

# emo•ji for all (LaTeX engines)

Robert Grimm

Version v0.5 (2023/05/01)

## 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. It also supports conversion to HTML with either LaTeXML or TeX4ht. Next, PDF graphics 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

<b>1</b>	<b>Installation</b>	<b>2</b>
<b>2</b>	<b>Usage</b>	<b>3</b>
2.1	One Main Macro . . . . .	3
2.1.1	Emoji Names . . . . .	3
2.2	Two Extra Macros . . . . .	5
2.3	Conversion to HTML . . . . .	5
<b>3</b>	<b>Configuration</b>	<b>5</b>
3.1	Running the Configuration Script . . . . .	5
<b>4</b>	<b>Copyright and Licensing</b>	<b>6</b>
<b>5</b>	<b>Implementation</b>	<b>7</b>
5.1	Package Options . . . . .	7
5.2	Package Dependencies and Backend Selection . . . . .	7
5.3	The Emoji Table . . . . .	8
5.4	Internal Macros . . . . .	9
5.5	User Macros and Hooks . . . . .	9
5.5.1	A Skeleton of Hooks . . . . .	10
5.5.2	The User Macros . . . . .	11
5.5.3	Activating the Hooks . . . . .	11

6	LaTeXML Binding	12
7	Testing Emo	14
7.1	Test Class . . . . .	14
7.2	Test Runner and Report . . . . .	17
7.3	Simple Test Document . . . . .	18

# 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`, `emo-test.cls`, `canary.tex`, `demo.tex`, 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 serves three purposes: (1) To test `emo` across engines, (2) to illustrate conversion to HTML, and (3) to generate the user documentation. In more detail it processes the tests in `canary.tex` and `emo-test.cls` with `pdflatex`, `xelatex`, and `lualatex`, combining the test results into `canary.pdf`. It also processes `demo.tex` with LaTeXML and TeX4ht to generate `demo-latexml.html` and `demo-tex4ht.html`, respectively. Finally, it processes `emo.dtx` with `pdflatex` and `makeindex` to produce documentation in `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 יהודה.

**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.

### 2.1 One Main Macro

`\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}`.

#### 2.1.1 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. Emo's distribution also includes the `emoji-test.txt` file, which is part of

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-america	globe-america
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

[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 Two Extra Macros

`\lingchi` The `\lingchi` and `\YHWH` macros take no arguments and produce 凌遲 and יהוה. `\YHWH` 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, 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.

## 2.3 Conversion to HTML

Emo supports conversion to HTML with either [LaTeXML](#) or [TeX4ht](#). LaTeXML support is implemented by a separate “binding” against LaTeXML’s Perl API. I chronicled my exploration of suitable options leading to that less than ideal choice in a [GitHub issue](#). TeX4ht support is implemented by the `emo` package itself. It requires processing with LuaLaTeX e.g., by passing `-l` to the `make4ht` tool.

# 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 file-names 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 on emo's implementation:

```
1 ⟨*package⟩
```

Except, that implementation started near the top of this `emo.dtx` file, well before the documentation's preamble. For completeness, here is the package declaration again:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{emo}
[2023/05/01 v0.5 emo·ji for all (LaTeX engines)]
```

Unfortunately, emo's version number appears in triplicate in this file, which makes cutting a release a little more tricky than it should be. Alas, the above, faux package declaration is not one of those repeat offenders. Really.

### 5.1 Package Options

`\ifemo@extra` Define a conditional flag for each package option.

```
\ifemo@index
\ifemo@debug 2 \newif\ifemo@extra\emo@extrafalse
3 \newif\ifemo@index\emo@indexfalse
4 \newif\ifemo@debug\emo@debugfalse
```

Wire each conditional to the package option and then process them all.

```
5 \DeclareOption{extra}{\emo@extratrue}
6 \DeclareOption{index}{\emo@indextrue}
7 \DeclareOption{debug}{\emo@debugtrue}
8 \ProcessOptions\relax
```

### 5.2 Package Dependencies and Backend Selection

Require `inputenc` to declare this file's character encoding as UTF-8. XeTeX and LuaTeX already use that encoding by default and hence this line is redundant. But pdfTeX originates from darker, pre-Unicode times and thus needs to be told. Though thankfully, it does support the encoding by now.

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

`\ifemo@use@unicode` Define a conditional flag for each major backend configuration. They are *not* orthogonal: `\ifemo@use@unicode` and `\ifemo@use@pdf` are mutually exclusive and determine whether emo generates Unicode text or PDF graphics. For `\ifemo@use@font` to be enabled, `\ifemo@use@unicode` must already be enabled as well.

