

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

1	Installation	2
2	Usage	3
2.1	One Main Macro	3
2.1.1	Emoji Names	3
2.2	Two Extra Macros	5
2.3	Conversion to HTML	5
3	Configuration	5
3.1	Running the Configuration Script	5
4	Copyright and Licensing	6
5	Implementation	7
5.1	Package Options	7
5.2	Setup Including Dependencies	7
5.3	The Emoji Table	8
5.4	Internal Macros	9
5.5	User Macros	10
6	LaTeXML Binding	11

7	Testing Emo	13
7.1	Test Class	13
7.2	Test Runner and Report	16
7.3	Simple Test Document	16

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 יהוה.

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 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}
[2023/05/01 v0.5 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` Emo's extra and index options are simple flags. So is the only incompletely documented `debug` option. We declare a new conditional for each and, if `\ifemo@debug` `\usepackage` includes an option, toggle the conditional's state.

```
2 \newif\ifemo@extra\emo@extrafalse
3 \DeclareOption{extra}{\emo@extratrue}
4 \newif\ifemo@index\emo@indexfalse
5 \DeclareOption{index}{\emo@indextrue}
6 \newif\ifemo@debug\emo@debugfalse
7 \DeclareOption{debug}{\emo@debugtrue}
8 \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.

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

`\emo@use@unicode` Emo currently supports three different backends for actually rendering emoji, namely the backend named `\emo@use@unicode` emits Unicode codepoints in a group, the one named `\emo@use@font` emits font selection before those same Unicode codepoints in the group, and the one named `\emo@use@pdf` emits PDF graphics. Once we know to name the backends, we can set `\emo@backend` to the currently active backend, determined by interrogating the runtime environment. Alas, we still need to implement the three backends; but `\emo@content` is defined closer to the end of the package implementation.

```
10 \def\emo@use@unicode{backend:unicode}
11 \def\emo@use@font{backend:font+unicode}
12 \def\emo@use@pdf{backend:pdf}
13 \RequirePackage{iftex}
```

```

14 \ifdefined\HCode
15   \let\emo@backend=\emo@use@unicode
16 \else
17 \ifluatex
18   \let\emo@backend=\emo@use@font
19 \else
20   \let\emo@backend=\emo@use@pdf
21 \fi
22 \fi

```

With the backend selected, we can now require backend-specific packages, namely `fontspec` for selecting fonts in the `\emo@use@font` backend and `graphicx` for including PDF graphics in the `\emo@use@pdf` backend. The `\emo@use@unicode` backend has no similar requirements.

```

23 \ifx\emo@backend\emo@use@font
24   \RequirePackage{fontspec}
25 \fi
26 \ifx\emo@backend\emo@use@pdf
27   \RequirePackage{graphicx}
28 \fi

```

Next, `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.

```

29 \RequirePackage{xcolor}

```

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

```

30 \ifemo@extra
31   \RequirePackage{xspace}
32 \fi
33 \ifemo@indexing
34   \RequirePackage{index}
35 \fi

```

5.3 The Emoji Table

`\emo@emoji@name` For each emoji with a PDF graphic in the `emo-graphics` directory and, if enabled, the two extra macros, the corresponding `\emo@emoji@<emoji-name>` macro 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. 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. Alas, that seems a bit much for turning two files into one.

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

5.4 Internal Macros

`\emo@error@fg` 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 over-
`\emo@error@bg` look the warning in the console, you *will* notice the error message in the doc-
`\emo@error` ument thusly formatted.

```
37 \definecolor{emo@error@fg}{rgb}{1,1,1}
38 \definecolor{emo@error@bg}{rgb}{.6824,.0863,.0863}
39 \def\emo@error#1{%
40     \colorbox{emo@error@bg}{%
41         \textcolor{emo@error@fg}{%
42             \textsf{Bad} \texttt{\textbackslash emo\{#1\}}%
43         }%
44     }%
45 }
```

`\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.

```
46 \def\emo@ifdef#1#2{%
47     \ifcsname emo@emoji@#1\endcsname#2\else%
48         \PackageWarning{emo}{Unknown emoji name in ‘\string\emo{#1}’}%
49         \emo@error{#1}%
50     \fi%
51 }
```

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

```
52 \ifemo@index
53     \newindex{emo}{edx}{end}{Emoji Index}
54     \def\emo@index#1{\index[emo]{#1}}
55 \else
56     \def\emo@index#1{}
57 \fi
```

`\emo@content` Render the emoji content. This macro interfaces with the backend and thus needs to be defined as many times as there are backends: The Unicode backend just expands the entry from the emoji table. The font backend does the same, but sets the font to Noto Color Emoji first. The PDF backend instead includes the corresponding PDF graphic.

