# What are Code Reviews and how can they help us?

## Making code understandable

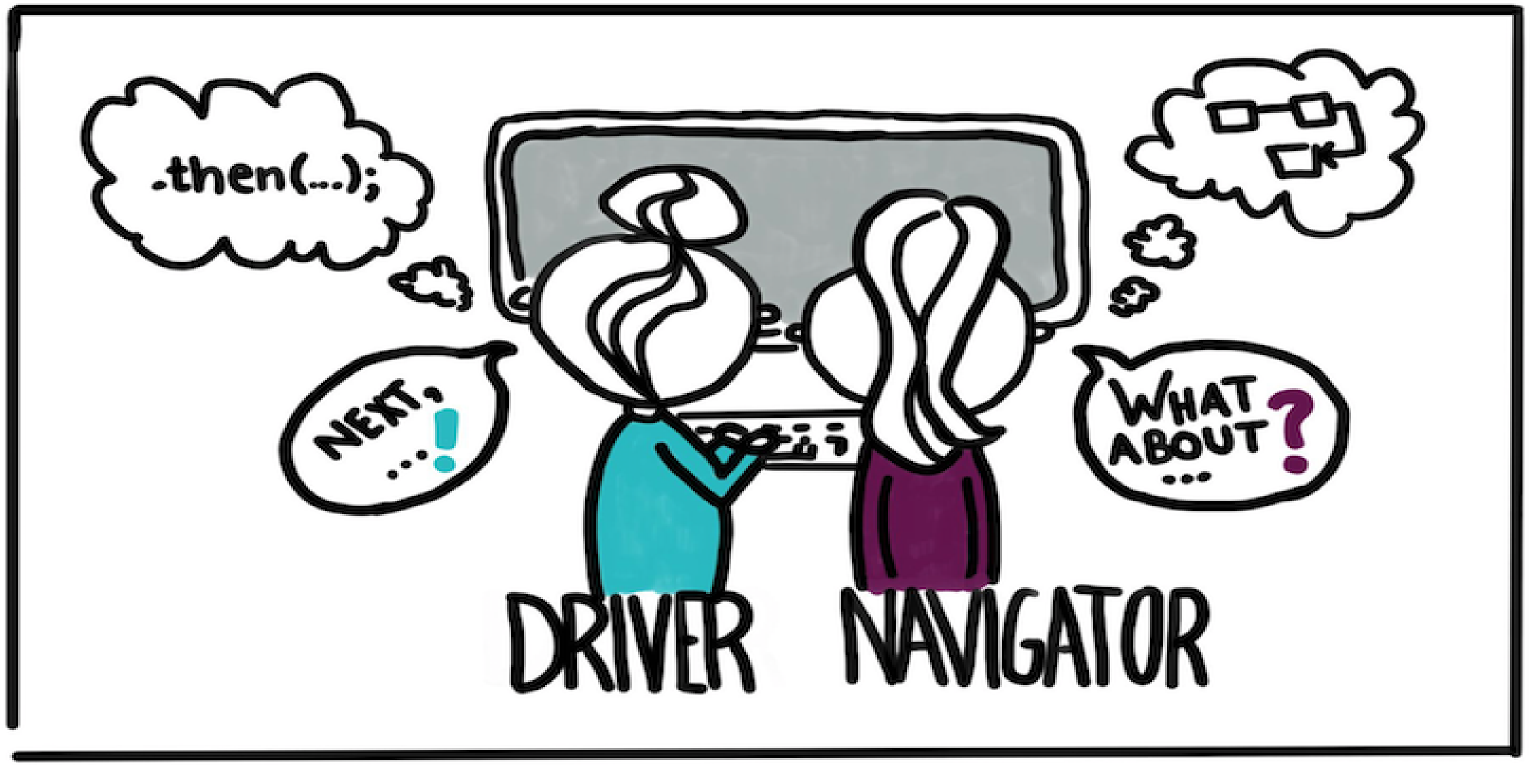
> The most difficult part of writing code is always to make it understandable to other people, including yourself a few months down the track. There's certainly no shame in finding out that your code wasn't as easy to understand or use as you'd hoped, so don't take it personally when it happens (which it always does, at least in my experience), but treat it as an opportunity to improve.
>
> Fernando Perez, Code reviews: the lab meeting for code

