Code Reviews

CSE 403 Software Engineering

Autumn 2023

Thanks to **Apollo Zhu** (UW CSE grad, now at Apple) for allowing us to leverage his deck delivered to Spr23 class


LGTMeow

“Looks Garbage to Me”?

Attribution: hopefully noone

“Let’s Go Team!”

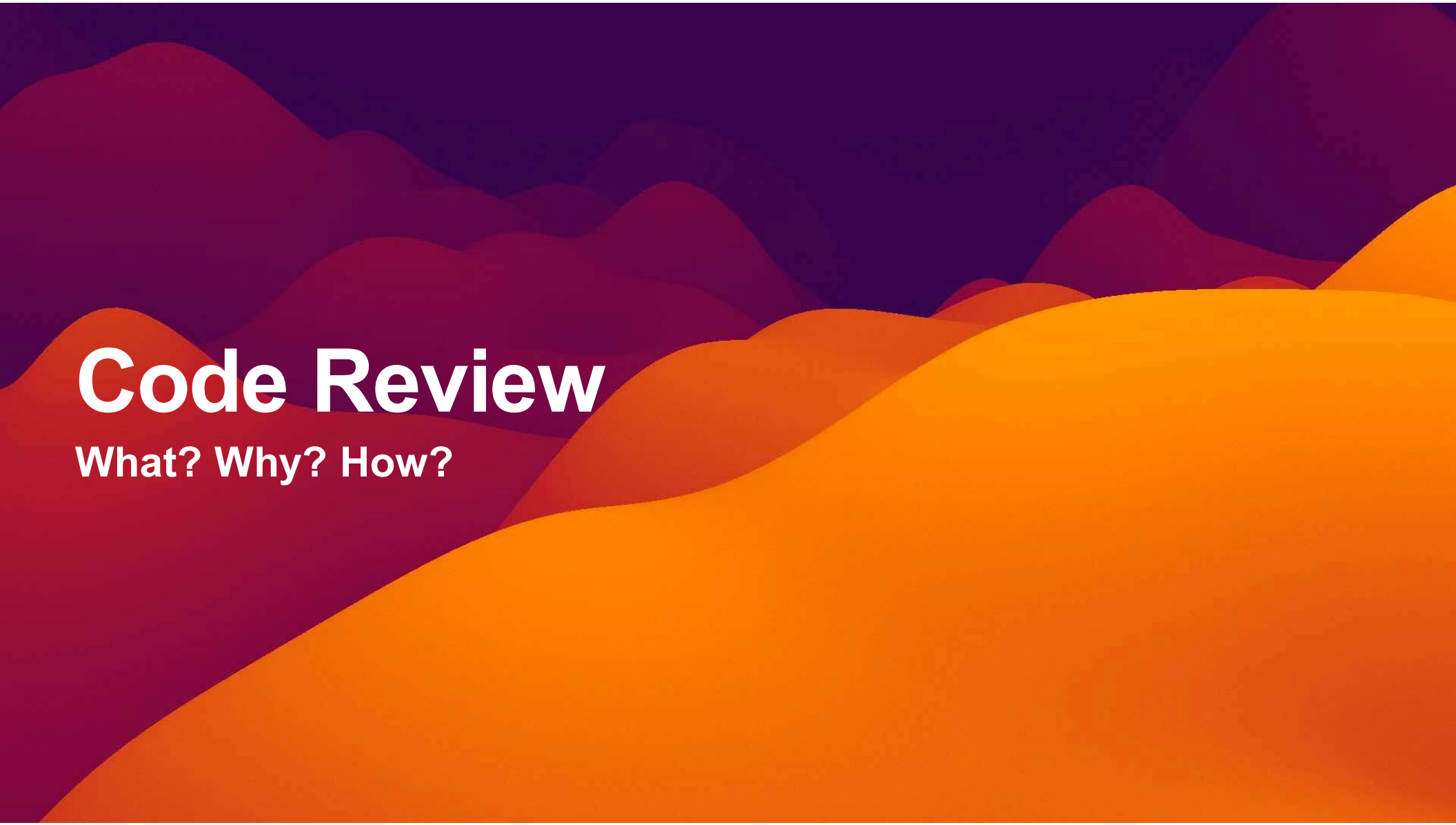
Attribution: an excited engineer with a great attitude