```
58 \ifx\emo@backend\emo@use@unicode
59     \def\emo@content#1{%
```

```

60      \begingroup\csname emo@emoji@#1\endcsname\endgroup%
61    }
62 \else
63 \ifx\emo@backend\emo@use@font
64   \newfontface\emo@font[Renderer=Harfbuzz]{NotoColorEmoji.ttf}
65   \def\emo@content#1{%
66     \begingroup\emo@font\csname emo@emoji@#1\endcsname\endgroup%
67   }
68 \else
69   \def\emo@content#1{%
70     \raisebox{-0.2ex}{%
71       \includegraphics[height=1em]{emo-graphics/emo-#1}}%
72   }
73 \fi
74 \fi

```

In debug mode, emo draws a box around the content of \emo. That may help when tracking down spurious whitespace.

```

75 \ifemo@debug
76   \let\emo@realcontent=\emo@content
77   \def\emo@content#1{\fbox{\emo@realcontent{#1}}}
78 \fi

```

5.5 User Macros

\emo Thanks to carefully defined internal macros, the implementation of the main \emo macro is almost trivial. If the emoji name passes muster, emit an index entry and render the emoji content:

```

79 \newcommand\emo[1]{%
80   \emo@ifdef{#1}{%
81     \emo@index{#1}%
82     \emo@content{#1}%
83   }%
84 }

```

\lingchi Since the emoji table in emo.def includes macros with the Unicode codepoints \YHWH for “lingchi” and “YHWH,” the implementation of \lingchi and \YHWH just delegates to \emo.

```

85 \ifemo@extra
86 \ifx\emo@backend\emo@use@font\else
87   \newcommand\lingchi{\emo{lingchi}\xspace}
88   \newcommand\YHWH{\emo{YHWH}\xspace}
89 \fi

```

Except, as hinted at by the backend test, delegating to \emo won’t work when using fonts, since \emo uses Noto color emoji whereas the two extra macros do not. In that case, we define alternative versions that, similar to \emo rely

on their own specialized `\emo@content` macros and also wrap them when the `debug` package option is enabled.

```

90 \ifx\emo@backend\emo@use@font
91   \newfontface\emo@chinese{emo-lingchi.ttf}
92   \newfontface\emo@hebrew{LinLibertine_R.otf}
93   \def\emo@lingchi@content{\begingroup\emo@chinese\emo@emoji@lingchi\endgroup}
94   \def\emo@YHWH@content{\begingroup\emo@hebrew\emo@emoji@YHWH\endgroup}
95
96   \ifemo@debug
97     \let\emo@lingchi@realcontent=\emo@lingchi@content
98     \let\emo@YHWH@content=\emo@YHWH@content
99     \def\emo@lingchi@content{\fbox{\emo@lingchi@realcontent}}
100    \def\emo@YHWH@content{\fbox{\emo@YHWH@realcontent}}
101  \fi
102
103  \newcommand\lingchi{\emo@index{lingchi}\emo@lingchi@content\xspace}
104  \newcommand\YHWH{\emo@index{YHWH}\emo@YHWH@content\xspace}
105 \fi
106 \fi

```

Et voilà. That's it!

```

107 </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, require the `xspace` package and then `\YHWH` provide minimal re-definitions of the `\lingchi` and `\YHWH` macros. Both simply expand to the necessary Unicode codepoints.

```
19if (IfCondition(T_CS('\ifemo@extra')) {
20  RequirePackage('xspace');
21  DefMacro('\lingchi', "\x{51cc}\x{9072}\xspace");
22  DefMacro('\YHWH', "\x{05D9}\x{05D4}\x{05D5}\x{05D4}\xspace");
23 }
```

That's it for the binding, too.

```
24</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.

