side subfloats with `\hfill` between.

```
\long\def\sf@@@subfloat#1[#2][#3]#4{%
\begin{minipage}{\linewidth}% new
\LWR@stoppars% new
    \@ifundefined{FBsc@max}{}%
        {\FB@readaux{\let\FBsuboheight\relax}}%
    \@tempcnta=\@ne
    \if@minipage
      \@tempcnta=\z@
    \else\ifdim \lastskip=\z@ \else
      \@tempcnta=\tw@
    \fi\fi
    \ifmaincaptiontop
      \sf@top=\sf@nearskip
      \sf@bottom=\sf@farskip
    \else
      \sf@top=\sf@farskip
      \sf@bottom=\sf@nearskip
    \fi
    \leavevmode
\setbox\@tempboxa \hbox{#4}%
    \@tempdima=\wd\@tempboxa
```

```
    \@ifundefined{FBsc@max}{}%
        {\global\advance\Xhsize-\wd\@tempboxa
         \dimen@=\ht\@tempboxa
         \advance\dimen@\dp\@tempboxa
         \ifdim\dimen@>\FBso@max
           \global\FBso@max\dimen@
         \fi}%
    \vtop\bgroup
      \vbox\bgroup
        \ifcase\@tempcnta
          \@minipagefalse
        \or
          \vskip\sf@top
        \or
          \ifdim \lastskip=\z@ \else
            \@tempskipb\sf@top\relax\@xaddvskip
          \fi
        \fi
        \sf@ifpositiontop{%
          \ifx \@empty#3\relax \else
            \sf@subcaption{#1}{#2}{#3}%
            \vskip\sf@capskip
            \vskip\sf@captopadj
          \fi\egroup
          \hrule width0pt height0pt depth0pt
\LWR@startpars% new
%  \box\@tempboxa
#4
\LWR@stoppars% new
        }{%
\LWR@startpars% new
        \@ifundefined{FBsc@max}%
            {
%  \box\@tempboxa
#4
}%
            {\ifx\FBsuboheight\relax
%                \box\@tempboxa
#4
             \else
%                \vbox to \FBsuboheight{\FBafil\box\@tempboxa\FBbfil}%
#4
             \fi}%
\LWR@stoppars% new
          \egroup
          \ifx \@empty#3\relax \else
            \vskip\sf@capskip
            \hrule width0pt height0pt depth0pt
            \sf@subcaption{#1}{#2}{#3}%
         \fi
```

```
        }%
      \vskip\sf@bottom
    \egroup
    \@ifundefined{FBsc@max}{}%
        {\addtocounter{FRobj}{-1}%
         \ifnum\c@FRobj=0\else
           \subfloatrowsep
         \fi}%
    \ifmaincaptiontop\else
      \global\advance\@nameuse{c@\@captype}\m@ne
    \fi
\end{minipage}% new
\LWR@startpars% new
  \endgroup\ignorespaces%
}%
```

\sf@subcaption {⟨*1 type*⟩} {⟨*2 lof entry*⟩} {⟨*3 caption*⟩}

```
\long\def\sf@subcaption#1#2#3{%
\LWR@stoppars% new
  \ifx \relax#2\relax \else
    \bgroup
      \let\label=\@gobble
      \let\protect=\string
      \def\@subcaplabel{%
        \caption@lstfmt{\@nameuse{p@#1}}{\@nameuse{the#1}}}%
      \sf@updatecaptionlist{#1}{#2}{\the\value{\@captype}}{\the\value{#1}}%
    \egroup
  \fi
  \bgroup
    \ifx \relax#3\relax
      \let\captionlabelsep=\relax
    \fi
%     \setbox0\vbox{%
%        \hb@xt@\the\@tempdima{%
%
% %          \hss
% %         \parbox[t]{\the\@tempdima}{%
%           \caption@make
%               {\@nameuse{sub\@captype name}}%
%               {\@nameuse{thesub\@captype}}%
%               {#3}
% %  }%
% %            \hss
%   }
%  }%
    \@ifundefined{FBsc@max}%
%         {\box0}%
{
%  \parbox[t]{\the\@tempdima}{%
```

```
\LWR@traceinfo{sfsubcap B1}% new
\LWR@htmlblocktag{figcaption}% new
\caption@make
{\@nameuse{sub\@captype name}}%
{\@nameuse{thesub\@captype}}%
{#3}
\LWR@htmlblocktag{/figcaption}% new
\LWR@traceinfo{sfsubcap B2}% new
%  }%
}%
        {\dimen@\ht0%
         \advance\dimen@\dp0%
         \ifdim\dimen@>\FBsc@max
           \global\FBsc@max\dimen@
         \fi
         \FB@readaux{\let\FBsubcheight\relax}%
         \ifx\FBsubcheight\relax
           \def\next{
%  \parbox[t]{\the\@tempdima}
}%
         \else
           \def\next{
%  \parbox[t][\FBsubcheight][t]{\the\@tempdima}
}%
         \fi
         \vbox{%
%            \hb@xt@\the\@tempdima{%

%              \hss
%              \next{%
\LWR@traceinfo{sfsubcap C1}% new
               \caption@make
                   {\@nameuse{sub\@captype name}}%
                   {\@nameuse{thesub\@captype}}%
                   {#3}
\LWR@traceinfo{sfsubcap C1}% new
%  }%
%              \hss

%  }
  }
}%
  \egroup
\LWR@startpars% new
}
```

`\caption@@@make` {⟨*caption label*⟩} {⟨*caption text*⟩}

```
\renewcommand\caption@@@make[2]{%
\LWR@startpars% new
```

```
  \sbox\@tempboxa{#1}%
  \ifdim\wd\@tempboxa=\z@
    \let\caption@lsep\relax
  \fi
  \caption@ifempty{#2}{%
    \let\caption@lsep\@empty
    \let\caption@tfmt\@firstofone
  }%
%   \@setpar{\@@par\caption@@par}\caption@@par
\renewcommand{\@@par}{\LWR@closeparagraph\LWR@orig@@par}% new
  \caption@applyfont
  \caption@fmt
    {\ifcaption@star\else
       \begingroup
         \captionlabelfont
         #1%
       \endgroup
     \fi}%
    {\ifcaption@star\else
       \begingroup
         \caption@iflf\captionlabelfont
         \relax\caption@lsep
       \endgroup
     \fi}%
    {{\captiontextfont
      \caption@ifstrut
        {\vrule\@height\ht\strutbox\@width\z@}%
        {}%
      \nobreak\hskip\z@skip % enable hyphenation
      \caption@tfmt{#2}
\LWR@ensuredoingapar% new
      \caption@ifstrut
        {\ifhmode\@finalstrut\strutbox\fi}%
        {}%
      \par}}
\LWR@stoppars% new
}
```

Patches for `\sf@sub@label`:

```
\def\subfloat@label{%
\LWR@ensuredoingapar% new
  \@ifnextchar(%   %) match left parenthesis
    {\sf@sub@label}
    {\sf@sub@label(Sub\@captype\space
                   \@ifundefined{thechapter}{}{\@nameuse{thechapter}\space}%
                   \@nameuse{p@sub\@captype}%
                   \@nameuse{thesub\@captype}.)}}
```

Patches for `\subref`.