## Synchronous - Pair Programming

Consider the following scenario:

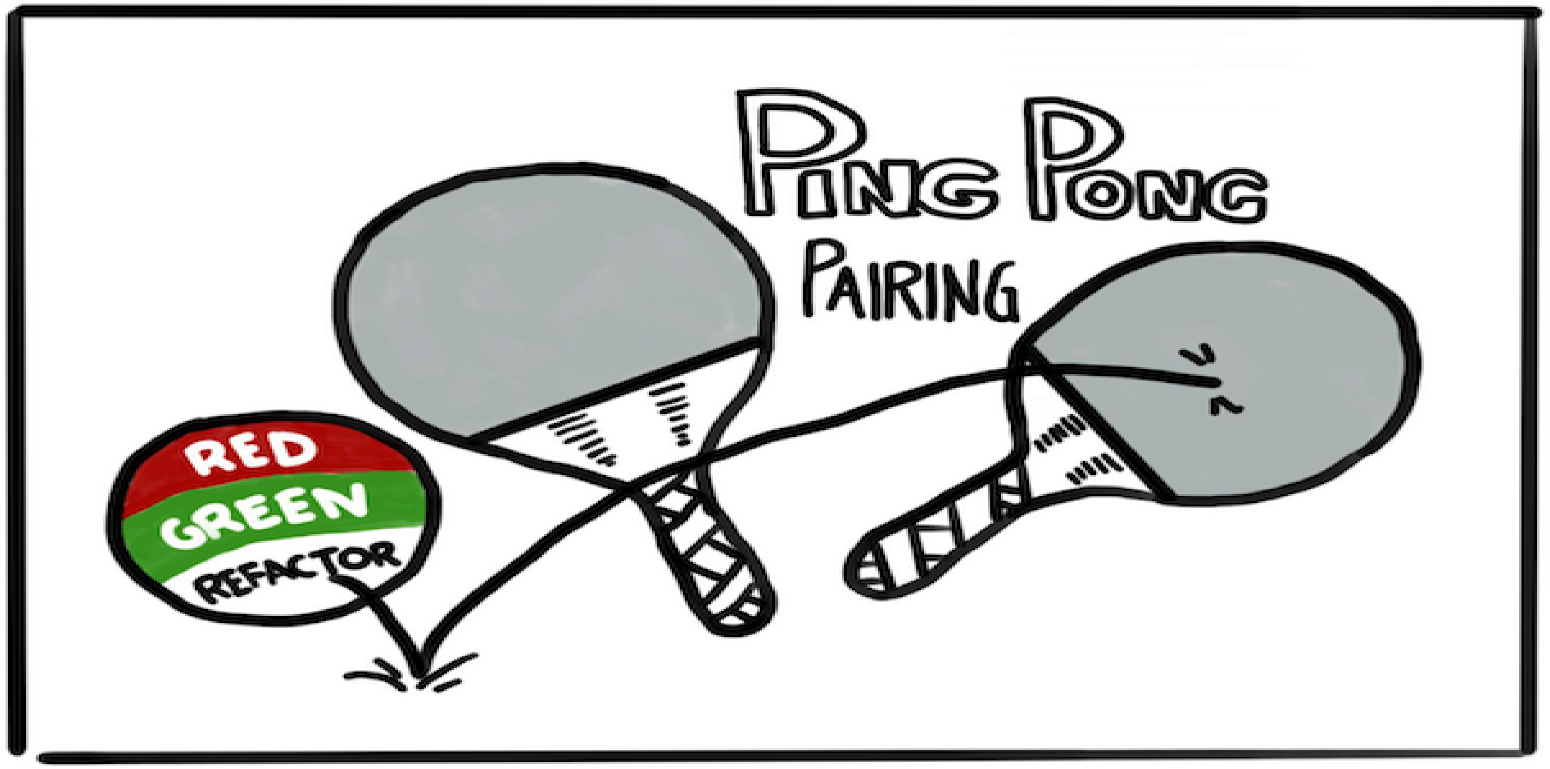
- i. The PI sits down with her student.
- ii. They discuss how the code could better.
- iii. They find and solve issues together.
- iv. The student learns something new.
- v. The code becomes more reusable.

