emo·ji for all (LaTeX engines)

Robert Grimm

Version v0.2 (2023/04/01)

Abstract

Emo implements the `\emo{`⟨*emoji-name*⟩`}` command for including color emoji such as `\emo{desert-island}` for 🏝 or `\emo{parrot}` for 🦜 in your documents independent of LaTeX engine. The implementation uses the Noto color emoji font if the engine supports it and includes PDF graphics otherwise. The latter are automatically derived from Noto's SVG sources, so the visual appearance is very similar. The source repository is at `https://github.com/apparebit/emo`. Emo may come in particularly handy when dealing with academic publishers that provide only minimal support for non-Latin scripts (cough, ACM, cough).

Contents

1 Installation

The emo package is available through its source repository or through CTAN. Installation is fairly straightforward, though it does involve a lot more files than usual.

1. Start by extracting this package's files from `emo.dtx` by running:

   ```
   $ pdftex emo.dtx
   ```

   Do *not* use `tex`; it mangles the embedded `README.md`. `pdflatex` also extracts the files and then builds the documentation. Embedded files are `build.sh`, `emo.ins`, `emo.sty`, `emo.sty.ltxml`, and `README.md`. Extraction will overwrite existing files with the same name without asking.

2. Build the package documentation with change and symbol indices by running:

   ```
   $ source build.sh
   ```

   The shell script invokes `pdflatex emo.dtx` thrice and `makeindex` once each for the change and symbol indices to produce `emo.pdf`.

3. Get started reconfiguring supported emoji by running:

   ```
   $ python config/emo.py -h
   ```

   For more detailed instructions, see §3 below.

4. Put the following files somewhere LaTeX can find them. In a pinch, your current project's directory will do. However, emo's installation potentially comprises thousands of files. So, you probably want to use a dedicated directory and add that to the search path for LaTeX, e.g., by setting the `TEXINPUTS` environment variable.

   (a) `emo.sty` with the package implementation;

   (b) `emo.sty.ltxml` with the binding for LaTeXML;

   (c) `emo.def` with the emoji table;

   (d) `emo-lingchi.ttf` with the two glyphs for `\lingchi`;

   (e) `emo-graphics` with the fallback PDF graphics.

   TeX Live requires that each package's files have unique names. For that reason, the PDF graphics in the `emo-graphics` directory start with the `emo-` prefix as well.

When running on the LuaLaTeX engine, the emo package also uses the Noto color emoji (`NotoColorEmoji.ttf`) and Linux Libertine (`LinLibertine_R.otf`) fonts, with the latter used for rendering `\YHWH` only. Neither file is included with emo's distribution, since both of them are distributed with major TeX distributions already. If they are not included with your LaTeX distribution, you can find them on CTAN. The `emo-lingchi.ttf` font distributed with emo is a two glyph subset of `NotoSerifTC-Regular.otf`, i.e., the traditional Chinese version of Noto serif.

2 Usage

As usual, you declare your document's dependency on emo with `\usepackage{emo}`. In addition to the unadorned form, emo takes up to two options:

extra Also define the `\lingchi` and `\YHWH` macros, which produce 凌遲 and יהוה, respectively, and are documented below.

index Create an emoji index tagged `emo` with the `.edx` extension for the raw index and the `.end` extension for the processed index. This option relies on the `index` package, generates the raw `.edx` file, but does not build or use the processed index.

`\emo` An `\emo{`⟨*emoji-name*⟩`}` invocation expands to the named emoji. For LuaLaTeX, it uses the Noto color emoji font. For all other engines, it uses PDF graphics. That way, `\emo{desert-island}` results in 🏝 and `\emo{parrot}` results in 🦜.

Since LaTeX tends to produce a lot of command line noise about underfull boxes and loaded fonts, it's a easy to miss meaningful warnings. For that reason, `\emo` expands to an attention-seeking error message upon undefined emoji names. For example, `\emo{boo}` produces `Bad \emo{boo}`.

2.1 Emoji Names

With some exceptions, emo's names for emoji are automatically derived from their Unicode names, with letters converted to lowercase, punctuation such as commas, colons, quotes, and parentheses stripped, and interword spaces replaced by dashes. Furthermore, instead of the rather verbose `dark-skin-tone`, `medium-dark-skin-tone`, etc modifiers, emo uses the more succinct `darkest`, `darker`, `medium`, `lighter`, and `lightest`.