“Let’s Get This Merged”

Attribution: an eager engineer with a literal translation

“Looks Good to Me”

Attribution: an engineer that probably doesn’t want to do code review, or a quick stamp of approval after a thorough code review


Code Review
What? Why? How?

A constructive review of a fellow developer's code.

A required sign-off from another team member before a developer is permitted to check in changes or new code.

A Code Review

Why Code Review

Didn’t we already test?

Why Code Review

Didn’t we already test?

- Average defect detection rates
  - Unit testing: 25%
  - Integration testing: 45%
- 11 programs developed by the same group of people
  - No reviews: average 4.5 errors per 100 LOC

Why Code Review

Didn’t we already test?

- Average defect detection rates
  - Unit testing: 25%
  - Integration testing: 45%
  - Design and code inspections: 55% and 60% <<<<<<<<<!!
- 11 programs developed by the same group of people
  - No reviews: average 4.5 errors per 100 LOC
  - With reviews: average 0.82 errors per 100 LOC <<<<<<<<<!!

Why Code Review

Didn’t we already test?

- IBM's Orbit project
  - 500,000 LOC, 11 levels of inspections
  - Delivered early with 1% of the predicted errors
- After AT&T introduced reviews
  - 14% increase in productivity and a 90% decrease in defects

Steve McConnell, *Code Complete2*

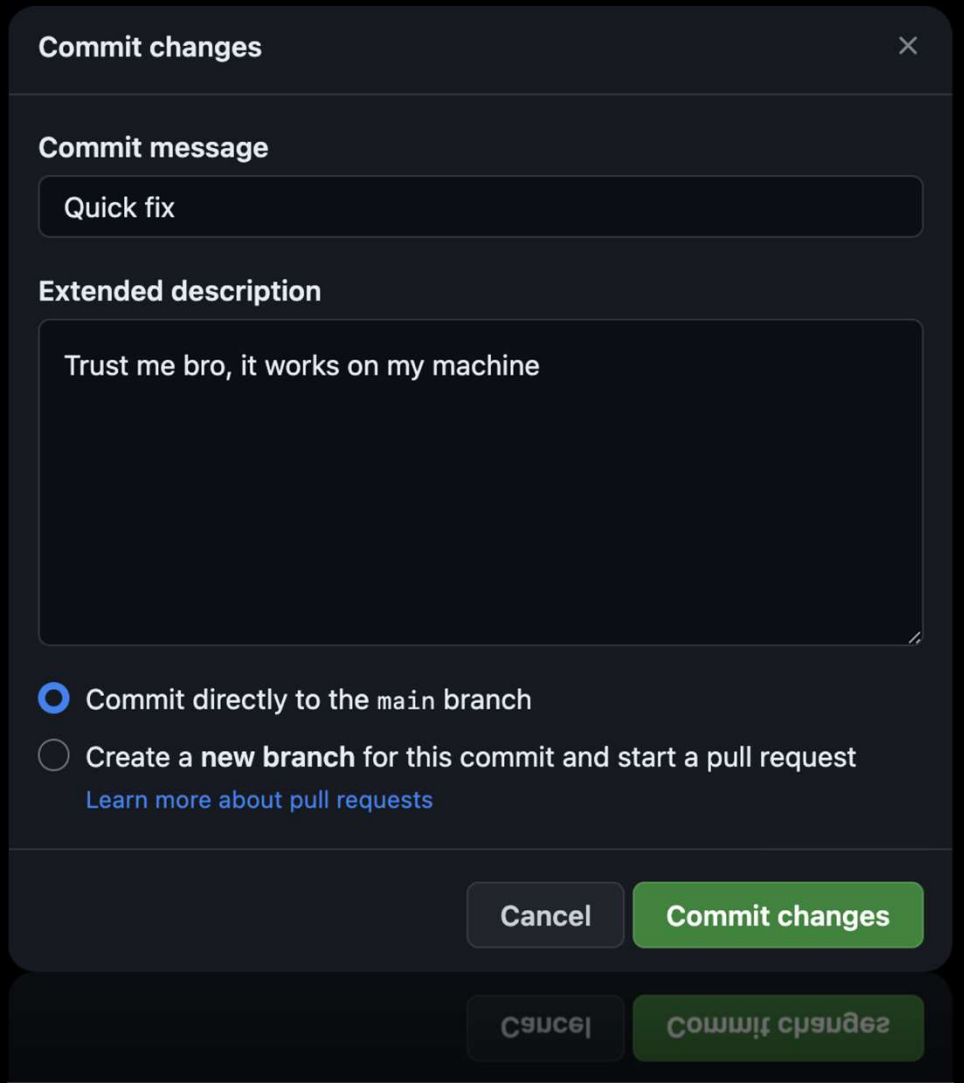
"***All code that gets submitted needs to be reviewed by at least one other person***, and either the code writer or the reviewer needs to have readability in that language. Most people use Mondrian [now Critique] to do code reviews, and obviously, we spend a good chunk of our time reviewing code."