## Pair Programming - Driver and Navigator



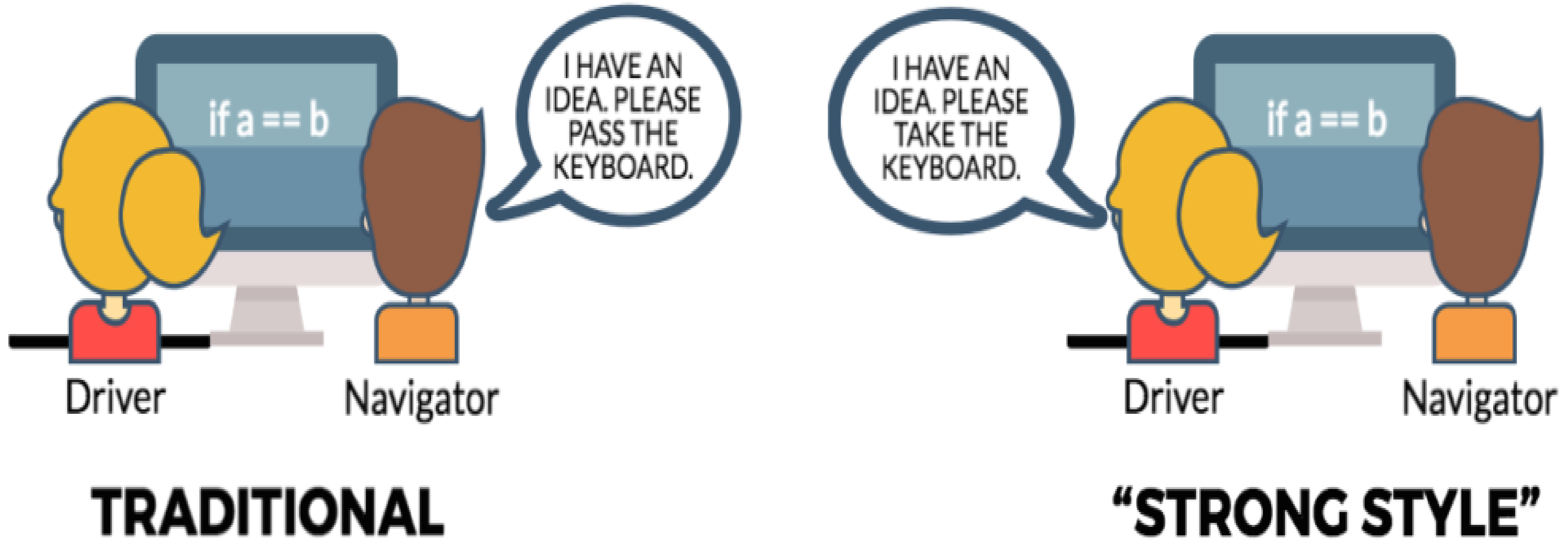
> Martin Fowler, On Pair Programming

## Pair Programming - Ping Pong



> Martin Fowler, On Pair Programming

## Pair Programming - Strong-Style Pairing


if a == b
I HAVE AN IDEA. PLEASE PASS THE KEYBOARD.
Driver
Navigator
TRADITIONAL
if a == b
I HAVE AN IDEA. PLEASE TAKE THE KEYBOARD.
Driver
Navigator
"STRONG STYLE"