The unstarred version uses a `\ref` link whose printed text comes from the `sub@<label>`:

```
\renewcommand{\sf@subref}[1]{%
\LWR@subnewref{#1}{sub@#1}%
}
```

The starred version uses the printed `sub@<label>` which is stored as if it were a page number:

```
\renewcommand{\sf@@subref}[1]{\LWR@origpageref{sub@#1}}
```

Defining new subfloats. The `l@sub<type>` for each is redefined.

```
\let\LWR@orig@newsubfloat\@newsubfloat

\def\@newsubfloat[#1]#2{%
\LWR@orig@newsubfloat[#1]{#2}%
\renewcommand{\l@sub#2}[2]{\hypertocfloat{2}{sub#2}{\ext@sub#2}{##1}{##2}}
}
```

Pre-defined for figures and tables:

```
\renewcommand{\l@subfigure}[2]{\hypertocfloat{2}{subfigure}{lof}{#1}{#2}}
\renewcommand{\l@subtable}[2]{\hypertocfloat{2}{subtable}{lot}{#1}{#2}}
% \def\subfigure{\subfloat}
% \def\subtable{\subfloat}
```

Package 76

lwarp-tabularx.sty

148 Tabularx

Pkg `tabularx` tabularx is emulated by lwarp.

for HTML output: Discard all options for lwarp-tabularx:

```
\LWR@ProvidesPackageDrop{tabularx}

\NewDocumentEnvironment{tabularx}{m o m}
{\tabular{#3}}
{\endtabular}

\NewDocumentEnvironment{tabularx*}{m o m}
{\tabular{#3}}
{\endtabular}
```

Package 77

lwarp-tabulary.sty

149 Tabulary

Pkg `tabulary` tabulary is emulated by lwarp.

for HTML output: Discard all options for lwarp-tabulary.

Column types `L`, `C`, `R`, and `J` are emulated by lwarp core code.

```
\LWR@ProvidesPackageDrop{tabulary}

\NewDocumentEnvironment{tabulary}{m o m}
{\tabular{#3}}
{\endtabular}

\NewDocumentEnvironment{tabulary*}{m o m}
{\tabular{#3}}
{\endtabular}

\newdimen\tymin
\newdimen\tymax
\def\tyformat{}
```

Package 78

lwarp-textpos.sty

150 Textpos

Pkg `textpos` textpos is emulated during HTML output, and the textpos package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{textpos}
```

```
\NewDocumentEnvironment{textblock}{m r()}{}{}
\NewDocumentEnvironment{textblock*}{m o r()}{}{}
\newcommand*{\TPGrid}[3][]{}
\NewDocumentCommand{\TPMargin}{s o}{}
\newcommand*{\textblockcolour}[1]{}
\newcommand*{\textblockrulecolour}[1]{}
\newcommand*{\textblockcolor}[1]{}
\newcommand*{\textblockrulecolor}[1]{}
\newcommand*{\tekstblokkulur}[1]{}
\newcommand*{\tekstblokrulekulur}[1]{}
\newlength{\TPHorizModule}
\newlength{\TPVertModule}
\newlength{\TPboxrulesize}
\newcommand{\textblocklabel}[1]{}
\newcommand*{\showtextsize}{}
\newcommand{\textblockorigin}[2]{}
```

Package 79

lwarp-theorem.sty

151 Theorem

(*Based on original code by* Frank Mittelbach.)

Pkg `theorem` theorem is patched for use by lwarp.

CSS styling of theorems and proofs:

Theorem: `<div>` of class `theorembody<theoremstyle>`

Theorem Header: `<span>` of class `theoremheader`

where `<theoremstyle>` is `plain`, `break`, etc.

for HTML output:

```
\LWR@ProvidesPackagePass{theorem}
```

Storage for the style being used for new theorems:

```
\newcommand{\LWR@newtheoremstyle}{plain}
```

Patched to remember the style being used for new theorems:

```
\gdef\theoremstyle#1{%
   \@ifundefined{th@#1}{\@warning
          {Unknown theoremstyle ‘#1’. Using ‘plain’}%
          \theorem@style{plain}%
           \renewcommand{\LWR@newtheoremstyle}{plain}% new
          }%
      {%
          \theorem@style{#1}%
          \renewcommand{\LWR@newtheoremstyle}{#1}% new
      }%
      \begingroup
        \csname th@\the\theorem@style \endcsname
      \endgroup}
```

Patched to remember the style for this theorem type, and set it later when the environment is started.

```
\gdef\@xnthm#1#2[#3]{%
   \expandafter\@ifdefinable\csname #1\endcsname
```

```
   {%
    \csedef{LWR@thmstyle#1}{\LWR@newtheoremstyle}% new
    \@definecounter{#1}\@newctr{#1}[#3]%
    \expandafter\xdef\csname the#1\endcsname
      {\expandafter \noexpand \csname the#3\endcsname
       \@thmcountersep \@thmcounter{#1}}%
    \def\@tempa{\global\@namedef{#1}}%
    \expandafter \@tempa \expandafter{%
      \csname th@\the \theorem@style
            \expandafter \endcsname \the \theorem@bodyfont
     \@thm{#1}{#2}}%
    \global \expandafter \let \csname end#1\endcsname \@endtheorem
    \AtBeginEnvironment{#1}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#1}}}% new
   }}

\gdef\@ynthm#1#2{%
\expandafter\@ifdefinable\csname #1\endcsname
   {
\csedef{LWR@thmstyle#1}{\LWR@newtheoremstyle}% new
\@definecounter{#1}%
    \expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
    \def\@tempa{\global\@namedef{#1}}\expandafter \@tempa
     \expandafter{\csname th@\the \theorem@style \expandafter
     \endcsname \the\theorem@bodyfont \@thm{#1}{#2}}%
    \global \expandafter \let \csname end#1\endcsname \@endtheorem
    \AtBeginEnvironment{#1}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#1}}}% new
   }}