For some emoji names, emo goes further by hard-coding shorter names. Those names are listed in Table 1.

Emo's `emo.def` contains the names and codepoints of all currently supported emoji. Its distribution also includes the `emoji-test.txt` file, which is part of Unicode TR-51 and contains the names and codepoints of all *potentially* supported emoji, i.e., all emoji. It further organizes emoji into groups and subgroups, with the current (sub)group being the one named on the closest line above the emoji that starts with `# (sub)group:`. As described in the next section, the group and subgroup names can be used during configuration for concisely naming a large number of emoji.

2.2 Extras

`\lingchi` `\YHWH` The `\lingchi` and `\YHWH` macros take no arguments and produce 凌遲 and יהוה, respectively. They are only available if emo is used with the `extra` option. The former renders the Chinese term for "death by a thousand cuts." While originally an execution method, the term applies to surprisingly many software systems as well. The latter produces the Tetragrammaton, the Hebrew name for God. Observant Jews never utter what's written, not even in their thoughts,

Table 1: Exceptional emoji names

Transformed Unicode Name	Emo Replacement Name
a-button-blood-type	a-button
ab-button-blood-type	ab-button
b-button-blood-type	b-button
o-button-blood-type	o-button
bust-in-silhouette	bust
busts-in-silhouette	busts
flag-european-union	eu
globe-showing-americas	globe-americas
globe-showing-asia-australia	globe-asia-australia
globe-showing-europe-africa	globe-africa-europe
hear-no-evil-monkey	hear-no-evil
index-pointing-at-the-viewer	index-pointing-at-viewer
index-pointing-at-the-viewer-darkest	index-pointing-at-viewer-darkest
index-pointing-at-the-viewer-darker	index-pointing-at-viewer-darker
index-pointing-at-the-viewer-medium	index-pointing-at-viewer-medium
index-pointing-at-the-viewer-lighter	index-pointing-at-viewer-lighter
index-pointing-at-the-viewer-lightest	index-pointing-at-viewer-lightest
keycap-*	keycap-star
keycap-#	keycap-hash
keycap-0	keycap-zero
keycap-1	keycap-one
keycap-2	keycap-two
keycap-3	keycap-three
keycap-4	keycap-four
keycap-5	keycap-five
keycap-6	keycap-six
keycap-7	keycap-seven
keycap-8	keycap-eight
keycap-9	keycap-nine
keycap-10	keycap-ten
magnifying-glass-tilted-left	loupe-left
magnifying-glass-tilted-right	loupe-right
palm-down-hand	palm-down
palm-down-hand-darkest	palm-down-darkest
palm-down-hand-darker	palm-down-darker
palm-down-hand-medium	palm-down-medium
palm-down-hand-lighter	palm-down-lighter
palm-down-hand-lightest	palm-down-lightest
palm-up-hand	palm-up
palm-up-hand-darkest	palm-up-darkest
palm-up-hand-darker	palm-up-darker
palm-up-hand-medium	palm-up-medium
palm-up-hand-lighter	palm-up-lighter
palm-up-hand-lightest	palm-up-lightest
rolling-on-the-floor-laughing	rofl
see-no-evil-monkey	see-no-evil
speak-no-evil-monkey	speak-no-evil

substituting Adonai (“My Lord”), Elohim (“God”), or HaShem (“The Name”) instead. In my mind, that nicely mirrors the very incomprehensibility of יהוה. Both macros preserve a subsequent space as space, no backslash needed.

3 Configuration

Emo’s implementation is actually split over two files: `emo.sty` is extracted from `emo.dtx` and defines the substance of the package, its options, its helper macros, and the user-visible `\emo`, `\lingchi`, and `\YHWH` macros. Currently supported emoji are defined by the emoji table in the second file, `emo.def`. For every supported emoji, the file contains a command `\emo@emoji@`⟨*emoji-name*⟩ with the emoji’s codepoints as value.

Configuration automates the regeneration of the emoji table for arbitrary numbers of emoji. `config/emo.py` is the script and `config/emoji-test.txt` is the list of all emoji from the Unicode standard.