> Keith McDonald, Strong Style Pairing

## Pair Programming - Benefits and challenges

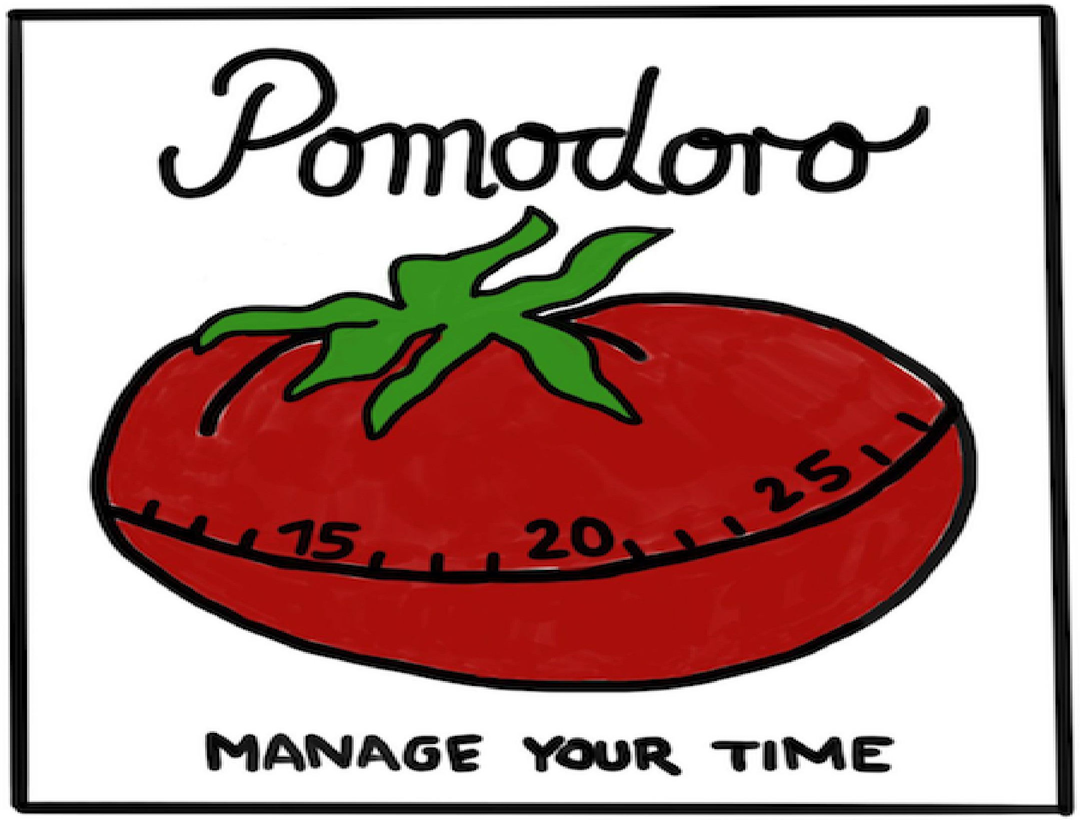
### Benefits

- We can produce better code working together
- We will ship fast for a longer time
- We build a "default" collaborative workflow in the group
- Less interruptions and happier people
- Recovering the flow becomes easier

### Challenges

- ??

## Pair Programming - The Pomodoro technique



> Martin Fowler, [On Pair Programming]

## Synchronous - Group Code Tour

Consider the following scenario:

- i. During a lab meeting, the student presents the steps of her code as logical steps.
- ii. The code is explained both to R programmers and non-experts.
- iii. The group discuss together and improve the code

## Suggestions for the meeting leader

- Keep it a safe environment
- Facilitate participation in the session
- Make it clear that your code isn’t perfect
- Patiently explain when things are not wrong but just not ideal