\gdef\@othm#1[#2]#3{%
  \expandafter\ifx\csname c@#2\endcsname\relax
   \@nocounterr{#2}%
  \else
   \expandafter\@ifdefinable\csname #1\endcsname
   {
\csedef{LWR@thmstyle#1}{\LWR@newtheoremstyle}% new
\expandafter \xdef \csname the#1\endcsname
     {\expandafter \noexpand \csname the#2\endcsname}%
    \def\@tempa{\global\@namedef{#1}}\expandafter \@tempa
     \expandafter{\csname th@\the \theorem@style \expandafter
     \endcsname \the\theorem@bodyfont \@thm{#2}{#3}}%
    \global \expandafter \let \csname end#1\endcsname \@endtheorem
    \AtBeginEnvironment{#1}{\edef\LWR@thisthmstyle{\csuse{LWR@thmstyle#1}}}% new
}%
  \fi}
```

The following are patched for CSS.

These were in individual files `thp.sty` for plain, `thmb.sty` for margin break, etc. They are gathered together here.

Each theorem is encased in a `BlockClass` environment of class `theorembody<style>`.

Each header is encased in an `\InlineClass` of class `theoremheader`.

```
\gdef\th@plain{%\normalfont\itshape
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
        \item[\hskip\labelsep
\InlineClass{theoremheader}{##1\ ##2}
]}%
\def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
   \item[\hskip\labelsep
\InlineClass{theoremheader}{##1\ ##2\ (##3)}
]}
}

\gdef\th@break{%\normalfont\slshape
  \def\@begintheorem##1##2{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip \labelsep
\InlineClass{theoremheader}{##1\ ##2}\newline%
]}%
\def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
  \item[\hskip \labelsep
\InlineClass{theoremheader}{##1\ ##2\ (##3)}\newline
]}
}

\gdef\th@marginbreak{%\normalfont\slshape
  \def\@begintheorem##1##2{
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep %
\InlineClass{theoremheader}{##2 \qquad ##1}\newline
]}%
\def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep %
\InlineClass{theoremheader}{##2 \qquad ##1\ %
(##3)}\newline
]}
}

```

```
\gdef\th@changebreak{%\normalfont\slshape
  \def\@begintheorem##1##2{
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep
\InlineClass{theoremheader}{##2\ ##1}\newline
]}%
\def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep
\InlineClass{theoremheader}{ ##2\ ##1\ %
(##3)}\newline
]}
}

\gdef\th@change{%\normalfont\slshape
  \def\@begintheorem##1##2{
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep
\InlineClass{theoremheader}{##2\ ##1}
]}%
\def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep
\InlineClass{theoremheader}{##2\ ##1\ (##3)}
]}
}

\gdef\th@margin{%\normalfont\slshape
  \def\@begintheorem##1##2{
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep
\InlineClass{theoremheader}{##2 \qquad ##1}
]}%
\def\@opargbegintheorem##1##2##3{%
\LWR@forcenewpage% new
\BlockClass{theorembody\LWR@thisthmstyle}% new
\item[\hskip\labelsep
\InlineClass{theoremheader}{##2 \qquad ##1\ (##3)}
]}
}
```

Patched for CSS:

```
\gdef\@endtheorem{\endBlockClass\endtrivlist}
```

Package 80

lwarp-threeparttable.sty

152 Threeparttable

Pkg threeparttable

threeparttable is emulated during HTML output, and the threeparttable package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{threeparttable}
```

Prints the table note item header inside a CSS class of `tnoteitemheader`.

```
\newcommand{\LWR@printtablenote}[1]{\InlineClass{tnoteitemheader}{#1}}
```

To emulate threeparttable:

```
\newenvironment*{threeparttable}[1][b]{}{}
```

```
\newenvironment*{tablenotes}[1][]
{%
\LWR@forcenewpage
\BlockClass{tnotes}%
\setlist[description]{format=\LWR@printtablenote}%
\description%
}
{%
\enddescription%
\endBlockClass%
}
```

```
\newcommand{\tnote}[1]{\textsuperscript{#1}}
```

Package 81

lwarp-tikz.sty

153 Tikz

Pkg `tikz` tikz is supported.

Accept all options for lwarp-tikz:

```
\LWR@ProvidesPackagePass{tikz}
```

catcodes lwarp changes the catcode of $ for its own use. The Ti*k*z `babel` library temporarily changes catcodes back to normal for Ti*k*z's use. tikz v3.0.0 introduced the `babel` library which handles catcode changes. For older versions, lwarp must change $'s catcode itself.

for HTML output:

```
\begin{warpHTML}
```

```
\newboolean{LWR@tikzbabel}

\@ifpackagelater{tikz}{2013/12/20}% Test for Tikz version v3.0.0
{\usetikzlibrary{babel}\booltrue{LWR@tikzbabel}}
{\boolfalse{LWR@tikzbabel}}
```

Env `tikzpicture` `tikzpicture` environment is enclosed inside a `\lateximage`. May be used as-is, and its contents will be converted to an image.

```
\BeforeBeginEnvironment{tikzpicture}{%
\lateximage%
\ifbool{LWR@tikzbabel}% Test for Tikz version v3.0.0
{}%
{\catcode`\$=3} % dollar sign is math shift
}

\AfterEndEnvironment{tikzpicture}{%
\endlateximage%
\ifbool{LWR@tikzbabel}% Test for Tikz version v3.0.0
{}%
{\catcode`\$=\active}%
}
```

```
\end{warpHTML}
```

Package 82

lwarp-titleps.sty

154 Titleps

Pkg `titleps` titleps is loaded and used by lwarp during HTML output. All user options and macros are ignored and disabled.

