DEFG - Generate documentation from code comments

Motivation

I find that README's and other documentations tend to get out of date quickly. Using `defg` you can simply generate your README from documentation comments, and as these comments are close to the code, they are be easier to access, modify, and update.

It still takes some discipline to update - instead of ignoring - the documentation comments, but it's easier to do as you are changing/reviewing the code.

How does it work?

`defg` trawls through all 'programming' files (.js, .sh, .java, .c, .cpp,...) it finds and extracts 'special' comments that start with `//**` or `##**`. These are considered 'user documentation' comments. It then uses them to update the README, generates a nice PDF, and opens it.

Why not just generate the PDF from the code documentation?

Because when we write documentation in code file it should be related to the code in the file. This documentation may be distributed across multiple files - close to classes and functions.

The actual user documentation will have (a) a nice, defined, flow (ordering of the documentation pieces) and (b) additional images, html etc, that may not quite fit into a specific code file.

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, it can get the order of comments all mixed up. You are encouraged to then go and reorder all the pieces in the README.md to get it into a nice shape.

Adding Images

In order to make your documentation prettier, you can chose to include images

etc. If you do this, `defg` will attempt to preserve these in the updated README.

Usage

```
$> defg
  # ensures README contains all 'user documentation' comments (//** or ##** comments)
  # and then generates and opens a PDF with the user documentation

Options
   -h, --help:     show help
   -v, --version:  show version
   --src:          path of source files (or path of a single source file)
   --readme:       path of README file to check (./README.md by default)
   --style:        path of CSS file containing stying (./README.css by default)
   --pdf:          path of pdf generated (./README.pdf by default)
   --ext:          list of valid source file extensions (js,py,java,sql,ts,sh,go,c,cpp by default)
   --ignore-src:   ignore source and just generate from readme
```