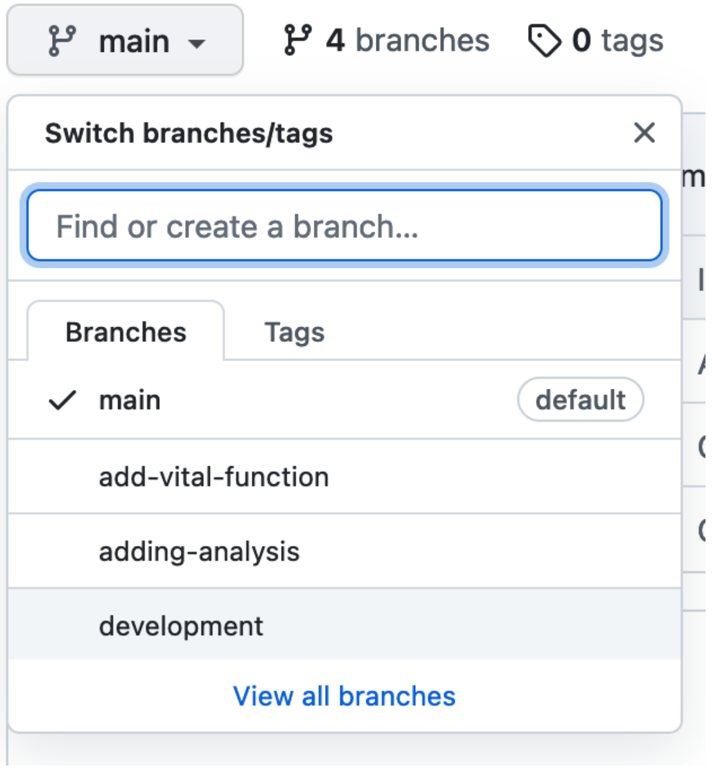
## Asynchronous - I'll get back to you on that

Consider the following scenario:

- i. A postdoc has created a model in Python
- ii. She sends the instructions to test it to her supervisor
- iii. The supervisor can then review it over the next week
- iv. The supervisor makes a PR to the repo with improvements