Discard all options for lwarp-titleps:

for HTML output:

```
\LWR@ProvidesPackageDrop{titleps}
```

`\pagestyle` and `\thispagestyle` are already disabled in the lwarp code.

```
\RenewDocumentCommand{\newpagestyle}{m o m}{}
\RenewDocumentCommand{\renewpagestyle}{m o m}{}

\RenewDocumentCommand{\sethead}{o o o m m m}{}
\RenewDocumentCommand{\setfoot}{o o o m m m}{}

\RenewDocumentCommand{\settitlemarks}{s m}{}

\renewcommand*{\headrule}{}
\renewcommand*{\footrule}{}

\renewcommand*{\setheadrule}[1]{}
\renewcommand*{\setfootrule}[1]{}

\newcommand*{\makeheadrule}{}
\newcommand*{\makefootrule}{}

\renewcommand{\setmarkboth}[1]{}

\RenewDocumentCommand{\widenhead}{s o o m m}{}

\renewcommand*{\bottitlemarks}{}
\renewcommand*{\toptitlemarks}{}
\renewcommand*{\firsttitlemarks}{}
\renewcommand*{\nexttoptitlemarks}{}
\renewcommand*{\outertitlemarks}{}
\renewcommand*{\innertitlemarks}{}

\RenewDocumentCommand{\newtitlemark}{s m}{}
```

```
\RenewDocumentCommand{\pretitlemark}{s m m}{}

\renewcommand{\ifsamemark}[4]{}

\NewDocumentCommand{\setfloathead}{s o o o m m m m m}{}
\NewDocumentCommand{\setfloatfoot}{s o o o m m m m m}{}

\NewDocumentCommand{\nextfloathead}{s o o o m m m m m}{}
\NewDocumentCommand{\nextfloatfoot}{s o o o m m m m m}{}

\newcommand{\newmarkset}[1]{}

\NewDocumentCommand{\newextramarkset}{s m m}{}

\newcommand{\botextramarks}[1]{}
\newcommand{\topextramarks}[1]{}
\newcommand{\firstextramarks}[1]{}
\newcommand{\nexttopextramarks}[1]{}
\newcommand{\outerextramarks}[1]{}
\newcommand{\innerextramarks}[1]{}
```

Package 83

lwarp-titlesec.sty

155 Titlesec

Pkg `titlesec` titlesec is emulated. All user options and macros are ignored and disabled.

Discard all options for lwarp-titlesec:

for HTML output:

```
\LWR@ProvidesPackageDrop{titlesec}

\newcommand*{\titlelabel}[1]{}

\newcommand\titleformat{%
  \@ifstar{\ttl@format@s}%
          {\ttl@format@i}}
\newcommand{\ttl@format@s}[1]{}
\NewDocumentCommand{\ttl@format@i}{m o m m m m o}{}

\@ifundefined{@chapapp}{\let\@chapapp\chaptername}{}
\newcommand\chaptertitlename{\@chapapp}

\NewDocumentCommand{\titlespacing}{s m m m m o}{}

\newcommand*{\filright}{}
\newcommand*{\filcenter}{}
\newcommand*{\filleft}{}
\newcommand*{\fillast}{}
\newcommand*{\filinner}{}
\newcommand*{\filouter}{}

\newcommand\wordsep{\fontdimen\tw@\font \@plus
  \fontdimen\thr@@\font \@minus \fontdimen4\font}

\NewDocumentCommand{\titleline}{s o m}{}

\providecommand*\titlerule{\@ifstar{\ttl@row}{\ttl@rule}}
\newcommand*{\ttl@rule}[1][]{}
\newcommand*{\ttl@row}[2][]{}

\newcommand{\iftitlemeasuring}[2]{#2}

\newcommand{\assignpagestyle}[2]{#2}

\NewDocumentCommand{\titleclass}{m o m o}
```

Package 84

lwarp-titletoc.sty

156 Titletoc

Pkg `titletoc` titletoc is emulated. All user options and macros are ignored and disabled.

Discard all options for lwarp-titletoc:

for HTML output:

```
\LWR@ProvidesPackageDrop{titletoc}

\NewDocumentCommand{\dottedcontents}{m o m m m}{}

\newcommand{\titlecontents}{\@ifstar{\ttl@tcstar}{\ttl@tcnostar}}
\NewDocumentCommand{\ttl@tcstar}{m o m m m m o o o}{}
\NewDocumentCommand{\ttl@tcnostar}{m o m m m m o}{}

\newcommand{\contentsmargin}[2][]{}

\newcommand*{\thecontentslabel}{thecontentslabel}
\newcommand*{\thecontentspage}{thecontentspage}

\newcommand{\contentslabel}[2][]{\thecontentslabel}
\newcommand{\contentspage}[1][]{\thecontentspage}

\newcommand{\contentspush}[1]{}

\newcommand{\contentsuse}[2]{}

\newcommand*{\startcontents}[1][]{}
\newcommand*{\stopcontents}[1][]{}
\newcommand*{\resumecontents}[1][]{}

\newcommand{\printcontents}[4][]{}

\newcommand{\startlist}[2][]{}
\newcommand{\stoplist}[2][]{}
\newcommand{\resumelist}[2][]{}

\newcommand{\printlist}[4][]{}
```

Package 85

lwarp-titling.sty

157 Titling

Pkg `titling` titling is used by lwarp. The following patches are not needed by lwarp, but are required if the user requests titling.

lwarp uses page notes for footnotes, so the various titling footnote restyling commands have no effect.

Pass all options to lwarp-titling:

for HTML output:

```
\LWR@ProvidesPackagePass{titling}
```

Patch `\@bsmtitlempty`:

```
\let\LWR@orig@bsmtitlempty\@bsmtitlempty
\renewcommand*{\@bsmtitlempty}{%
\LWR@orig@bsmtitlempty%
\global\let\published\relax%
\global\let\subtitle\relax%
}
```

Patch `\keepthetitle`:

```
\let\LWR@origkeepthetitle\keepthetitle
\renewcommand*{\keepthetitle}{%
\LWR@orig@keepthetitle%
\global\let\@published\@empty%
\global\let\@subtitle\@empty%
}
```

Patch `\killtitle`:

```
\let\LWR@origkilltitle\killtitle
\renewcommand*{\killtitle}{%
\LWR@orig@killtitle%
\global\let\thepublished\relax%
\global\let\thesubtitle\relax%
}
```

Package 86

lwarp-tocloft.sty

158 Tocloft

Pkg `tocloft` tocloft is emulated. Most user options and macros are ignored and disabled. `\newlistof` and `\cftchapterprecis` are supported.

