github-flavored-markdown-to-html

Convert Markdown to html via python or with a command line interface. Uses Githubs online Markdown-to-html-API as well as Githubs Markdown-CSS. Requires internet connection to work.

This module is intended to be used for the creation of static pages from markdown files, for example in conjunction with a static website builder or github actions if you host on Github, but can be very well used for any other purpose. It also allows you to convert the html files to pdf on the fly.

Advantages include:

- Lets you specify the markdown to convert as a string, as a repository path, as a local file name or as a hyperlink.
- Pulls any images referenced in the markdown files from the web/ your local storage and places them in a directory relative to your website root, so you can host it all locally without relying on third-party-websites.
- Creates all links as root-relative hyperlinks and lets you specify the root directory as well as the locations for css and images, but uses smart standard values for everything.
- Supports inline LaTeX-formulas (use `$`-formula-`$` to use them), which GitHub usually doesn't (this is done using the Codecogs EqnEditor).
- Supports exporting as pdf with or without Github styling, using the pdfkit python module (if it is installed).
- Tested and optimized to look good when using Darkreader (the .js-module, not nessesarily the browser extension. This means that formulas are displayed with a light text when in darkmode, amongst other things).

Installation

Use `pip3 install gh_md_to_html` (might require `sudo` on Linux) and optionally `pip3 install pdfkit` (if you want to use the optional pdf features) to install it.

Usage

If you want to access the interface with your command line, you can just supply `gh_markdown_to_html` with the arguments documented in th help text (accessible with `gh_markdown_to_html -h` and shown below.)

If you want to access the interface via python, you can use

```
import gh_md_to_html
```

and then use `gh_md_to_html.main()` with the same arguments (and default values) you would supply to the command line interface.

Documentation

All arguments and how they work are documented in the help text of the program, which looks like this:

```
usage: __init__.py [-h] [-t {file,repo,web,string}] [-w WEBSITE_ROOT]
                   [-d DESTINATION] [-i IMAGE_PATHS] [-c CSS_PATHS]
                   [-n OUTPUT_NAME] [-p OUTPUT_PDF] [-s STYLE_PDF] [-f FOOTER]
                   [-m MATH] [-r FORMULAS_SUPPORTING_DARKREADER]
                   MD-origin

Convert markdown to HTML using the GitHub API and some additional tweaks with
python.

positional arguments:
  MD-origin             Where to find the markdown file that should be
                        converted to html

optional arguments:
  -h, --help            show this help message and exit
  -t {file,repo,web,string}, --origin-type {file,repo,web,string}
                        In what way the MD-origin-argument describes the origin
                        of the markdown file to use. Defaults to file. The
                        options mean:
                        * file: takes a relative or absolute path to a file
                        * repo: takes a path to a markdown-file in a github
                        repository, such as <user_name>/<repo_name>/<branch-
                        name>/<path_to_markdown>.md
                        * web: takes an url to a markdown file
                        * string: takes a string containing the files content
  -w WEBSITE_ROOT, --website-root WEBSITE_ROOT
                        Only relevant if you are creating the html for a static
                        website which you manage using git or something similar.
                        --html-root is the directory from which you serve your
                        website (which is needed to correctly generate the links
                        within the generated html, such as the link pointing to
                        the css, since they are all root- relative), and can be
                        a relative as well as an absolute path. Defaults to the
                        directory you called this script from. If you intent to
                        view the html file on your laptop instead of hosting it
                        on a static site, website-root should be a dot and
                        destination not set. The reason the generated html files
                        use root-relative links to embed images is that on many
                        static websites, https://foo/bar/index.html can be
                        accessed via https://foo/bar, in which case relative
                        (non-root- relative) links in index.html will be
                        interpreted as relative to foo instead of bar, which can
                        cause images not to load.
  -d DESTINATION, --destination DESTINATION
```

```
                        Where to store the generated html. This path is relative
                        to --website-root. Defaults to "".
  -i IMAGE_PATHS, --image-paths IMAGE_PATHS
                        Where to store the images needed or generated for the
                        html. This path is relative to website-root. Defaults to
                        the "images"-folder within the destination folder.
  -c CSS_PATHS, --css-paths CSS_PATHS
                        Where to store the css needed for the html (as a path
                        relative to the website root). Defaults to the
                        "<WEBSITE_ROOT>/github-markdown-css"-folder.
  -n OUTPUT_NAME, --output-name OUTPUT_NAME
                        What the generated html file should be called like. Use
                        <name> within the value to refer to the name of the
                        markdown file that is being converted (if you don't use
                        "-t string"). You can use '-n print' to print the file
                        (if using the command line interface) or return it (if
                        using the python module), both without saving it.
  -p OUTPUT_PDF, --output-pdf OUTPUT_PDF
                        If set, the file will also be saved as a pdf file in the
                        same directory as the html file, using pdfkit, a python
                        library which will also need to be installed for this to
                        work. You may use the <name> variable in this value like
                        you did in --output-name.
  -s STYLE_PDF, --style-pdf STYLE_PDF
                        If set to false, the generated pdf (only relevant if you
                        use --output-pdf) will not be styled using github's css.
  -f FOOTER, --footer FOOTER
                        An optional piece of html which will be included as a
                        footer where the 'hosted with <3 by github'-footer
                        usually is. Defaults to None, meaning that the section
                        usually containing said footer will be omitted
                        altogether.
  -m MATH, --math MATH  If set to True, which is the default, LaTeX-formulas
                        using $formula$-notation will be rendered.
  -r FORMULAS_SUPPORTING_DARKREADER, --formulas-supporting-darkreader FORMULAS_SUPPORTING_DARKREADER
                        If set to true, formulas will be shown light if the
                        darkreader .js library is included in the html and the
                        user prefers darkmode. This is checked by looking for a
                        script embedded from a src ending with "darkreader.js"
                        and by checking the prefers-color- scheme option in the
                        browser. You can also supply any other script src to
                        look for. Please note that this won't have any effect
                        unless you inject the darkreader .js library into the
                        generated html; doing so is not included in this module.
```