```
10 \newif\ifemo@use@unicode
11 \newif\ifemo@use@font
12 \newif\ifemo@use@pdf
```

Inspect the current engine via `iftex` and configure the just defined backend flags.

```
13 \RequirePackage{iftex}
14 \ifdefined\HCode
15     \emo@use@unicodetrue
16 \else
17 \ifluatex
18     \emo@use@unicodetrue
19     \emo@use@fonttrue
20 \else
21     \emo@use@pdftrue
22 \fi
23 \fi
```

Require `xcolor` for formatting highly visible error messages within the generated document. Always including another package that is only used when there are errors is not ideal. But I couldn't get on-demand package loading to work. So we have to eagerly require the package.

```
24 \RequirePackage{xcolor}
```

Each of the three remaining requirements either depends on a package option or on the backend.

```
25 \ifemo@indexing
26     \RequirePackage{index}
27 \fi
28 \ifemo@use@font
29     \RequirePackage{fontspec}
30 \fi
31 \ifemo@use@pdf
32     \RequirePackage{graphicx}
33 \fi
```

### 5.3 The Emoji Table

`\emo@emoji@name` The emoji table defines an entry for each enabled emoji and, if the extra option is present, also for the two extra macros. Each entry maps the name of the emoji to its Unicode sequence. For example, the control sequence `\emo@emoji@parrot` has 🦜 as its value. Since there are over 3,000 distinct emoji in Unicode 15, `emo` does not support all of them out of the box. Rather it relies on a Python script for populating the `emo-graphics` directory with the necessary PDF files and writing the table to `emo.def`. This nicely keeps code that may change after installation, the emoji table, from code that doesn't, the package implementation itself.

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



## 5.4 Internal Macros

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

```
35 \definecolor{emo@error@fg}{rgb}{1,1,1}
36 \definecolor{emo@error@bg}{rgb}{.6824,.0863,.0863}
37 \def\emo@error#1{%
38     \colorbox{emo@error@bg}{%
39         \textcolor{emo@error@fg}{%
40             \textsf{Bad} \texttt{\textbackslash emo\{#1\}}}%
41         }%
42     }%
43 }
```

`\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 im-  
plementation relies on the emoji table as the primary source of truth.

```
44 \def\emo@ifdef#1#2{%
45     \ifcsname emo@emoji@#1\endcsname#2\else%
46         \PackageWarning{emo}{Unknown emoji name in ‘\string\emo{#1}’}%
47         \emo@error{#1}%
48     \fi%
49 }
```

## 5.5 User Macros and Hooks

Between versions 0.1 and 0.4, the implementation evolved considerably. In ver-  
sions 0.1 and 0.2, I focused mostly on getting `emo` working across all three ma-  
jor LaTeX engines. I also tried to manage variability by introducing a number  
of internal macros, with some of them defined at least twice to account for  
features being present or not. Hence it wasn't surprising when I also ended up  
implementing each of the three user macros `\emo`, `\lingchi`, and `\YHWH` twice,  
with one version selecting an appropriate font and then emitting Unicode and  
the second, fallback version using PDF graphics.

For version 0.3, I added support for TeX4ht as the second option for convert-  
ing to HTML. That meant adding a third backend, which emits Unicode only,  
*without* font selection. I also integrated the Unicode code points for `\lingchi`  
and `\YHWH` into the emoji table and abstracted over the three macros' content  
versus surrounding grouping, indexing, and error checking logic. As a result,  
the implementation appeared better organized and simpler despite support-  
ing more feature variability.

With version 0.4, I introduced the `debug` option to `emo`, which increased imple-  
mentation complexity again. Now, each macro of the three macros required an  
optional wrapper with the framed box visualizing the emoji's bounding box.  
Just as my discontent with the implementation grew, my work on a bespoke

testing class also significantly deepened my TeX-fu and so I started looking out for a more ergonomic approach to emo’s implementation as well. I now believe I found that approach thanks to [LaTeX hooks](#).

### 5.5.1 A Skeleton of Hooks

emo/render/before To define emo’s hooks, notice that, in addition to entity name, Unicode code points, and PDF graphics, the only other major source of variability is the font selection or lack thereof. In fact, each of the three macros necessarily must have its own font selection logic and hence dedicated hook, since each macro produces text in a different “language” and writing system. Otherwise, all three macros have the exact same structure and hooks: A group combines the emo/render/emoji, emo/render/chinese, or emo/render/hebrew hook for font selection with the emo/render/content hook for the actual text. The emo/render/before and emo/render/after hooks enable aspect-oriented functionality, such as indexing or logging.