```
25 < *scaffold>
26 \iffalse
27 < /scaffold>
```

7.1 Test Class

```
1 < *testing>
```

The emo-test class builds on the standalone class so that the resulting PDF has just the size of the output (plus a small margin). We require emo for quality control, iftex for determining the LaTeX engine, and xcolor for appearances sake.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesClass{emo-test}[2023/05/01 Testing emo]
4 \LoadClass[border=10pt, varwidth=6in]{standalone}
5 \RequirePackage[extra]{emo}
6 \RequirePackage{iftex}
7 \RequirePackage{xcolor}
```

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

```
8 \iftutex
9 \RequirePackage{fontspec}
10 \RequirePackage{libertinus}
11 \setmonofont{inconsolata}
12 \else
13 \RequirePackage{libertinus}
14 \RequirePackage{inconsolata}
15 \fi
```

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

```
16 \ifxetex
17 \def\enginename{XeTeX}
18 \else
19 \ifluatex
20 \def\enginename{LuaTeX}
21 \else
22 \ifpdfTEX
23 \def\enginename{pdfTeX}
```

```

24 \else
25 \def\enginename{unknown engine}
26 \fi
27 \fi
28 \fi

```

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

emo@canary@frameouter

```

emo@canary@background 29 \definecolor{emo@canary@frameinner}{HTML}{636366}
30 \definecolor{emo@canary@frameouter}{HTML}{48484A}
31 \definecolor{emo@canary@background}{HTML}{E5E5EA}

```

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

```

32 \setlength{\fboxrule}{0.5pt}
33 \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

```

34 \newcommand\emo@nobox[1]{#1}
35 \newcommand\emo@wordbox[1]{\fcolorbox{emo@canary@frameinner}{white}{#1}}
36 \newcommand\emo@linebox[1]{\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.

```

37 \def\@sampletext#1{%
38   \emo@linebox{%
39     #1{It's} #1{\lingchi}:
40     #1{Please}, #1{\YHWH}, #1{have} #1{mercy}
41     #1{\emo{pleading-face}}!%
42   }%
43   \vspace{1ex}%
44 }
45 \newcommand*\sampletext{%
46   \ifstar{\@sampletext{\emo@wordbox}}{\@sampletext{\emo@nobox}}%
47 }

```

\emo@canary@actual Validating emo's macros turned out to be a bit trickier than I had expected. The
\emo@canary@expected 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.

```
48 \newsavebox{\emo@canary@actual}
49 \newsavebox{\emo@canary@expected}
```

`checkwidth` \lingchi and \YHWH rely on `xspace` to avoid (ugly) trailing backslash characters. But that makes both macros fundamentally context-sensitive. To ensure meaningful test results, we must fix the context too. `\checkwidth` does that by following the macro output with a period.

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.

```
50 \newcommand\checkwidth[5]{%
51   \sbox\emo@canary@actual{#2.}%
52   \ifx\emo@backend\emo@use@unicode%
53     \sbox\emo@canary@expected{\begingroup #4\endgroup.}%
54   \else%
55     \ifx\emo@backend\emo@use@font%
56       \sbox\emo@canary@expected{%
57         \begingroup\csname#3\endcsname #4\endgroup.}%
58     \else%
59       \sbox\emo@canary@expected{%
60         \raisebox{-0.2ex}{%
61           \includegraphics[height=1em]{emo-graphics/emo-#5}}.}%
62   \fi%
63   \fi%
64   \def\macroname{\texttt{\char'\#1}}%
65   \ifdim\wd\emo@canary@actual=\wd\emo@canary@expected%
66     \mbox{\macroname{} \emo{check-mark-button}}%
67   \else%
68     \edef\emo@actual{\the\wd\emo@canary@actual}%
69     \edef\emo@expected{\the\wd\emo@canary@expected}%
70     \mbox{\macroname{} \emo{cross-mark} \emo@actual{} \emo@expected}%
71   \fi%
72 }
```

Use small gap between paragraphs instead of indentation.



```
73 \setlength\parindent{0pt}
74 \setlength{\parskip}{1ex}
```

That's it for emo's testing class.

```
75 </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 \engine: {\Large Width of
6 \checkwidth{emo}{\emo{robot}}{\emo@font}{\char"1F916}{robot},
7 \checkwidth{lingchi}{\lingchi}{emo@chinese}{\char"51CC\char"9072}{lingchi},
8 \checkwidth{YHWH}{\YHWH}{emo@hebrew}{\char"5D9\char"5D4\char"5D5\char"5D4}{YHWH}}
9 \vspace{1ex}\par
```

