DEFG

Generate beautiful PDF documentation from code

v1.7.9

charles.lobo@gmail.com

https://github.com/theproductiveprogrammer/defg#readme

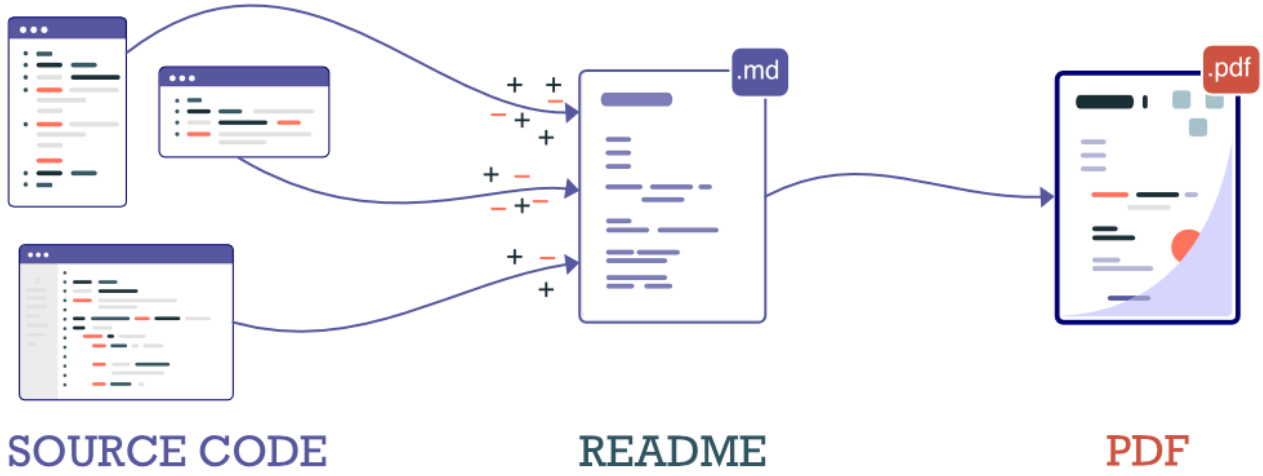
You can find this document formatted nicely here.

How it Works

1. Step 1: Write documentation comments in code (`//**` or `##**` comments)
2. Step 2: `defg` merges comments into your README/markdown file, updating it.
3. Step 3: `defg` uses plugins to generate a beautifully formatted PDF.

As you update the documentation in the code, `defg` will find and merge the updates, always keeping your markdown and PDF documentation up to date. You can add images, styling, and layout to the markdown and `defg` will still find and merge your updates correctly.

Below is a schematic of how this works:



Installation & Use

`defg` is a npm script and can be installed globally using:

```
$> npm install -g defg
```

Then enter the directory where your README.md is present, and run

```
$> defg
```

You can get help using:

```
$> defg -h
```

Motivation

I find that README's and other documentations tend to get out of date quickly. `defg` generates a README from documentation comments, and as these comments are close to the code, they are easier to access, modify, and update.

First Lines

If you start your documentation with a title, subtitle, and image, the default plugin will create a cover page for you.

```
//** # Use a Heading as the Title
//** The next line is a subtitle (optional), followed by a logo
//** ![logo](./logo.png)
```

It will also pull version number, author, and homepage from NodeJS `package.json` files.

First Run

If you run `defg` and there is no README.md file, it will generate one from all the documentation it has found. During this process, because it has no reference, it can get the order of comments mixed up. If this happens, you need to reorder the README to get it into shape.

Improving the Documentation

You can also update your document to make it more readable. You can:

1. Add pictures.
2. Add styling.
3. Add text.
4. Configure the page layout.

As you do all these, `defg` will preserve your changes whenever it updates your document.

HOW TO IMPROVE THE DOCUMENT

- You can add images in markdown or using the `<img..` tag. You can also add other block HTML styling `<div class="..."` in the document to improve it's look.
- You can add a README.css which will apply the CSS styles while generating the PDF.
- To insert additional text in the README, wrap the additional text in a `<div class="insert-block">...</div>` .
- To insert a page break insert a `<div class="page-break" />` and add the style to your CSS: ```css .page-break { page-break-after: always }
- To design the page layout, create a `pages.defg` file. Here you can decide the page size, header & footer using the following Puppeteer options: https://pptr.dev/api/puppeteer.pdfoptions

Usage

```
$> defg -h
  # Generates beautiful PDF documentation from documentation comments
  # (Comments starting with //** or ##**)
Options
   -h, --help:     show help
   -v, --version:  show version
   --src:          glob paths to source files [can be multiple]
   --skip:         glob paths to exclude/ignore when searching source
files
   --ext:          list of valid source file extensions
(js,py,java,sql,ts,sh,go,c,cpp by default)
   --readme:       path of README file to merge (./README.md by default)
   --style:        path of CSS file containing styling (./README.css by
default)
   --page-def:     path of file containing pdf page definition
   --pdf:          path of output pdf generated (./README.pdf by default)
   --ignore-src:   ignore source and just generate PDF from README.md
   --quick:        use faster (but less accurate) resolution algorithm
   --plugin:       update the documentation or style using this plugin
```

For those interested, a technical discussion of `defg` can be found here.