```
50 \NewHook{emo/render/before}
51 \NewHook{emo/render/emoji}
52 \ifemo@extra
53 \NewHook{emo/render/chinese}
54 \NewHook{emo/render/hebrew}
55 \fi
56 \NewHook{emo/render/content}
57 \NewReversedHook{emo/render/after}
```

\emo@makecommand With the hooks defined, it’s fairly straight-forward to write a macro that invokes those hooks in the prescribed order. Since the font selection hook differs between the three macros, it is the only argument to \emo@makecommand. Hooks have no arguments beyond their labels. So to make the current key and, after the group has been rendered, the corresponding value available in hooks, the macro below introduces a naming convention, with \emo@key providing the name across all hooks and \emo@value providing the rendered content to the emo/render/after hook. The latter can modify the macro’s result by redefining \emo@value. In fact, as you can see shortly, that’s just how the debug package option is implemented.

```
58 \def\emo@makecommand#1{%
59   \noexpand\def\noexpand\emo@key{##1}%
60   \noexpand\emo@ifdef{##1}{%
61     \noexpand\UseHook{emo/render/before}%
62     \noexpand\def\noexpand\emo@value{%
63       \noexpand\begin{group}%
64       \noexpand#1%
65       \noexpand\UseHook{emo/render/content}%
66       \noexpand\end{group}%
67     }%
68     \noexpand\UseHook{emo/render/after}%
69     \noexpand\emo@value%
70   }%
```

71 }

### 5.5.2 The User Macros

`\emo` Render the named emoji either as Unicode or as a PDF graphic. To do so, instantiate the skeleton with the font selection hook for emoji. We eagerly evaluate the instantiation of the skeleton with `\edef`, so that it forms a coherent macro body. At the same time, to ensure that eager evaluation doesn't fail, the definition of `\emo@makecommand` liberally uses `\noexpand`.

```
72 \edef\emo#1{\emo@makecommand{\UseHook{emo/render/emoji}}}
```

`\lingchi` The definitions for the two extra macros aren't much more involved. Though `\YHWH` they do require a helper macro each, since the user-facing macros do not accept arguments.

```
73 \ifemo@extra
74 \edef\@lingchi#1{\emo@makecommand{\UseHook{emo/render/chinese}}}
75 \edef\@YHWH#1{\emo@makecommand{\UseHook{emo/render/hebrew}}}
76 \def\lingchi{\@lingchi{lingchi}}
77 \def\YHWH{\@YHWH{YHWH}}
78 \fi
```

### 5.5.3 Activating the Hooks

So far, the three user-facing macros have little impact beyond generating empty groups. We change that by activating the hooks as required by the current backend configuration and package options. The neat part about this approach is that we don't need to do anything if an option is disabled.

If the `index` option is enabled, create the necessary index and emit entries through the `emo/render/before` hook.

```
79 \ifemo@index
80 \newindex{emo}{edx}{end}{Emoji Index}
81 \def\emo@index#1{\index[emo]{#1}}
82 \AddToHook{emo/render/before}{\expandafter\emo@index\expandafter{\emo@key}}
83 \fi
```

Next, if the backend uses fonts, set up font selection in the corresponding hooks. We only activate the `emo/render/chinese` and `emo/render/hebrew` hooks if the `extra` package option is enabled. Note that since Hebrew is written right-to-left, the font selection hook also set the text direction. This is safe to do because font selection and content are always enclosed in a group.

```
84 \ifemo@use@font
85 \newfontface\emo@font@emoji[Renderer=Harfbuzz]{NotoColorEmoji.ttf}
86 \AddToHook{emo/render/emoji}{\emo@font@emoji}
87 \ifemo@extra
88 \newfontface\emo@font@chinese{emo-lingchi.ttf}
89 \newfontface\emo@font@hebrew{LinLibertine_R.otf}
```

```

90 \AddToHook{emo/render/chinese}{\emo@font@chinese}
91 \AddToHook{emo/render/hebrew}{\emo@font@hebrew\textdir TRT}
92 \fi
93 \fi

```

Next, we activate the `emo/render/content` hook to either emit Unicode code points or PDF graphics.