3.1 Running the Configuration Script

To update emo’s configuration, invoke the `config/emo.py` script:

```
$ python3 config/emo.py ⟨selector⟩ ⟨selector⟩ ...
```

Each selector may be:

- The literal `ALL` (case-sensitive) for *all* emoji.
- Name of a group in `emoji-test.txt` lowercased and with spaces replaced by dashes and ampersand `&` replaced by an `and`; e.g., `travel-and-places`.
- Name of a group, a double colon `::`, and name of a subgroup, again lowercased and with spaces replaced by dashes and `&` by an `and`; e.g., `travel-and-places::place-geographic`.
- The name of an emoji; e.g., `desert-island`.

For conjunctive group names, such as “Smileys & Emotion” (`emoji-test.txt`) or “smileys-and-emotion” (`emo.py`), the configuration script also accepts either of the two nouns as a shortcut, e.g., “smileys” or “emotion.”

For data safety, `emo.py` does not overwrite PDF graphics and hence can only *add* emoji to the configuration. To remove emoji, simply remove their PDF graphics from `emo-graphics` and then run `emo.py` without selector arguments, which updates the emoji table accordingly.

`emo.py` effectively treats `emoji-test.txt` as registry of all emoji and the filenames of PDF graphics in `emo-graphics` as emo’s current inventory. For all emoji named by selector arguments but not in the inventory, `emo.py` converts the SVG source graphic from the Noto color emoji sources to a PDF file and deletes the `/Page /Group` object from the the PDF again, since that object trips up `pdflatex`. And yeah, `emo.py` automatically downloads the Noto color emoji sources if necessary.

4 Copyright and Licensing

Since emo's distribution includes not only LaTeX code but also a substantial Python script, Unicode data about emoji, as well as graphics and fonts derived from Google's Noto project, a number of different licenses apply. All of them are OSI approved and non-copyleft:

- This package's LaTeX and also Perl code extracted from `emo.dtx` is © Copyright 2023 by Robert Grimm and has been released under the LPPL v1.3c or later.
- The `config/emo.py` script also is © Copyright 2023 by Robert Grimm but has been released under the Apache 2.0 license.
- The [config/emoji-test.txt] configuration file is a data file from Unicode TR-51 and hence subject to the Unicode License.
- The `emo-lingchi.ttf` font is a two-glyph subset of the traditional Chinese version of Google's Noto serif and hence subject to the SIL Open Font License v1.1.
- The PDF graphics in the `emo-graphics` directory are derived from the sources for Noto's color emoji and hence subject to the Apache 2.0 license.

5 Implementation

Let's get started with emo's implementation:

```
1 ⟨*package⟩
```

Except, the package implementation started near the top of the `emo.dtx` file, before the documentation preamble. We repeat it here for completeness:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{emo}
    [2023/04/01 v0.2 emo•ji for all (LaTeX engines)]
```

And no, I didn't repeat the version number, date, or package information. Check `emo.dtx`.

5.1 Package Options

`\ifemo@extra` `\ifemo@index` Emo's extra and index options are simple flags. So we declare a new conditional for each and, if `\usepackage` includes an option, toggle the conditional's state.

```
2 \newif\ifemo@extra\emo@extrafalse
3 \DeclareOption{extra}{\emo@extratrue}
4 \newif\ifemo@indexing\emo@indexingfalse
5 \DeclareOption{index}{\emo@indexingtrue}
6 \ProcessOptions\relax
```

5.2 Setup Including Dependencies

The dependency on `inputenc` effectively declares this file's encoding to be UTF-8. The XeTeX and LuaTeX engines already expect files to be encoded that way and hence ignore the declaration. However, pdfTeX supports other (legacy) encodings and needs to be told.

```
\RequirePackage[utf8]{inputenc}
```

`\ifemo@usefont` Define a third conditional for selecting the backend. Originally, I didn't bother with this, simply required `iftex`, and used `\ifluatex` throughout the package. But that unnecessarily complicates support for other LaTeX-like engines.