Discard all options for lwarp-tocloft:

for HTML output:

```
\LWR@ProvidesPackageDrop{tocloft}

\newcommand{\tocloftpagestyle}[1]{}

\newcommand*{\cftmarktoc}{}
\newcommand*{\cfttoctitlefont}{}
\newcommand*{\cftaftertoctitle}{}

\newlength{\cftbeforetoctitleskip}
\newlength{\cftaftertoctitleskip}

\newcommand*{\cftmarklof}{}
\newcommand*{\cftloftitlefont}{}
\newcommand*{\cftafterloftitle}{}

\newlength{\cftbeforeloftitleskip}
\newlength{\cftafterloftitleskip}

\newcommand*{\cftmarklot}{}
\newcommand*{\cftlottitlefont}{}
\newcommand*{\cftafterlottitle}{}

\newlength{\cftbeforelottitleskip}
\newlength{\cftafterlottitleskip}

\newcommand*{\cftdot}{.}
\providecommand*{\cftdotsep}{1}
\newcommand*{\cftnodots}{5000}

\providecommand{\cftdotfill}[1]{}

\newcommand*{\cftsetpnumwidth}[1]{}
\newcommand*{\cftsetrmarg}[1]{}
```

```
\newcommand*{\cftpnumalign}[1]{}

\newlength{\cftparskip}

\newlength{\cftbeforepartskip}
\newlength{\cftpartindent}
\newlength{\cftpartnumwidth}
\newcommand*{\cftpartfont}{}
\newcommand*{\cftpartpresnum}{}
\newcommand*{\cftpartaftersnum}{}
\newcommand*{\cftpartaftersnumb}{}
\newcommand*{\cftpartleader}{}
\newcommand*{\cftpartdotsep}{1}
\newcommand*{\cftpartpagefont}{}
\newcommand*{\cftpartafterpnum}{}

\newlength{\cftbeforechapskip}
\newlength{\cftchapindent}
\newlength{\cftchapnumwidth}
\newcommand*{\cftchapfont}{}
\newcommand*{\cftchappresnum}{}
\newcommand*{\cftchapaftersnum}{}
\newcommand*{\cftchapaftersnumb}{}
\newcommand*{\cftchapleader}{}
\newcommand*{\cftchapdotsep}{1}
\newcommand*{\cftchappagefont}{}
\newcommand*{\cftchapafterpnum}{}

\newlength{\cftbeforesecskip}
\newlength{\cftsecindent}
\newlength{\cftsecnumwidth}
\newcommand*{\cftsecfont}{}
\newcommand*{\cftsecpresnum}{}
\newcommand*{\cftsecaftersnum}{}
\newcommand*{\cftsecaftersnumb}{}
\newcommand*{\cftsecleader}{}
\newcommand*{\cftsecdotsep}{1}
\newcommand*{\cftsecpagefont}{}
\newcommand*{\cftsecafterpnum}{}

\newlength{\cftbeforesubsecskip}
\newlength{\cftsubsecindent}
\newlength{\cftsubsecnumwidth}
\newcommand*{\cftsubsecfont}{}
\newcommand*{\cftsubsecpresnum}{}
\newcommand*{\cftsubsecaftersnum}{}
\newcommand*{\cftsubsecaftersnumb}{}
\newcommand*{\cftsubsecleader}{}
\newcommand*{\cftsubsecdotsep}{1}
```

```
\newcommand*{\cftsubsecpagefont}{}
\newcommand*{\cftsubsecafterpnum}{}

\newlength{\cftbeforesubsubsecskip}
\newlength{\cftsubsubsecindent}
\newlength{\cftsubsubsecnumwidth}
\newcommand*{\cftsubsubsecfont}{}
\newcommand*{\cftsubsubsecpresnum}{}
\newcommand*{\cftsubsubsecaftersnum}{}
\newcommand*{\cftsubsubsecaftersnumb}{}
\newcommand*{\cftsubsubsecleader}{}
\newcommand*{\cftsubsubsecdotsep}{1}
\newcommand*{\cftsubsubsecpagefont}{}
\newcommand*{\cftsubsubsecafterpnum}{}

\newlength{\cftbeforeparaskip}
\newlength{\cftparaindent}
\newlength{\cftparanumwidth}
\newcommand*{\cftparafont}{}
\newcommand*{\cftparapresnum}{}
\newcommand*{\cftparaaftersnum}{}
\newcommand*{\cftparaaftersnumb}{}
\newcommand*{\cftparaleader}{}
\newcommand*{\cftparadotsep}{1}
\newcommand*{\cftparapagefont}{}
\newcommand*{\cftparaafterpnum}{}

\newlength{\cftbeforesubparaskip}
\newlength{\cftsubparaindent}
\newlength{\cftsubparanumwidth}
\newcommand*{\cftsubparafont}{}
\newcommand*{\cftsubparapresnum}{}
\newcommand*{\cftsubparaaftersnum}{}
\newcommand*{\cftsubparaaftersnumb}{}
\newcommand*{\cftsubparaleader}{}
\newcommand*{\cftsubparadotsep}{1}
\newcommand*{\cftsubparapagefont}{}
\newcommand*{\cftsubparaafterpnum}{}

\newlength{\cftbeforefigskip}
\newlength{\cftfigindent}
\newlength{\cftfignumwidth}
\newcommand*{\cftfigfont}{}
\newcommand*{\cftfigpresnum}{}
\newcommand*{\cftfigaftersnum}{}
\newcommand*{\cftfigaftersnumb}{}
\newcommand*{\cftfigleader}{}
\newcommand*{\cftfigdotsep}{1}
\newcommand*{\cftfigpagefont}{}
```

```
\newcommand*{\cftfigafterpnum}{}
```

```
\newlength{\cftbeforesubfigskip}
\newlength{\cftsubfigindent}
\newlength{\cftsubfignumwidth}
\newcommand*{\cftsubfigfont}{}
\newcommand*{\cftsubfigpresnum}{}
\newcommand*{\cftsubfigaftersnum}{}
\newcommand*{\cftsubfigaftersnumb}{}
\newcommand*{\cftsubfigleader}{}
\newcommand*{\cftsubfigdotsep}{1}
\newcommand*{\cftsubfigpagefont}{}
\newcommand*{\cftsubfigafterpnum}{}
```

```
\newlength{\cftbeforetabskip}
\newlength{\cfttabindent}
\newlength{\cfttabnumwidth}
\newcommand*{\cfttabfont}{}
\newcommand*{\cfttabpresnum}{}
\newcommand*{\cfttabaftersnum}{}
\newcommand*{\cfttabaftersnumb}{}
\newcommand*{\cfttableader}{}
\newcommand*{\cfttabdotsep}{1}
\newcommand*{\cfttabpagefont}{}
\newcommand*{\cfttabafterpnum}{}
```

```
\newlength{\cftbeforesubtabskip}
\newlength{\cftsubtabindent}
\newlength{\cftsubtabnumwidth}
\newcommand*{\cftsubtabfont}{}
\newcommand*{\cftsubtabpresnum}{}
\newcommand*{\cftsubtabaftersnum}{}
\newcommand*{\cftsubtabaftersnumb}{}
\newcommand*{\cftsubtableader}{}
\newcommand*{\cftsubtabdotsep}{1}
\newcommand*{\cftsubtabpagefont}{}
\newcommand*{\cftsubtabafterpnum}{}
```

```
\newcommand{\cftsetindents}[3]{}
```

```
\newcommand{\pagenumbersoff}[1]{}
\newcommand{\pagenumberson}[1]{}
```

