DEFG - Generate documentation from code comments

WHY!?

I find that README's and other documentations tend to get out of date really *really* quickly. The hope here is that user documentation kept close to code should be easier to access, modify, and update.

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 checks that all these lines are present in the README. If they are, it generates a nice PDF from and opens it.

Why not just generate the PDF from the code documentation?

Because when we write documentation in code file it's probably related to the code in the file. This documentation may be distributed across multiple files.

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

All in all, having the documentation updated separately turns out to be better than trying to auto-generate it.

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
```