As mentioned above, any image referenced in the markdown file is stored locally and referenced using a root-relative hyperlinks in the generated html. How the converter guesses the location of the image is shown in the following table, with the type of imagelink noted on the top and the type of input markdown noted on the left:

	`https://` or `http://`	abs. filepath	rel. filepath	starting with `/` (e.g. `/image.png`)	not starting with `/` (e.g. `image.png`)
`-t file`	from the address	abs. filepath	rel. filepath (from where the `.md`-file lies)	-	-
`username/repo/dir/file.md -t repo`	from the address	-	-	`username/repo/imagedir/image.png`	`username/repo/dir/imagedir/image.png`
`-t string`	from the address	abs.filepath, but needs confirmation for security reasons	rel. filepath (to where the tool is called from), but needs confirmation for security reasons	-	-
`https://foo.com/bar/baz.md -t web`	from the address	-	-	`https://foo.com/image.png`	`https://foo.com/bar/image.png`

Demonstration

I added the following demonstration, whose files where created from the root directory of this projects directory, which relates to the root directory of the site I am hosting them on:

generated with:	view:	demonstrates what:	notes:
python3 github-flavored-markdown-to-html/**init**.py github-flavored-markdown-to-html/README.md -d github-flavored-markdown-to-html/docs -c github-flavored-markdown-to-html/docs/css -f "test footer <3"	here	html (+footer)	
python3 github-flavored-markdown-to-html/**init**.py github-flavored-markdown-to-html/README.md -n README-darkmode.html -d github-flavored-markdown-to-html/docs -c github-flavored-markdown-to-html/docs/css -r true	here	html (without a footer) and that the html supports embedding the darkreader .js library without showing dark formulas on dark ground etc.	I injected the following into the html: <script type="text/javascript" src="https://phseiff.com/darkreader/darkreader.js"></script>DarkReader.setFetchMethod(window.fetch); DarkReader.auto({brightness: 100,contrast: 90, sepia:

10});</script>

python3 github-flavored-markdown-to-html/**init**.py github-flavored-markdown-to-html/README.md -d github-flavored-markdown-to-html/docs -n print -c github-flavored-markdown-to-html/docs/css -p README.pdf	here	Converting to pdf.
python3 github-flavored-markdown-to-html/**init**.py github-flavored-markdown-to-html/README.md -d github-flavored-markdown-to-html/docs -n print -c github-flavored-markdown-to-html/docs/css -p README-unstyled.pdf -s false	here	Converting to pdf without styling.

I also added a $formula$ here ($\sum_{i\geq e^2}^{7.3}\frac{4}{5}$) to demonstrate the formula rendering (which you won't see when viewing this README directly on github since, like I said, github usually doesn't support it.)

Some Notes

In case you are not happy with the margin left and right of the text, you can manually adjust it by modifying the margin-values hardcoded in prototype.html in this repository.

DISCLAIMER: This module is neither written by Github, nor is it endorsed with Github, supported by Github, powered by Github or affiliated with Github. It only uses a public API provided by Github as well as a .css-file licensed by Github under the MIT license.