Emulated through the `\newfloat` mechanism.

```
\NewDocumentCommand{\newlistof}{o m m m}
{%
\IfValueTF{#1}
```

```
{\newfloat{#2}{tbp}{#3}[#1]}
{\newfloat{#2}{tbp}{#3}}
\@namedef{listof#2}{\listof{#2}{#4}}
\@namedef{#2depth}{1}
\expandafter\newlength\csuse{cftbefore#2skip}
\expandafter\newlength\csuse{cft#2indent}
\expandafter\newlength\csuse{cft#2numwidth}
\@namedef{cft#2font}{}
\@namedef{cft#2presnum}{}
\@namedef{cft#2aftersnum}{}
\@namedef{cft#2aftersnumb}{}
\@namedef{cft#2leader}{}
\@namedef{cft#2dotsep}{1}
\@namedef{cft#2pagefont}{}
\@namedef{cft#2afterpnum}{}
}
```

`\cftchapterprecis` from tocloft:

```
\newcommand{\cftchapterprecis}[1]{%
  \cftchapterprecishere{#1}
  \cftchapterprecistoc{#1}}
\newcommand{\cftchapterprecishere}[1]{%
  \begin{quote}\textit{#1}\end{quote}}
\newcommand{\cftchapterprecistoc}[1]{
  \addtocontents{toc}{%
  {
      \protect\begin{quote}#1\protect\end{quote}}
  }
}
```

Package 87

lwarp-trivfloat.sty

159 Trivfloat

Pkg `trivfloat` trivfloat is forced to use the built-in lwarp emulation for floats.

Discard all options for lwarp-trivfloat. This tells trivfloat not to use floatrow or memoir.

for HTML output:

```
\LWR@ProvidesPackageDrop{trivfloat}
\LWR@origRequirePackage{trivfloat}
```

for HTML & PRINT:

```
\begin{warpall}
```

To create a new float type and change its name:

```
\trivfloat{example}
\renewcommand{\examplename}{Example Name}
\crefname{example}{example}{examples}
\Crefname{example}{Example}{Examples}
```

```
\end{warpall}
```

`\tfl@chapter@fix` Nullified at the beginning of the document. Is used by trivfloat to correct float chapter numbers, but is not needed for lwarp.

for HTML output:

```
\begin{warpHTML}
```

```
\AtBeginDocument{\DeclareDocumentCommand{\tfl@chapter@fix}{m m}{}}
```

```
\end{warpHTML}
```

159.1 Combining `\newfloat`, `\trivfloat`, and algorithmicx

for HTML & PRINT:

```
\begin{warpall}
```

For both print and HTML output:

⚠ When using float, trivfloat, or algorithmicx at the same time, be aware of conflicting file usage. algorithmicx uses `.loa`. trivfloat by default starts with `.loa` and goes up

for additional floats, skipping `.lof` and `.lot`.

⚠ When using `\newfloat`, be sure to manually assign higher letters to the `\newfloat` files to avoid `.loa` used by algorithmicx, and any files used by trivfloat. Also avoid using `.lof` and `.lot`.

⚠ When using `\trivfloat`, you may force it to avoid conflicting with algorithmicx by starting trivfloat's file extensions with `.lob`:

```
\makeatletter
\setcounter{tfl@float@cnt}{1}} % start trivfloats with .lob
\makeatletter
```

```
\end{warpall}
```

Package 88

lwarp-ulem.sty

160 Ulem

(*Based on original code by* Donald Arseneau.)

Pkg `ulem` Emulated.

for HTML output: Original lwarp definitions:

```
\let\LWR@ulemorigemph\emph
\let\LWR@ulemorigtextbf\textbf
```

Basic markup commands, using CSS:

```
\NewDocumentCommand{\uline}{+m}{%
\InlineClass{uline}{#1}%
}

\NewDocumentCommand{\uuline}{+m}{%
\InlineClass{uuline}{#1}%
}

\NewDocumentCommand{\uwave}{+m}{%
\InlineClass{uwave}{#1}%
}

\NewDocumentCommand{\sout}{+m}{%
\InlineClass{sout}{#1}%
}

\NewDocumentCommand{\xout}{+m}{%
\InlineClass{xout}{#1}%
}

\NewDocumentCommand{\dashuline}{+m}{%
\InlineClass{dashuline}{#1}%
}

\NewDocumentCommand{\dotuline}{+m}{%
\InlineClass{dotuline}{#1}%
}
```

Nullified parameters:

```
\NewDocumentCommand{\ULthickness}{}{}
\newlength{\ULdepth}
```

Nullified/emulated macros:

```
\NewDocumentCommand{\markoverwith}{m}{}
\NewDocumentCommand{\ULon}{+m}{\uline{#1}\egroup}
```

`\useunder` only works with `\textbf`, etc, but not `\bfseries`, etc.

```
\NewDocumentCommand{\useunder}{m m m}{%
\relax%
\ifx\relax#3\relax\else % argumentative command
    \def#3{#1}\MakeRobust{#3}\fi
}
```

Triggered by package options, also available for the users:

```
\newcommand*{\normalem}{\let\emph\LWR@ulemorigemph}
\newcommand*{\ULforem}{\let\emph\uline}
\ULforem% default
```

Package options:

```
\DeclareOption{normalem}{\normalem}
\DeclareOption{ULforem}{\ULforem}
\DeclareOption{normalbf}{}
\DeclareOption{UWforbf}{\useunder{\uwave}{\bf}{\textbf}}
```

Emulate the original package:

```
\LWR@ProvidesPackageDrop{ulem}
```

Package 89

lwarp-verse.sty

161 Verse

(*Based on original code by* Peter Wilson.)

Pkg `verse` verse is supported and patched by lwarp.

for HTML output: Pass all options for lwarp-verse:

```
\LWR@ProvidesPackagePass{verse}
```

`\attrib` The documentation for the verse and memoir packages suggest defining an `\attrib` command, which may already exist in current documents, but it will only work for print output. lwarp provides `\attribution`, which works for both print and HTML output. To combine the two so that `\attrib` is used for print and `\attribution` is used for HTML:

```
\begin{warpHTML}
\let\attrib\attribution
\end{warpHTML}
```

Len `\leftskip`
Len `\leftmargini`
Len `\TMLvleftskip`
Len `\TMLleftmargini`

These lengths are used by verse and memoir to control the left margin, and they may already be set by the user for print output. New lengths `\HTMLvleftskip` and `\HTMLleftmargini` are provided to control the margins in HTML output. These new lengths may be set by the user before any `verse` environment, and persist until they are manually changed again. One reason to change `\HTMLleftmargini` is if there is a wide `\flagverse` in use, such as the word "Chorus", in which case the value of `\HTMLleftmargini` should be set to a wide enough length to contain "Chorus". The default is wide enough for a stanza number.