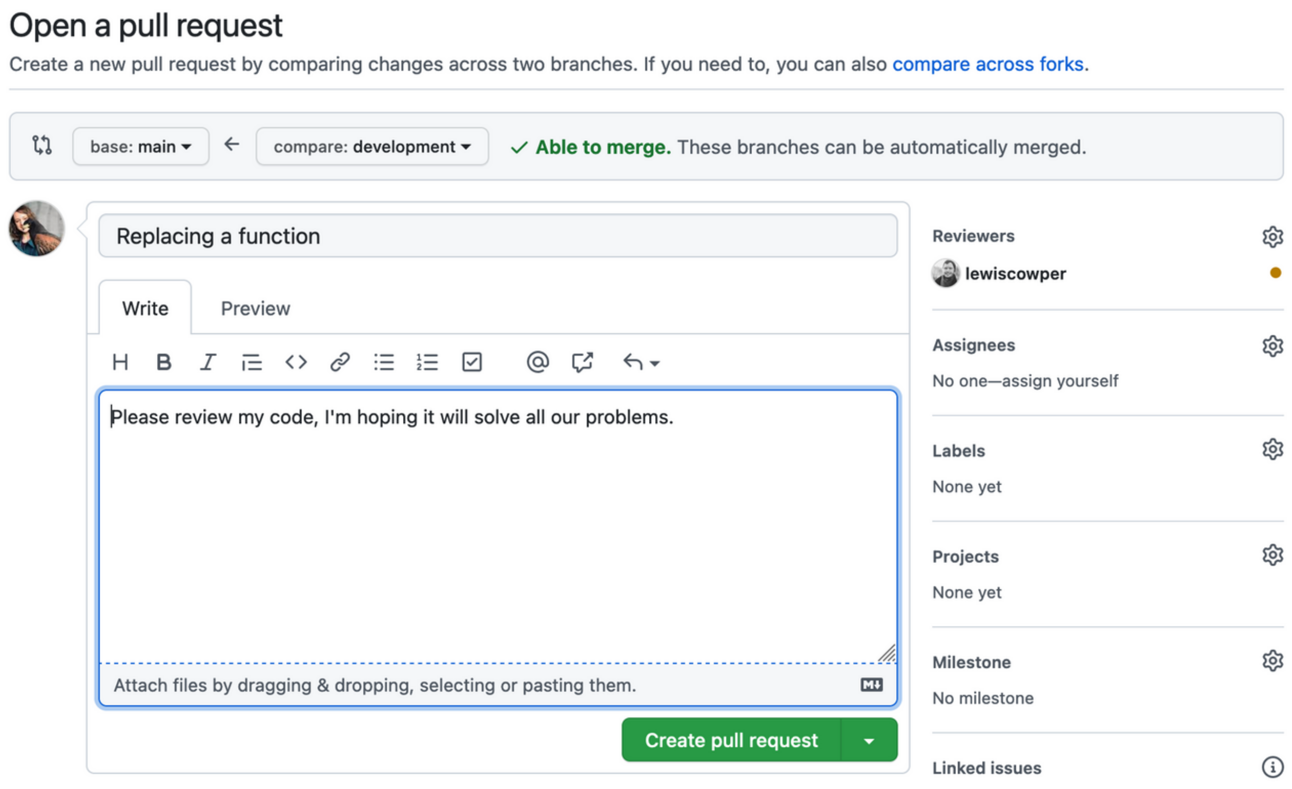
## Branching

> **Branching**: keep a version of the code separate while making experimental changes or keeping track of collaborative work.



## Pull Requests

> **Pull Request**: a code review request prior to merging the changes made on a branch over to the main branch.



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: main ← compare: development ✓ Able to merge. These branches can be automatically merged.

Replacing a function

Please review my code, I'm hoping it will solve all our problems.

Attach files by dragging & dropping, selecting or pasting them.

## Reviewing is not about creating more work

- Part of the scientific process
- An opportunity for everyone to learn better practices
- Reviews are rarely anonymous
- Often public-facing

## Where to put the focus? (I)

- Repetitive code
- Code saying one thing, documentation saying another
- Off-by-one errors
- Making sure each function does one thing only
- Lack of tests and sanity checks
- Magic numbers

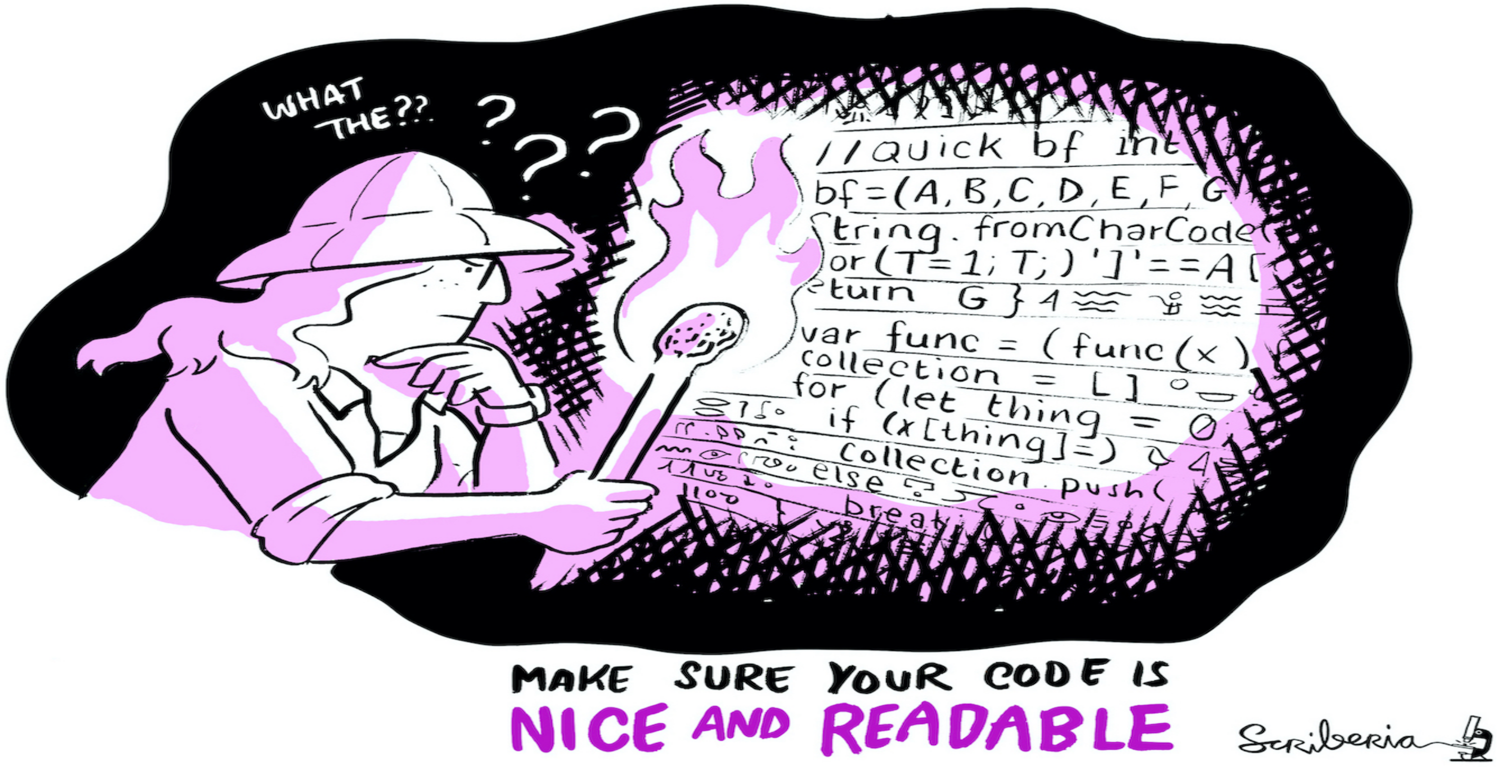
## Where to put the focus? (II)