```

94 \ifemo@use@unicode
95 \AddToHook{emo/render/content}{\csname emo@emoji@\emo@key\endcsname}
96 \fi
97 \ifemo@use@pdf
98 \AddToHook{emo/render/content}{%
99   \raisebox{-0.2ex}{\includegraphics[height=1em]{emo-graphics/emo-\emo@key}}%
100 }
101 \fi

```

Finally, if the `debug` package option is enabled, we wrap the rendered content in a frame.

```

102 \ifemo@debug
103 \AddToHook{emo/render/after}{%
104   \let\emo@realvalue\emo@value%
105   \def\emo@value{\fbox{\emo@realvalue}}%
106 }
107 \fi

```

Et voilà. That's it!

```

108 \end{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 to support LaTeXML, except `latexml.sty` contains lots of other stuff that isn't needed. Always loading lots of macros only to detect LaTeXML slows down compilation and wastes memory. 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:

```
1⟨*latexml-binding⟩
```

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

```
2## emo's LaTeXML binding.
3## (C) 2023 by Robert Grimm.
4## Released under LPPL v1.3c or later.
5use strict;
6use warnings;
7use 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.

```
8RawTeX(<<'EOTeX');
9\ProvidesPackage{emo}
10    [2023/05/01 v0.5 emo·ji for all (LaTeX engines)]
11\newif\ifemo@extra\emo@extrafalse
12EOTeX
```

Option processing is almost trivial:

```
13DeclareOption('extra', '\emo@extratrue');
14DeclareOption('index', '');
15DeclareOption('debug', '');
16ProcessOptions();
```

`\emo@emoji@name` Just like the actual package implementation, the LaTeXML binding reads the `\emo 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.

```
17InputDefinitions('emo', type => 'def', noltxml => 1);
18DefMacro('\emo{', '\csname emo@emoji@#1\endcsname');
```

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

```
19if (IfCondition(T_CS('\ifemo@extra')) {
20    DefMacro('\lingchi', "\x{51cc}\x{9072}");
21    DefMacro('\YHWH', "\x{05D9}\x{05D4}\x{05D5}\x{05D4}");
22}
```

That's it for the binding, too.

```
23⟨/latexml-binding⟩
```

## 7 Testing Emo

As emo's tagline so loudly proclaims, this package is intended to enable emoji across all major LaTeX engines. That requires testing across all major LaTeX engines. The necessary macros and documents follow.

Alas, we first start an always false conditional to prevent execution of this code while generating the documentation.

```
24 \scaffold
25 \iffalse
26 \scaffold
```

### 7.1 Test Class

```
1 \testing
```

The `emo-test` class contains all the code for testing emo.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesClass{emo-test}[2023/05/01 v0.5 Testing emo]
```

`emo-test` builds on `standalone` so that the resulting PDF has just the size of the output (plus some small margin). Its verbose option results in additional output from `\checkwidth` to help debug failing tests. All other options are passed down to the `standalone` class. Consider the `lua-visual-debug` package for verbose mode under LuaTeX.

```
4 \newif\ifemo@verbose@canary\emo@verbose@canaryfalse
5 \DeclareOption{verbose}{\emo@verbose@canarytrue}
6 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{standalone}}
7 \ProcessOptions\relax
8 \LoadClass[border=10pt, varwidth=6in]{standalone}
```

We require `emo` since it is being tested, `iftex` for determining the LaTeX engine, and `xcolor` for appearances sake. This class reuses several of `emo`'s internal macros, including `\ifemo@use@unicode`, `\ifemo@use@font`, and `\ifemo@use@pdf` for backend selection as well as `\emo@font@emoji`, `\emo@font@chinese`, and `\emo@font@hebrew` for font selection.

```
9 \RequirePackage[extra]{emo}
10 \RequirePackage{iftex}
11 \RequirePackage{xcolor}
```

When requiring `Libertinus` and `Inconsolata`, relative order matters and does so depending on LaTeX engine.

```
12 \iftutex
13 \RequirePackage{fontspec}
14 \RequirePackage{libertinus}
15 \setmonofont{inconsolata}
16 \else
17 \RequirePackage{libertinus}
```

```

18 \RequirePackage{inconsolata}
19 \fi