Amanda Camp, Software Engineer, Google

Resource: Google Code Review Developer Guide
https://google.github.io/eng-practices/review/

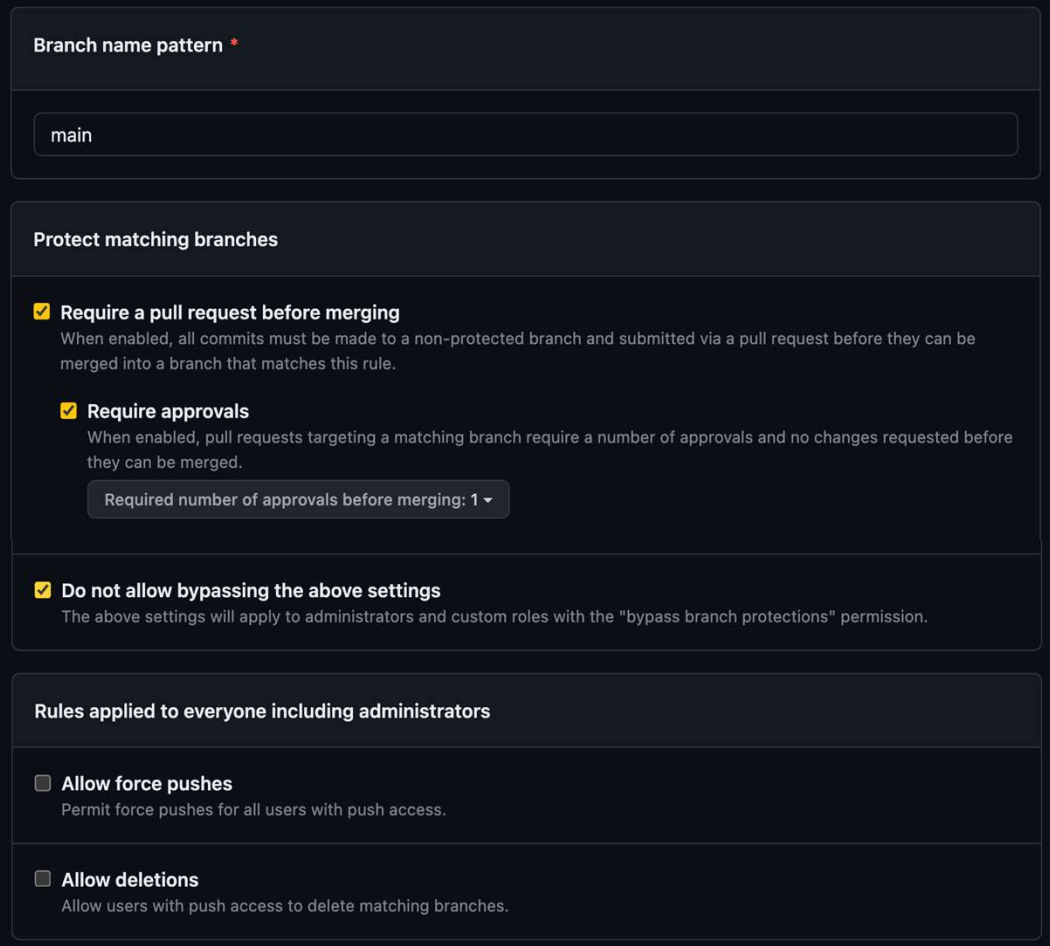
“What could go wrong?”