- Bad variable/method names
- Inconsistent indentation
- The order of the different steps
- Lack of comments and signposting
- Tailor-made and manual steps
- Only works with the given data

*Modified from What to look for when code reviewing*

## Who should do it?

> "Anyone" should be able to perform code reviews.



```
//Quick bf int
bf=(A,B,C,D,E,F,G
String.fromCharCode(
or(T=1;T;)']'==A[
eturn G}
var func = (func(x)
collection = []
for (let thing = 0
if (x[thing]=)
collection.push(
else
break
```

## Benefits - a case study

> - In a group of 11 programs developed by the same group of people, the first 5 were developed without reviews.
> - The remaining 6 were developed with reviews. After all the programs were released to production, the first 5 had an average of 4.5 errors per 100 lines of code.
> - The 6 that had been inspected had an average of only 0.82 errors per 100. Reviews cut the errors by over 80 percent.
>
> Code Complete by Steve McConnell

## Benefits - software developers

- Less time redoing work or refactoring
- Increased productivity
- Greater confidence in own work
- Learning better techniques
- Reduced time debugging alone
- Knowledge exchange and group cohesion

## Benefits - team leaders

- Better understanding of the projects
- Maintainable and better-documented code
- Earlier and better visibility of issues
- Group reviews reduce work burden
- Reusability and modification
- High-quality code that can be released

## To bear in mind

- Should not be used to evaluate individuals
- Revealing mistakes should not come with penalties or shame.
- Should also be done early and often

## Potential obstacles

- Conflicts of interest
- Strong personal views about non crucial matters
- Misunderstandings and/or misinterpretations
- Code ownership
- Psychological safety

## References

This material is based on the Code Review lecture by The Carpentries.

- Code Review by The Carpentries is licensed under CC BY 4.0. Modifications were made in several chapters.
- *The Turing Way* Community. (2021). The Turing Way: A handbook for reproducible, ethical and collaborative research (1.0.1). Zenodo. https://doi.org/10.5281/zenodo.5671094. Code Reviewing Process Chapter.
- Fernando Perez, Code reviews: the lab meeting for code
- Martin Fowler, On Pair Programming