Horizontal spacing relies on `pdftotext`'s ability to discern the layout (`-layout` option) of the text in the HTML-tagged PDF output. For some settings of `\HTMLleftmargini` or `\HTMLleftskip` the horizontal alignment may not work out exactly, in which case a label may be shifted by one space.

Env `verse` The `verse` environment will be placed inside a HTML `pre`.

```
\AfterEndPreamble{
```

At the beginning of the `verse` environment:

```
\AtBeginEnvironment{verse}
{%
```

Pkg `verse`
Pkg `memoir`
`\flagverse`
Len `\leftskip`

The verse or memoir packages can place stanza numbers to the left with their `\flagverse` command. Do not allow them to go into the left margin, which would cause `pdfcrop` to crop the entire page further to the left:

```
\ifdef{\vleftskip}{%
\setlength{\vleftskip}{\HTMLvleftskip}
\setlength{\leftmargini}{\HTMLleftmargini}
}{}
\LWR@forcenewpage
\LWR@atbeginverbatim{verse}
\unskip\vspace{-\baselineskip}
}
```

After the end of the `verse` environment, which places the `pre` tag at the regular left margin:

```
\AfterEndEnvironment{verse}{
\unskip\vspace{-\baselineskip}
\LWR@afterendverbatim
}
```

Patch to place poemtitle inside an HTML span of class `poemtitle`:

```
\ifdef{\poemtitle}{
\DeclareDocumentCommand{\@vstypeptitle}{m}{%
   \vspace{\beforepoemtitleskip}%
   {\InlineClass{poemtitle}{\poemtitlefont #1}\par}%
   \vspace{\afterpoemtitleskip}%
 }
}{}

}
```

Package 90

lwarp-wallpaper.sty

162 Wallpaper

Pkg `wallpaper` wallpaper is emulated during HTML output, and the wallpaper package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{wallpaper}

\newcommand*{\CenterWallPaper}[2]{}
\newcommand*{\ThisCenterWallPaper}[2]{}
\newcommand*{\TileWallPaper}[3]{}
\newcommand*{\ThisTileWallPaper}[3]{}
\newcommand*{\TileSquareWallPaper}[2]{}
\newcommand*{\ThisTileSquareWallPaper}[2]{}
\newcommand*{\ULCornerWallPaper}[2]{}
\newcommand*{\ThisULCornerWallPaper}[2]{}
\newcommand*{\LLCornerWallPaper}[2]{}
\newcommand*{\ThisLLCornerWallPaper}[2]{}
\newcommand*{\URCornerWallPaper}[2]{}
\newcommand*{\ThisURCornerWallPaper}[2]{}
\newcommand*{\LRCornerWallPaper}[2]{}
\newcommand*{\ThisLRCornerWallPaper}[2]{}
\newcommand*{\ClearWallPaper}{}
\newlength{\wpXoffset}
\newlength{\wpYoffset}
```

Package 91

lwarp-wrapfig.sty

163 Wrapfig

Pkg `wrapfig` wrapfig is emulated during HTML output, and the wrapfig package is ignored.

for HTML output:

```
\LWR@ProvidesPackageDrop{wrapfig}
```

Computed width of a wrapped object. Used to print the HTML style.

```
\newlength{\LWR@wrapwidth}


\newcommand*{\LWR@wrapposition}{}

\newcommand*{\LWR@subwrapfigure}[2]{%
\LWR@maybeincthisfloat%
\renewcommand*{\LWR@wrapposition}{}%
\ifthenelse{%
\equal{#1}{r}\OR\equal{#1}{R}\OR%
\equal{#1}{o}\OR\equal{#1}{O}%
}%
{\renewcommand*{\LWR@wrapposition}{float:right}}%
{\renewcommand*{\LWR@wrapposition}{float:left}}%
\setlength{\LWR@wrapwidth}{#2}%
\addtolength{\LWR@wrapwidth}{4em}%
\uselengthunit{PT}%
\LWR@forcenewpage
\LWR@stoppars%
\LWR@htmltag{div class="marginblock" id="autofloat-\arabic{LWR@thisfloat}"
style="width:\rndprintlength{\LWR@wrapwidth} ; %
\LWR@wrapposition"%
}
\LWR@startpars
}


\NewDocumentEnvironment{wrapfigure}{o m o m}
{%
\LWR@subwrapfigure{#2}{#4}%
\captionsetup{type=figure}%
}
{
\LWR@htmldivclassend{div}
```

```
}


\NewDocumentEnvironment{wraptable}{o m o m}
{%
\LWR@subwrapfigure{#2}{#4}%
\captionsetup{type=table}%
}
{
\LWR@htmldivclassend{div}
}


\NewDocumentEnvironment{wrapfloat}{m o m o m}
{%
\LWR@subwrapfigure{#3}{#5}%
\captionsetup{type=#1}%
}
{
\LWR@htmldivclassend{div}
}

\newlength{\wrapoverhang}
```

Package 92

lwarp-xcolor.sty

164 Xcolor

Pkg `xcolor` xcolor is supported by lwarp.

support Color definitions, models, and mixing are fully supported without any changes required.

tables Colored tables are ignored so far. Use CSS to style tables.

colored text and boxes `\textcolor`, `\colorbox`, and `\fcolorbox` are supported.

`\color` and `\pagecolor` `\color` and `\pagecolor` are ignored. Use CSS or `\textcolor` where possible.

for HTML output:

```
\LWR@ProvidesPackagePass{xcolor}
```

```
\newcommand*{\LWR@tempcolor}{}
```

defaulting to black.

```
\newcommand*{\LWR@currenttextcolor}{black}
```

`\LWR@colorstyle` {⟨*1: styletext*⟩} {⟨*2: model*⟩} {⟨*3: color*⟩} {⟨*4: spancontents*⟩}

Creates a styled span with a color converted to HTML hex colorspace. Uses `LWR@spandepth` to prevent paragraph tags inside the span. If used for `\textcolor`, with a styletext of `color:`, then the new color is copied into `\LWR@currenttextcolor` for possible re-use in `\rule`.