Famous last words



Add Branch Protection

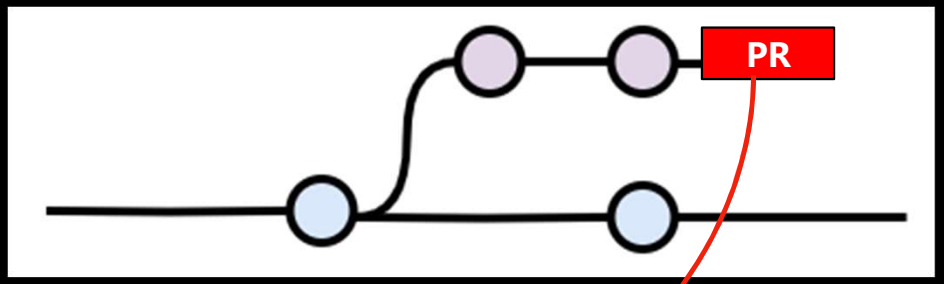
Require a pull request

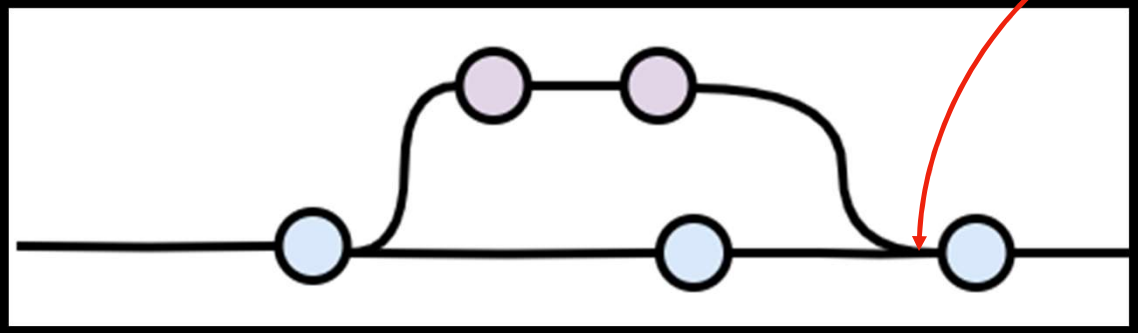


New Workflow - Happy Path

Dev-1

- New Branch
- Commit, Push, (Repeat)
- Open Pull Request



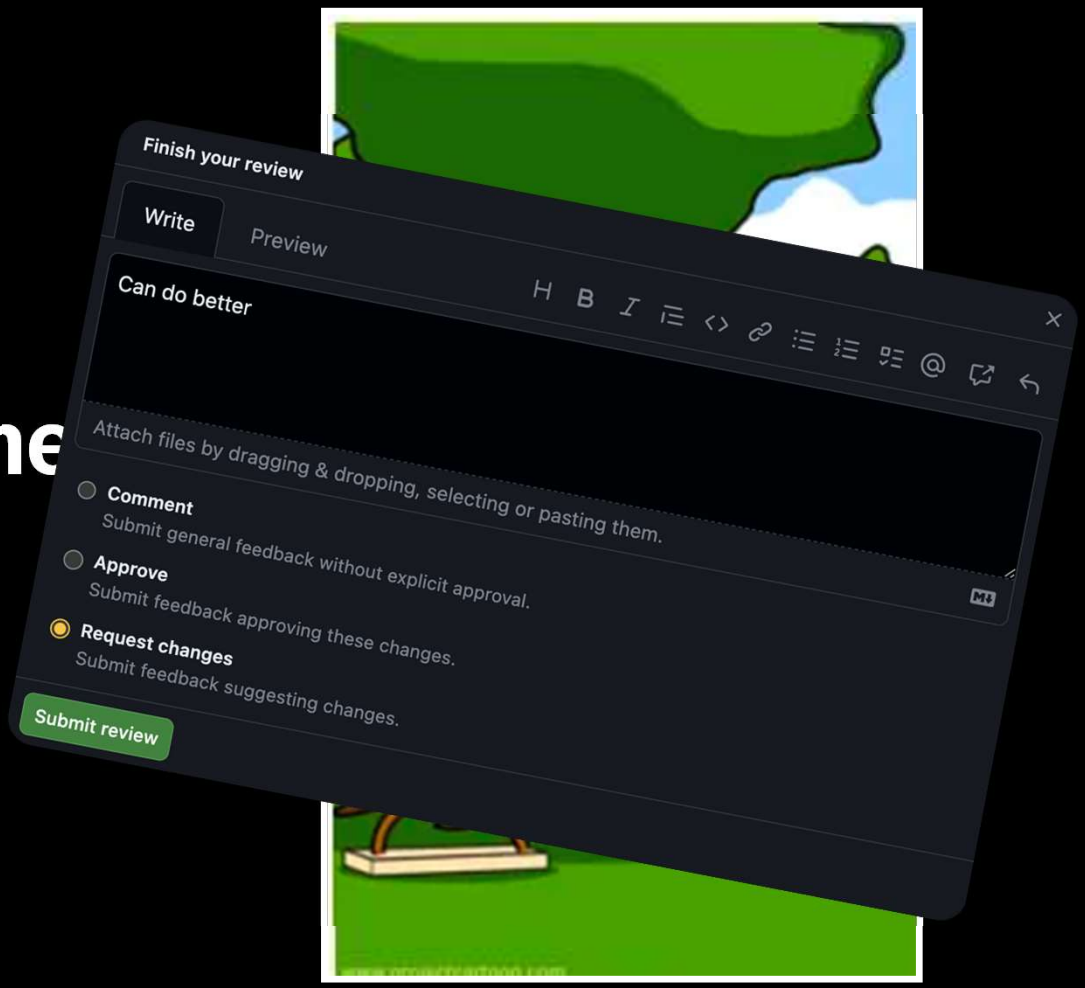


Dev-2

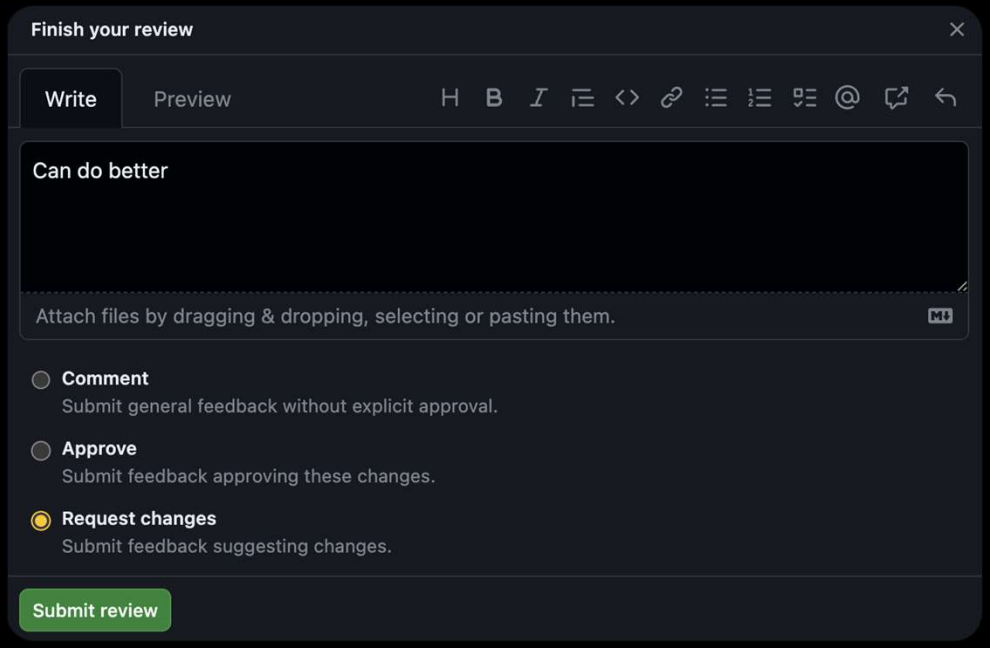
- Code Review
- Merge

