Converting MD containing emojis to PDF

Markdown to PDF conversion is quite easy using `pandoc`. The real problem arises when trying to use unicode emojis. These emojis are becoming a real standard right now, and they are also useful, since not only faces and hands are represented, but also, as one can see below, they can represent common concepts such as packages, folders and files.

However useful it is, it's quite difficult to convert a markdown file containing emoji characters. You must setup special fonts, overcome lack of support by the latex engines, only to have black-and-white emojis.

Representing a file-system structure

📦package
┣ 📁dir1
┃ ┗ 📁subdir
┗ 📜file1

Limitations

- Pre/Code tags are ignored by Pandoc when converting to PDF. This filter is not supposed to do anything about this, so I won't be doing anything here about it.

  The following code block uses `<pre>...</pre>` tags. They will not be rendered in the PDF, but are visible in the markdown visualization.

  There should be a code block above... if you are looking at the example.pdf, you'll see nothing above.

- Codeblocks and Code, like `📁dir1`, are now being parsed. The `CodeBlock` nodes are replaced by the latex `Verbatim` provided by the package `fvextra`. The `Code` nodes are replaced by a sequence of `RawInline` and `Code` nodes, literally it just remove the emojis from the code node, splitting the code node if necessary. This is less than ideal, but for now it is working. It could be placed inside the latex `texttt` command, but that would be more laborious.

  ```
  📦package
  ┣ 📁dir1
  ┃ ┗ 📁subdir
  ┗ 📜file1
  ```

- Literal emojis like `:name:` are not converted to real emojis. Use `--from markdown+emoji` or `--from gfm`.

> 😄 ✈️
>
> See Non-pandoc extensions - Pandoc manual

Compiling a PDF from this `readme.md` file

In this project, I experimented with multiple ways to do what I wanted. First, I tried with Javascript emoji converter + Python Pandoc filter. Then, I discovered that Pandoc filters could be made in Javascript, using NodeJs, and made a completely JS solution.

Use the NodeJS solution instead of the Python as it is the one I am currently developing more actively.

Javascript + Python filter

```
node app.js
pandoc --template="template.tex" -o out.pdf output.md \
    --filter=svg_filter.py
```

Javascript only filter via NodeJs

First, install JS script dependencies using npm:

```
npm install
```

Then, run `pandoc` passing in the filter file name:

```
pandoc --template="template.tex" -o out.pdf readme.md \
    --filter=emoji_filter.js
```

Passing parameters:

```
pandoc --template="template.tex" -o out.pdf readme.md \
    --filter=emoji_filter.js -M __debug=1 -M emoji=twemoji
```

Debugging using VSCode:

```
pandoc --template="template.tex" -o out.pdf readme.md \
    --filter=emoji_filter.js -M __debug=1
```

Then, in VSCode, attach to Node process... the JS filter code will be waiting, and will only continue execution after the debugger is attached. If you don't attach the debugger, then it will stall in an infinite loop.

Example: compiling this `readme.md`

This `readme.md` file was compiled using the following command:

```
pandoc --template="template.tex" -o example.pdf readme.md \
    --filter=emoji_filter.js -M emoji=noto-emoji --from gfm
```

This is the resulting PDF: example.pdf

You can also run `create-example.sh`.

Filter parameters

Parameters are passed to the filter in the form of metadata. Pandoc receives metadata using the `--metadata` or `-M` keywords. Also, metadata can be specified inside markdown files, or YAML files via `--metadata-file`. See Pandoc's manual - Metadata Blocks for more info.

__debug

This is used to debug the filter using NodeJS inspector. VSCode can debug NodeJS instances, even the ones that started without debug parameters, via `SIGUSR1` signal.

The problem the this flag solves it that the filter execution must wait for the debugger to be attached, otherwise it finished before the user has the opportunity to debug the code.

Usage: pass `-M __debug=1` to pandoc along with other params.

emoji

This is used to select the emoji source. At this moment only two sources are available:

- `twemoji`: Twemoji by Twitter
- `noto-emoji`: Noto Color Emoji by Google

Usage: pass `-M emoji=noto-emoji` or `-M emoji=twemoji` to pandoc along with other params.

See Emojipedia for a list of emojis.

Changing emoji cache directory

Just set the environment variable `SVG_FILTER_CACHE_DIR`.

You may need to use `declare -x SVG_FILTER_CACHE_DIR` in your script, so that the filter can see the environment variable.

References

- Full Emoji List - unicode.org
- Emojipedia
- VSCode extension: file-tree-generator
- https://github.com/googlefonts/noto-emoji
- `\usepackage{pmboxdraw}`: this is used to draw file-structure lines