Writing your first documentation PR

The goal of this worksheet is to guide you through the steps needed to successfully create your first PR (pull request). Empty spaces have been placed on this sheet for your notes.

0. Get ready

☐ `install.packages("pak")`

☐ `pak::pak("devtools")`

☐ Set up your GitHub personal access token.

- `usethis.r-lib.org` -> Articles -> Managing Git(Hub) Credentials

☐ Call `usethis::git_sitrep()` and check that:

- ☐ Your name and email address appears in "Git config (global)"
- ☐ Your GitHub user name is found under "Github"
- ☐ Your GitHub personal access token is discovered and there is no warning about its scopes
- ☐ If `Vaccinated: FALSE`, consider running `usethis::git_vaccinate()`.

Place Post-it here

1. Find and claim an issue

Browse the Post-its, looking for something of interest. Since you've never done a PR before, we recommend starting with a documentation issue since there are slightly fewer moving parts. We also encourage you to team up with a (new) friend and tackle something together!

Once you've picked a Post-it, open the issue and read the details in full. At this point, you might discover the issue is outside your wheelhouse; if so, no problem, just return the Post-it to the wall and try again.

2. Get the source locally

☐ **Fork** and clone the repository using this syntax:

- `usethis::create_from_github("_____/_____")`
- (e.g `usethis::create_from_github("tidyverse/ggplot2")`)

☐ `pak::local_install_dev_deps()` to make sure you've got the necessary packages.

☐ Run `devtools::document()` to make sure you have everything necessary to update the docs before you make any changes.

☐ If running `devtools::document()` fails or produces changes, something is probably off with your setup and you should seek help.

3. Make the change

☐ Create a new branch for your changes to live with

- `usethis::pr_init("_____")`
- (e.g `usethis::pr_init("review-count-docs")`)

☐ Make your change.

☐ use `devtools::document()` to update the docs.

☐ Commit the change.

- ☐ Check changed files in the git pane.
- ☐ Press `Commit` button.
- ☐ Write a short message (<80 characters) detailing the change. use the magic fixes keyword, e.g. "Fixes #123", to link & close your issue.
- ☐ Press `Commit` button.

4. Submit your PR

Push and make a PR with `usethis::pr_push()`.

☐ Make the title descriptive:

- Not good: "Working on issue 45"
- Good: "Use stricter regexes for package and file name checks"

☐ Example with descriptive title and description containing magic keywords: https://github.com/r-lib/usethis/pull/742

☐ Write your PR number on the Post-it and move it to the review wall.

☐ Ring the gong to celebrate your successful submission!

5. Wait for review

A person from the tidyverse team will try and review your issue as quickly as possible, reading through your PR and offering suggestions for how to improve it.

- `pr_pause()`: If you want to work on something else while you wait.
- `pr_resume()`: Lets you resume work on a PR you paused with `pr_pause()` by switching back to it.
- `pr_pull()`: If the reviewer makes changes directly to your PR, use this to get their code back onto your computer.
- `pr_finish()`: Does post-PR clean up, but does NOT merge or close a PR. To be done when the PR is finished.

If you've committed a bigger PR, it might take a bit longer to review, and we might not be able to complete it on the day. Don't worry if this happens to you! We really appreciate your contribution and just need a bit more time to take it in.

Wrap up

At the end of the day, make sure that you've reinstalled the CRAN versions of any packages you might have installed development versions of. You can find a list of dev versions with this code:

```
subset(
  pak::lib_status(),
  as.numeric(package_version(version)[, 4]) >= 9000,
  c(package, version)
)
```