```
\NewDocumentCommand{\LWR@colorstyle}{m m m m}{%
```

Use the xcolor package to convert to an HTML color space:

```
\convertcolorspec{#2}{#3}{HTML}\LWR@tempcolor%
```

If is a `\textcolor`, save a copy of this color for use by `\rule`:

```
\ifthenelse{\equal{#1}{color:}}%
{\renewcommand*{\LWR@currenttextcolor}{\#\LWR@tempcolor}}{}%
```

Create the HTML `<span>` with the styled color:

```
\LWR@htmltagc{span style="#1\#\LWR@tempcolor"{}}%
\begin{LWR@nestspan}%
```

Prevent additional paragraph tags inside this span:

Print the contents then close the span:

```
#4%
\LWR@htmltagc{/span}%
\end{LWR@nestspan}%
```

For paragraph-tag handling:

```
\LWR@ensuredoingapar%
}
```

`\color` appears in the LaTeX PDF output, but is ignored by `pdftotext` and thus is ignored in the HTML file. Text styling by local group is not yet supported.

Each of the following macros is given a temporary name, and is `\let` to the final name once the HTML conversion starts.

`\textcolor [`⟨*model*⟩`] {`⟨*color*⟩`} {`⟨*text*⟩`}` is converted into an HTML hex color span.

```
\NewDocumentCommand{\LWR@textcolor}{O{named} m m}{%
\begingroup%
\LWR@colorstyle{color:}{#1}{#2}{#3}%
\endgroup%
}
```

`\pagecolor [`⟨*model*⟩`] {`⟨*color*⟩`}` is ignored. Use `\CSSFilename` instead.

```
\newcommand*{\LWR@pagecolor}[2][named]{}
```

`\colorbox [`⟨*model*⟩`] {`⟨*color*⟩`} {`⟨*text*⟩`}` is converted into an HTML hex background color span.

```
\NewDocumentCommand{\LWR@colorbox}{O{named} m m}{%
\begingroup%
\LWR@colorstyle{background:}{#1}{#2}{#3}%
\endgroup%
}
```

`\fcolorbox [`⟨*framemodel*⟩`] {`⟨*framecolor*⟩`} [`⟨*boxmodel*⟩`] {`⟨*boxcolor*⟩`} {`⟨*text*⟩`}` is converted into a framed HTML hex background color span.

A background color of "none" creates a colored frame without a background color.

```
\NewDocumentCommand{\LWR@fcolorbox}{O{named} m O{named} m m}{%
\begingroup%
\ifthenelse{\equal{#4}{none}}{% no background color
\LWR@colorstyle{border:1px solid }{#1}{#2}{#5}%
}{% yes background color
\LWR@colorstyle{border:1px solid }{#1}{#2}%
{\LWR@colorstyle{background:}{#3}{#4}{#5}}%
}%
\endgroup%
}
```

Redirect to new definitions:

```
\let\textcolor\LWR@textcolor
\let\pagecolor\LWR@pagecolor
\let\colorbox\LWR@colorbox
\let\fcolorbox\LWR@fcolorbox
```

Package 93

lwarp-xfrac.sty

165 Xfrac

Pkg `xfrac` Supported by adding xfrac instances.

for HTML output:

```
1 \LWR@ProvidesPackagePass{xfrac}
```

font size In the user's document preamble, lwarp should be loaded after font-related setup. During HTML conversion, this font is used by lwarp to generate its initial PDF output containing HTML tags, later to be converted by `pdftotext` to a plain text file. While the text may be in any font which `pdftotext` can read, the math is directly converted into SVG images using this same user-selected font. `xfrac` below is set for the Latin Modern (lmr) font. If another font is used, it may be desirable to redefine `\xfracHTMLfontsize` with a different em size.

`\sfrac` [⟨*instance*⟩] {⟨*num*⟩} [⟨*sep*⟩] {⟨*denom*⟩}

A text-mode instance for the default font is provided below. The numerator and denominator formats are adjusted to encase everything in HTML tags. `\scalebox` is made null inside the numerator and denominator, since the HTML tags should not be scaled, and we do not want to introduce additional HTML tags for scaling.

In math mode, which will appear inside a `lateximage`, no adjustments are necessary.

for HTML & PRINT:

```
2 \begin{warpall}
```

User-redefinable macro which controls the font size of the fraction.

```
3 \newcommand*{\xfracHTMLfontsize}{.6em}
```

```
4 \end{warpall}
```

for HTML output:

```
5 \begin{warpHTML}
```

font size A span for a small font, used in the numerator and denominator:

```
6 \newcommand*{\LWR@htmlsmallfontstart}{%
7 \LWR@htmltagc{span style="font-size:\xfracHTMLfontsize"{}}%
8 \LWR@nestspan%
9 %
10 }
```

```

\newcommand*{\LWR@htmlsmallfontend}{%
\LWR@htmltagc{/span}%
\endLWR@nestspan%
}
```

\scalebox A nullified \scalebox command, to avoid introducing HTML scaling tags:

```
\NewDocumentCommand{\LWR@noscalebox}{m o m}{#3}
```

instances Instances of xfrac for various font choices:

Produce HTML tags for a small superscript numerator and a small (non-subscript) denominator.

Scaling is turned off so that pdftotext correctly reads the result.

```
\DeclareInstance{xfrac}{default}{text}{
numerator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart\textsuperscript{#1}\,\LWR@htmlsmallfontend},
denominator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart{}\,#1\LWR@htmlsmallfontend},
```

For pdftotext, do not scale the text:

```
scaling = false
}
\DeclareInstance{xfrac}{lmr}{text}{
numerator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart\textsuperscript{#1}\,\LWR@htmlsmallfontend},
denominator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart{}\,#1\LWR@htmlsmallfontend},
```

For pdftotext, do not scale the text:

```
scaling = false
}
\DeclareInstance{xfrac}{lmss}{text}{
numerator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart\textsuperscript{#1}\,\LWR@htmlsmallfontend},
denominator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart{}\,#1\LWR@htmlsmallfontend},
```

For pdftotext, do not scale the text:

```
scaling = false
}
\DeclareInstance{xfrac}{lmtt}{text}{
numerator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart\textsuperscript{#1}\,\LWR@htmlsmallfontend},
denominator-format = {%
\let\scalebox\LWR@noscalebox%
\LWR@htmlsmallfontstart{}\,#1\LWR@htmlsmallfontend},
```

For pdftotext, do not scale the text:

```
scaling = false
}

\end{warpHTML}
```

Change History and Index

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

A

D

E

F

G

H

M

N

O

P

Q

R

S

T

U

V

W

X

Z