A Whirlwind Tour of RHEOS Documentation

J Louis Kaplan

Abstract

This document is a short overview of how the RHEOS documentation system works. Many aspects of it also apply to the documentation systems of other Julia packages – the process is fairly standardised. The process can be summarised by the following. The documentation structure is specified in the file 'RHEOS/docs/make.jl' using the string pair syntax discussed below. This will then create links to the various markdown files in the 'RHEOS/docs/src' directory. The documentation can then be built locally by the user, and is built automatically on `git push` by the Travis CI service.

1 Local Documentation

1.1 Directory Structure

1.1.1 User Modified Files

Everything documentation related lives in the 'RHEOS/docs' folder. At this top level directory, the only file you should ever need to modify is 'make.jl'. This file contains all the various documentation build parameters as well as the page structure of the documentation – it will be discussed in further detail later. The rest of the files that are user-modified live in the 'RHEOS/docs/src' folder. This is where there all of the markdown files are placed. It also contains another subfolder 'RHEOS/docs/src/assets' where non-markdown files live such as the RHEOS logo, plots and other images that might need to be displayed in the documentation.

1.1.2 Documenter.jl Generated Files

When you run 'make.jl' by typing `julia make.jl` into a terminal, the documentation will be built locally. (*Note that you would have to be in the 'RHEOS/docs' directory and you may have to* `add Documenter` *in* `pkg` *mode before you run 'make.jl' if you do not have the documenter package installed.*)

Running the script will build the html documentation into the 'RHEOS/docs/build' directory. Try building the docs. Go into the 'build' folder and open up 'index.html' in your favourite browser. You should see the documentation

in the same format as it exists on-line. When you click on a link in this page, it won't take you directly to that page, it takes you a directory structure page with only 'index.html' present. You will need to click on 'index.html' again. This is just the way it works locally. As long as clicking that 'index.html' takes you to the link you were originally expecting to get to, everything is fine.

A last important point on the 'RHEOS/docs/build' directory. It is listed in the '.gitignore' file which means it is not tracked by git. This is how it should be. The documentation seen online is built remotely and pushed to the gh-pages branch of the RHEOS repository. Therefore there is no need for the local build folder to be tracked by git.

1.2 Modifying the Documentation

The documentation of Documenter.jl is the best resource for understanding how to modify the markdown files to meet your requirements. It is well written and so it should be straightforward to learn what you need from it. However, a few features that are commonly used in the RHEOS documentation are listed below for the convenience of the reader.

1.2.1 Special Documenter.jl Markdown Features

The individual documentation pages in 'RHEOS/docs/src' are written in markdown. There are many instructional resources on the web on markdown syntax so it is not discussed in detail here. However, two nice features which are specific to the Documenter.jl parser, and used in the RHEOS documentation, are 1) Automatically gathering function docstrings into the documentation and 2) Linking to other functions and sections. These are briefly described below.

For the RHEOS functions which have help docstrings above them, these can be conveniently imported into the documentation in an aesthetically pleasing way. This is achieved by the `@docs` macro. More specifically, see this excerpt from the 'API.md' documentation source file:

````
```@docs
resample
cutting
smooth
extract
```
````

The resultant html file produced, assuming each of the `resample`, `cutting`, `smooth` and `extract` functions have docstrings, will be each function's argument signature, followed by its docstring.

Functions documented using the above mentioned `@docs` macro will automatically have a reference assigned to them, as will sections (and sub-headings

if desired). This means that easily be linked to through use of the `@ref` macro. For example, from the 'index.md' markdown file:

```
The [API](@ref) section is a comprehensive list of RHEOS types and
functions, and brief descriptions of their use.
```

The word 'API' would then be built as an html link to the 'API' section of the docs in the resultant html. For more info on the `@ref` macro, including how to handle reference conflicts and other more complex cases, see the Documenter.jl documentation.

1.2.2 Modifying the website structure via make.jl

So we have all our individual documentation files in the 'RHEOS/docs/src' directory. Now what? We need to tell Documenter.jl how to structure our documentation so that all our pages are accessible from the side-bar. This is where 'make.jl' comes in.

There are two important parts of the 'make.jl' file. The call to the `makedocs` function and the call to the `deploydocs` function. The first function does the building of the documentation, the second function deploys it to the gh-pages branch on the github repository. For building locally, you don't actually need the call to the `deploydocs` function but it doesn't do any harm if it runs. The `deploydocs` function is relevant to the 'Remote Documentation' section below. This section is concerned with the `makedocs` function. In particular, the `pages` keyword argument. Let's take a look at the code:

```
pages = [
"Home" => "index.md",
  "Fitting Data" => "fittingdata.md",
  "Predicting Responses" => "predictingresponse.md",
  "Generating Data" => "generatingdata.md",
  "Sampling and Filtering" => "samplingandfiltering.md",
  "File I/O" => "fileIO.md",
  "Models" => "models.md",
  "API" => "API.md"
]
```

Now look at the built documentation. You will see that it the first string in each string-pair is listed as a section on the side-bar of the documentation. The second element of each string pair simply points to the actual markdown file which should be linked to that side-bar button.

2 Remote Documentation

As mentioned above, the actual on-line documentation is built remotely. This section is brief overview of how that process works.

Note that Travis CI is integrated with the RHEOS repository via GitHub. Assuming you are logged into GitHub and are a member of the JuliaRheology group, navigate to the RHEOS github repository page. Click 'settings', then click 'Integrations & services'. Note that 'Travis CI' is listed as a 'GitHub App'.

Travis CI reads the '.travis.yml' file in the RHEOS repository and carries out the jobs specified in that file on its own Linux servers. (Could also run OSX but currently not running on that platform.) Open up the '.travis.yml' file. Notice that there are 'stages' under the 'jobs' header. The first stage build the documentation and pushes it to the gh-pages branch, the second stages run the RHEOS tests.