```

\enginename I couldn't find an existing macro that provides the desired functionality, so we gotta round up the usual suspects...

```

20 \ifxetex
21 \def\enginename{XeTeX}
22 \else
23 \ifluatex
24 \def\enginename{LuaTeX}
25 \else
26 \ifpdfTEX
27 \def\enginename{pdfTeX}
28 \else
29 \def\enginename{unknown engine}
30 \fi
31 \fi
32 \fi

```

emo@canary@frameinner Define frame and background colors for boundary boxes of sample text.

emo@canary@frameouter

```

emo@canary@background 33 \definecolor{emo@canary@frameinner}{HTML}{636366}
34 \definecolor{emo@canary@frameouter}{HTML}{48484A}
35 \definecolor{emo@canary@background}{HTML}{E5E5EA}

```

Adjust settings for \fcolorbox so that it serves as bounding box.

```

36 \setlength{\fboxrule}{0.5pt}
37 \setlength{\fboxsep}{0pt}

```

\emo@nobox The sample text may or may not show bounding boxes for words and emoji.

\emo@wordbox It always shows the bounding box for the entire line.

\emo@linebox

```

38 \newcommand\emo@nobox[1]{#1}
39 \newcommand\emo@wordbox[1]{%
40   \fcolorbox{emo@canary@frameinner}{white}{#1}}
41 \newcommand\emo@linebox[1]{%
42   \fcolorbox{emo@canary@frameouter}{emo@canary@background}{#1}}

```

\sampletext Show a single line of text that makes use of emo's three macros. To help identify incorrect font metrics, spurious whitespace, and other issues, show the line's bounding box and, for the starred version, the bounding boxes for words and emoji, too.

```

43 \def\@sampletext#1{%
44   \emo@linebox{%
45     #1{It's} #1{\lingchi}:
46     #1{Please}, #1{\YHWH}, #1{have} #1{mercy}
47     #1{\emo{pleading-face}}!%
48   }%

```

```

49     \vspace{1ex}%
50 }
51 \newcommand*\sampletext{%
52     \ifstar{\@sampletext{\emo@wordbox}}{\@sampletext{\emo@nobox}}%
53 }

```

`\emo@canary@actual` Validating emo’s macros turned out to be a bit trickier than I had expected. The obvious approach, fully expanding the macros and then comparing the result, doesn’t work. TeX does support eager expansion via, for example, `\expandafter` and `\edef`. But it does so only for macros that expand to text but not boxes, graphics, and so on.

Instead, we need to take a sneakier approach: Generate a box with the macro invocation and another box with the expected result and then compare the widths of the two boxes. While that is an incomplete comparison and hence cannot detect all bugs, it *can* detect any bug where the macro’s output is shorter or longer than expected. That conveniently includes whitespace, which is one of the bigger dangers for regressions.

We get started on that testing strategy by defining two box registers.

```

54 \newsavebox{\emo@canary@actual}
55 \newsavebox{\emo@canary@expected}

```

`\checkwidth` Before version 0.5, `\lingchi` and `\YHWH` relied on `xspace` to avoid any trailing backslashes. Now that all three user macros share the same code, implementing that feature would require another hook. So I dropped it. Alas, before version 0.5, the presence of `xspace` meant that both macros were context-sensitive. To ensure meaningful test results, `\checkwidth` fixed the context by following macro output with a period. The period remains even if `xspace` does not.

Even though `\checkwidth` only tests three macros and two of them take no arguments, the test macro still needs five arguments to cover all variability:

1. name of macro being tested;
2. macro invocation being tested;
3. name of font variable used in LuaLaTeX’s output;
4. Unicode code sequence in LuaLaTeX’s output;
5. file name for fallback PDF graphic without “emo-” prefix.