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

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

That's it for the tests and report.

```
12 </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	General: Make initial release	1	symbolic names	7
0.2	General: Add LaTeXXML binding for conversion to HTML	11	\emo@emoji@name: Include \lingchi and \YHWH as necessary	8
	Prefix font and graphic files with “emo-”	1	\lingchi: Build on \emo by default	10
	Support pdftex for extracting emo.dtx	1	\YHWH: Build on \emo by default . .	10
0.3	General: Support TeX4ht for conversion to HTML	1	0.4	General: Automate testing across engines with canary.tex
	\emo@backend: Make backend support scalable through			Introduce a simple unit testing framework
				\emo@content: Remove extra space from PDF renderer
				\ifemo@debug: Add debug option for drawing boundary boxes . . .
				7

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	
\ifstar	46
\@sampletext	37, 46
B	
\begin	3, 3
C	
\checkwidth	6, 7, 8, <u>50</u> , 50
\colorbox	40
D	
\DeclareOption	3, 5, 7
\definecolor	29, 30, 31, 37, 38
\documentclass	2, 2
E	
\edef	68, 69
\emo	6, <u>17</u> , 41, 48, 66, 70, <u>79</u> , 87, 88
\emo@actual	68, 70
\emo@backend	
.	<u>10</u> , 23, 26, 52, 55, 58, 63, 86, 90
\emo@canary@actual	<u>48</u> , 51, 65, 68
\emo@canary@background	<u>29</u>
\emo@canary@expected	
.	<u>48</u> , 53, 56, 59, 65, 69
\emo@canary@frameinner	<u>29</u>
\emo@canary@frameouter	<u>29</u>
\emo@chinese	91, 93
\emo@content	<u>58</u> , 82
\emo@debugfalse	6
\emo@debugtrue	7
\emo@emoji@lingchi	93
\emo@emoji@name	<u>17</u> , <u>36</u>
\emo@emoji@YHWH	94
\emo@error	<u>37</u> , 49
\emo@error@bg	<u>37</u>
\emo@error@fg	<u>37</u>
\emo@expected	69, 70
\emo@extrafalse	2, 11
\emo@extratrue	3, 13
\emo@font	64, 66
\emo@hebrew	92, 94
\emo@ifdef	<u>46</u> , 80
\emo@index	<u>52</u> , 81, 103, 104
\emo@indexingfalse	4
\emo@indexingtrue	5
\emo@linebox	<u>34</u> , 38
\emo@lingchi@content	93, 97, 99, 103
\emo@lingchi@realcontent	97, 99
\emo@nobox	<u>34</u> , 46
\emo@realcontent	76, 77
\emo@use@font	<u>10</u> , 23, 55, 63, 86, 90
\emo@use@pdf	<u>10</u> , 26
\emo@use@unicode	<u>10</u> , 52, 58

\emo@wordbox	34, 46
\emo@YHWH@content	94, 98, 100, 104
\emo@YHWH@realcontent	100
\end	5, 11
\enginename	5, 16
F	
\fbox	77, 99, 100
\fboxrule	32
\fboxsep	33
\fcolorbox	35, 36
H	
\HCode	14
\Huge	4, 4
I	
\ifdefined	14
\ifdim	65
\ifemo@debug	2, 75, 96
\ifemo@extra	2, 8, 19, 30, 85
\ifemo@index	2
\ifemo@indexing	4, 33, 52
\iffalse	26
\ifpdfTeX	22
\iftutex	8
\ifx	23, 26, 52, 55, 58, 63, 86, 90
\ifxetex	16
\includegraphics	61, 71
\input	36
L	
\Large	5
\let	15, 18, 20, 76, 97, 98
\lingchi	7, 19, 39, 85
\LoadClass	4
M	
\macroname	64, 66, 70
\mbox	66, 70
N	
\NeedsTeXFormat	2
\newfontface	64, 91, 92
\newsavebox	48, 49
P	
\PackageWarning	48
\par	9, 10
\parindent	73
\parskip	74
\ProcessOptions	8
\ProvidesClass	3
\ProvidesPackage	9
R	
\raisebox	60, 70
\RequirePackage	5, 6, 7, 9, 9, 10, 13, 13, 14, 24, 27, 29, 31, 34
S	
\sampletext	4, 10, 37
\sbox	51, 53, 56, 59
\setlength	32, 33, 73, 74
\setmonofont	11
T	
\textcolor	41
\the	68, 69
V	
\vspace	9, 43
W	
\wd	65, 68, 69
X	
\xspace	87, 88, 103, 104
Y	
\YHWH	8, 19, 40, 85