```
\newif\ifemo@usefont\emo@usefontfalse
\RequirePackage{iftex}
\ifluatex\emo@usefonttrue\fi
```

By now `\ifemo@usefont` should be correctly set up. So we use it for loading the right emoji-emitting backend, `fontspec` or `graphicx`:

```
\ifemo@usefont
\RequirePackage{fontspec}
\else
\RequirePackage{graphicx}
\fi
```

Emo requires `xcolor` for formatting highly visible error messages within the text. Always including another package that is only used when there are errors is not ideal. But when I tried calling `\RequirePackage` for `xcolor` from inside the error macro, it didn't work. Alternatively, I could make in-text errors optional.

```
\RequirePackage{xcolor}
```

Finally, emo's options also have dependencies, with `extra` requiring the `xspace` package and `index` requiring the `index` package:

```
\ifemo@extra
\RequirePackage{xspace}
\fi
\ifemo@indexing
\RequirePackage{index}
\fi
```

5.3 The Emoji Table

`\emo@emoji@<name>` For each emoji with a PDF graphic in the `emo-graphics` directory, a macro named `\emo@emoji@`⟨*emoji-name*⟩ expands to its Unicode sequence. With over 3,000 distinct emoji in Unicode 15, emo relies on a Python script for populating the graphics directory and writing the table to the `emo.def` file. Since the package code does not change after installation but the emoji table may very well change, they are kept separate for now. Alternatively, we could use DocStrip

to assemble the package file from three parts, the code from the previous sections, then the contents of the emoji table in `emo.def`, and then all subsequent code.

```
\input{emo.def}
```

5.4 Internal Macros

`\emo@error@fg` `\emo@error@bg` `\emo@error` Define two colors and a function that uses the two colors for formatting an attention-grabbing error message. If you use an invalid emoji name and overlook the warning in the console, you *will* notice the error messsage in the document thusly formatted.

```
\definecolor{emo@error@fg}{rgb}{1,1,1}
\definecolor{emo@error@bg}{rgb}{.6824,.0863,.0863}
\def\emo@error#1{%
    \colorbox{emo@error@bg}{%
        \textcolor{emo@error@fg}{%
            \textsf{Bad} \texttt{\textbackslash emo\{#1\}}%
        }%
    }%
}
```

`\emo@ifdef` Validate the emoji name given as first argument. The macro expands to the second argument if the name is valid and an error message otherwise. Its implementation relies on the `emo@emoji` table.

```
\def\emo@ifdef#1#2{%
    \ifcsname emo@emoji@#1\endcsname#2\else%
       \PackageWarning{emo}{Unknown emoji name in ‘\string\emo{#1}’}%
        \emo@error{#1}%
    \fi%
}
```

`\emo@index` If indexing is enabled, record the use of an emoji. Otherwise, do nothing.

```
\ifemo@indexing
\newindex{emo}{edx}{end}{Emoji Index}
\def\emo@index#1{\index[emo]{#1}}
\else
\def\emo@index#1{}
\fi
```

5.5 User Macros

`\emo` Emit the named color emoji. Both the font-based version for LuaTeX and the graphics-based fallback validate the emoji name and then invoke the `\emo@index` macro. But they differ in how they actually display the emoji. The LuaTeX version turns the emoji name into its Unicode sequence and wraps that

in a group that also uses the previously declared Noto color emoji font. The fallback version instead includes a suitably sized PDF graphic.

```
\ifemo@usefont
\newfontface\emo@font[Renderer=Harfbuzz]{NotoColorEmoji.ttf}
\newcommand\emo[1]{%
    \emo@ifdef{#1}{%
        \emo@index{#1}%
        {\emo@font\csname emo@emoji@#1\endcsname}%
    }%
}
\else
\newcommand\emo[1]{%
    \emo@ifdef{#1}{%
        \emo@index{#1}%
        \raisebox{-0.2ex}{%
            \includegraphics[height=1em]{emo-graphics/emo-#1}}%
    }%
}
\fi
```

`\lingchi` `\YHWH` The definitions for the optional `\lingchi` and `\YHWH` macros follow from that of `\emo`, except that (a) there are no arguments to validate and hence no equivalent to `\emo@ifdef`; (b) Hebrew is written right-to-left and hence `\YHWH` requires a `\textdir TRT`; (c) subsequent space should be preserved and hence both macros end with `\xspace`. While it would be nice to use Unicode inside the groups for font-using versions, this version is more robust given TeX pre-Unicode heritage.