The third and fourth arguments are separate because the font variable only exists when running under LuaLaTeX.

```

56 \newcommand\checkwidth[5]{%
57     \sbox\emo@canary@actual{#2.}%
58     \ifemo@use@font%
59         \sbox\emo@canary@expected{%
60             \begingroup\csname emo@font@#3\endcsname #4\endgroup.}%
61     \else%

```



```

62 \ifemo@use@unicode%
63 \sbox\emo@canary@expected{\begingroup #4\endgroup.}%
64 \else%
65 \sbox\emo@canary@expected{%
66 \raisebox{-0.2ex}{%
67 \includegraphics[height=1em]{emo-graphics/emo-#5}}.}%
68 \fi%
69 \fi%
70 \ifemo@verbose@canary%
71 \fbox{\usebox{\emo@canary@actual}}\space%
72 \fbox{\usebox{\emo@canary@expected}}\space%
73 \fi%
74 \def\macroname{\texttt{\char'\#1}}%
75 \ifdim\wd\emo@canary@actual=\wd\emo@canary@expected%
76 \mbox{\macroname{} \emo{check-mark-button}}%
77 \else%
78 \edef\emo@actual{\the\wd\emo@canary@actual}%
79 \edef\emo@expected{\the\wd\emo@canary@expected}%
80 \mbox{\macroname{} \emo{cross-mark} \emo@actual{} \emo@expected}%
81 \fi%
82 }

```

Use small gap between paragraphs instead of indentation.

```

83 \setlength\parindent{0pt}
84 \setlength{\parskip}{1ex}



```

That's it for emo's testing class.

```
85 </testing>
```

## 7.2 Test Runner and Report

```
1 <*canary>
```

Each test report identifies the LaTeX engine and then shows the results of the width tests for emo's three user-visible macros. If a test passes, the output only contains the macro name and a  check mark. If a test fails, the output contains the macro name, the divergent box widths, and a  cross mark. In the latter case, the width test results spill into the next line (at the least).

```

2 \documentclass{emo-test}
3 \begin{document}
4 \Huge
5 \enginename: {\Large Width of
6 \checkwidth{emo}{\emo{robot}}{emoji}{\char"1F916}{robot},
7 \checkwidth{lingchi}{\lingchi}{chinese}{\char"51CC\char"9072}{lingchi},
8 \checkwidth{YHWH}{\YHWH}{hebrew}{%
9 \csname textdir\endcsname TRT\char"5D9\char"5D4\char"5D5\char"5D4}{YHWH}}
10 \vspace{1ex}\par

```

Next is the sample text, first with and then without boundary boxes for words and emoji.

```
11 \sampletext*\par\sampletext
12 \end{document}
```

That’s it for the tests and report.

```
13 \</canary>
```

7.3 Simple Test Document

```
1 \<*oneliner>
```

Not much to see here besides one line of content.s

```
2 \documentclass{emo-test}
3 \begin{document}
4 \Huge\sampletext
5 \end{document}
```

It’s a wrap 🥳

```
6 \</oneliner>
```

Change History

0.1	symbolic names . . . . .	7
General: Make initial release . . . . .	1	\lingchi: Build on \emo by default . . 11
0.2	\YHWH: Build on \emo by default . . .	11
General: Add LaTeXML binding for	0.4	General: Automate testing across
conversion to HTML . . . . .	12	engines with canary.tex . . . . . 1
Prefix font and graphic files		Introduce a simple unit testing
with “emo-” . . . . .	1	framework . . . . . 14
Support pdftex for extracting		\ifemo@debug: Add debug option for
emo.dtx . . . . .	1	drawing boundary boxes . . . . . 7
0.3		0.5
General: Support TeX4ht for		General: Use LaTeX hooks for
conversion to HTML . . . . .	1	handling engines and options . . 1
\emo@emoji@name: Include \lingchi		\lingchi: Drop use of trailing
and \YHWH as necessary . . . . .	8	xspace . . . . . 11
\ifemo@use@pdf: Make backend		\YHWH: Drop use of trailing xspace . 11
support scalable through		

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.