```
\ifemo@extra
\ifemo@usefont
\newfontface\emo@chinese{emo-lingchi.ttf}
\newfontface\emo@hebrew{LinLibertine_R.otf}
\newcommand\lingchi{%
    \emo@index{lingchi}%
    \begingroup\emo@chinese \char"51CC\char"9072\endgroup%
    \xspace}
\newcommand\YHWH{%
    \emo@index{YHWH}%
    \begingroup%
        \textdir TRT%
        \emo@hebrew \char"5D9\char"5D4\char"5D5\char"5D4%
    \endgroup%
    \xspace}
\else
\newcommand\lingchi{%
    \emo@index{lingchi}%
    \raisebox{-0.2ex}{%
        \includegraphics[height=1em]{emo-graphics/emo-lingchi}}%
    \xspace}
```

```
\newcommand\YHWH{%
    \emo@index{YHWH}%
    \raisebox{-0.2ex}{%
        \includegraphics[height=1em]{emo-graphics/emo-YHWH}}%
    \xspace}
\fi
\fi
```

Et voilà. That's it!

⟨/package⟩

6 LaTeXML Binding

To support conversion from LaTeX to HTML, emo includes a so-called binding for LaTeXML. It effectively is a (much simplified) re-implementation of emo's core functionality, only written in Perl against LaTeXML's API. The binding ignores the index option and does not perform error checking on emoji names. If either is important to you, please compile the document with LaTeX first. Furthermore, the binding emits necessary Unicode codepoints only, without font annotations. If you want to specify fonts, please use a CSS fontstack.

Asking package authors to reimplement their packages for LaTeXML seems unreasonable to me. It leads to code duplication and places the maintenance burden on package authors. Yet, right after announcing emo, the question of LaTeXML support came up. LaTeXML includes the `latexml.sty` package, which defines `\iflatexml`. I would have used that command to make the three-line change to `emo.sty` necessary for LaTeX support, except `latexml.sty` contains lots of other stuff that isn't needed. Always loading that only to detect LaTeXML seems like a bad idea. Since reimplementing `\iflatexml` would require a binding anyways, I just wrote a minimal binding. As I said, LaTeXML's approach is broken.

With that out of the way, let's get started:

⟨*latexml-binding⟩

The binding starts with an explicit preamble because `docstrip` does not alllow for a redefinition of the starting characters of a line comment. It is followed by the Perl dependencies.

```
## emo's LaTeXML binding.
## (C) 2023 by Robert Grimm.
## Released under LPPL v1.3c or later.
use strict;
use warnings;
use LaTeXML::Package;
```

`\ifemo@extra` Next, we use raw TeX to declare the LaTeX package and define the `emo@extra` conditional. There is no need to define the `emo@indexing` conditional, since it

corresponds to the unsupported index option.

```
RawTeX(<<'EOTeX');
\ProvidesPackage{emo}
    [2023/03/21 v0.2a1 emo•ji for all (LaTeX engines)]
\newif\ifemo@extra\emo@extrafalse
EOTeX
```

Option prcessing is almost trivial:

```
DeclareOption('extra', '\emo@extratrue');
DeclareOption('index', '');
ProcessOptions();
```

`\emo@emoji@<name>` `\emo` Just like the actual package implementation, the binding reads the emoji table from `emo.def`. Similar to the actual implementation of the `\emo` macro when running under LuaLaTeX, the binding expands the named entry from the emoji table, producing the emoji's Unicode codepoints.

```
InputDefinitions('emo', type => 'def', noltxml => 1);
DefMacro('\emo{}', '\csname emo@emoji@#1\endcsname');
```

`\lingchi` `\YHWH` If the `emo@extra` conditional is enabled, require the `xspace` package and then provide minimal re-definitions of the `\lingchi` and `\YHWH` macros. Both simply expand to the necessary Unicode codepoints.

```
if (IfCondition(T_CS('\ifemo@extra'))) {
    RequirePackage('xspace');
    DefMacro('\lingchi', "\x{51cc}\x{9072}\\xspace");
    DefMacro('\YHWH', "\x{05D9}\x{05D4}\x{05D5}\x{05D4}\\xspace");
}
```

That's it for the binding, too.

⟨/latexml-binding⟩

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.