<b>Symbols</b>	\@ifstar . . . . .	52
\@YHWH . . . . .	75, 77	

\@lingchi .....	74, 76
\@sampletext .....	43, 52
<b>A</b>	
\AddToHook ...	82, 86, 90, 91, 95, 98, 103
<b>B</b>	
\begin .....	3, 3
<b>C</b>	
\checkwidth .....	6, 7, 8, 56
\colorbox .....	38
\CurrentOption .....	6
<b>D</b>	
\DeclareOption .....	5, 5, 6, 6, 7
\definecolor .....	33, 34, 35, 35, 36
\documentclass .....	2, 2
<b>E</b>	
\edef .....	72, 74, 75, 78, 79
\emo .....	6, 17, 46, 47, 72, 76, 80
\emo/render/after .....	50
\emo/render/before .....	50
\emo/render/chinese .....	50
\emo/render/content .....	50
\emo/render/emoji .....	50
\emo/render/hebrew .....	50
\emo@actual .....	78, 80
\emo@canary@actual ...	54, 57, 71, 75, 78
\emo@canary@background .....	33
\emo@canary@expected .....	54, 59, 63, 65, 72, 75, 79
\emo@canary@frameinner .....	33
\emo@canary@frameouter .....	33
\emo@debugfalse .....	4
\emo@debugtrue .....	7
\emo@emoji@name .....	17, 34
\emo@error .....	35, 47
\emo@error@bg .....	35
\emo@error@fg .....	35
\emo@expected .....	79, 80
\emo@extrafalse .....	2, 11
\emo@extratrue .....	5, 13
\emo@font@chinese .....	88, 90
\emo@font@emoji .....	85, 86
\emo@font@hebrew .....	89, 91
\emo@ifdef .....	44, 60
\emo@index .....	81, 82
\emo@indexingfalse .....	3
\emo@indexingtrue .....	6
\emo@key .....	58, 82, 95, 99
\emo@linebox .....	38, 44
\emo@makecommand .....	58, 72, 74, 75
\emo@nobox .....	38, 52
\emo@realvalue .....	104, 105
\emo@use@fonttrue .....	19
\emo@use@pdftrue .....	21
\emo@use@unicodetrue .....	15, 18
\emo@value .....	58, 104, 105
\emo@verbose@canaryfalse .....	4
\emo@verbose@canarytrue .....	5
\emo@wordbox .....	38, 52
\end .....	5, 12
\enginename .....	5, 20
<b>F</b>	
\fbox .....	71, 72, 105
\fboxrule .....	36
\fboxsep .....	37
\fcolorbox .....	40, 42
<b>H</b>	
\HCode .....	14
\Huge .....	4, 4
<b>I</b>	
\ifdefined .....	14
\ifdim .....	75
\ifemo@debug .....	2, 102
\ifemo@extra .....	2, 8, 19, 52, 73, 87
\ifemo@index .....	2
\ifemo@indexing .....	3, 25, 79
\ifemo@use@font .....	10, 28, 58, 84
\ifemo@use@pdf .....	10, 31, 97
\ifemo@use@unicode .....	10, 62, 94
\ifemo@verbose@canary .....	4, 70
\iffalse .....	25
\ifpdfTeX .....	26
\iftutex .....	12
\ifxetex .....	20
\includegraphics .....	67, 99
\input .....	34
<b>L</b>	
\Large .....	5
\let .....	104
\lingchi .....	7, 19, 45, 73
\LoadClass .....	8
<b>M</b>	
\macroname .....	74, 76, 80
\mbox .....	76, 80
<b>N</b>	
\NeedsTeXFormat .....	2
\newfontface .....	85, 88, 89
\NewHook .....	50, 51, 53, 54, 56
\NewReversedHook .....	57
\newsavebox .....	54, 55

<code>\noexpand</code> .....	59, 60, 61, 62, 63, 64, 65, 66, 68, 69	<code>\setlength</code> .....	36, 37, 83, 84
		<code>\setmonofont</code> .....	15
		<code>\space</code> .....	71, 72
<b>P</b>			
<code>\PackageWarning</code> .....	46	<b>T</b>	
<code>\par</code> .....	10, 11	<code>\textcolor</code> .....	39
<code>\parindent</code> .....	83	<code>\textdir</code> .....	91
<code>\parskip</code> .....	84	<code>\the</code> .....	78, 79
<code>\PassOptionsToClass</code> .....	6		
<code>\ProcessOptions</code> .....	7, 8	<b>U</b>	
<code>\ProvidesClass</code> .....	3	<code>\usebox</code> .....	71, 72
<code>\ProvidesPackage</code> .....	9	<code>\UseHook</code> .....	61, 65, 68, 72, 74, 75
<b>R</b>			
<code>\raisebox</code> .....	66, 99	<b>V</b>	
<code>\RequirePackage</code> .....	9, 9, 10, 11, 13, 13, 14, 17, 18, 24, 26, 29, 32	<code>\vspace</code> .....	10, 49
<b>S</b>			
<code>\sampletext</code> .....	4, 11, <u>43</u>	<b>W</b>	
<code>\sbox</code> .....	57, 59, 63, <u>65</u>	<code>\wd</code> .....	75, 78, 79
<b>Y</b>			
		<code>\YHWH</code> .....	8, <u>19</u> , 46, <u>73</u>