How the programmer wrote it

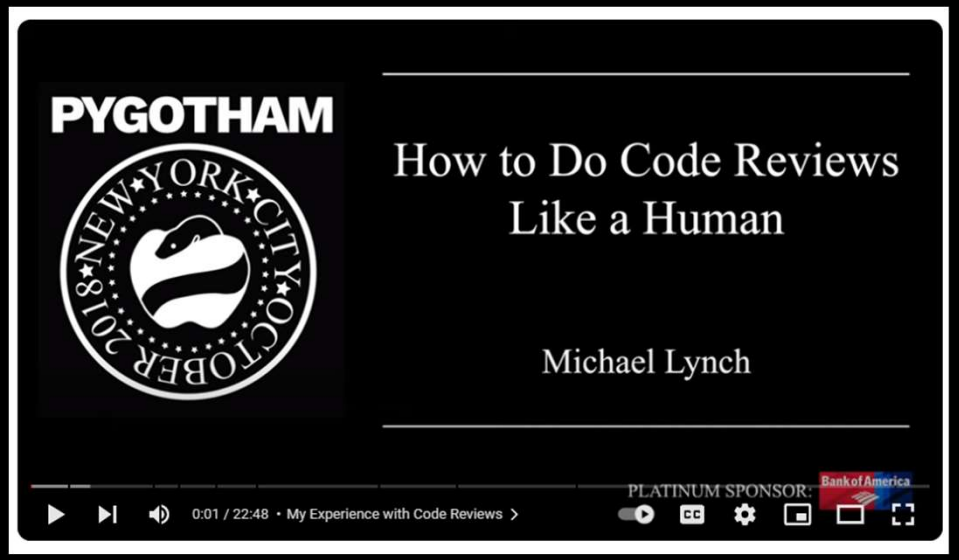
How the programme wrote it



See any issues with this?



CSE 403 Required Watching



How to Do Code Reviews Like a Human – YouTube
https://www.youtube.com/watch?v=0t4_MfHgb_A

Code review like a human

- Define a style guide as a team
- Let computers do the boring parts: linters/formatters (and CI)
- Give code examples instead of “possible change requests” (build trust)
- Never say “YOU” (focus on the code, not the coder!): we == team ownership
- Requests not commands… frame it as an in-person conversation
- Add sincere positive praises
- Incremental improvements instead of perfection
- Handle stalemates proactively: talk it out, design review?, concede or escalate

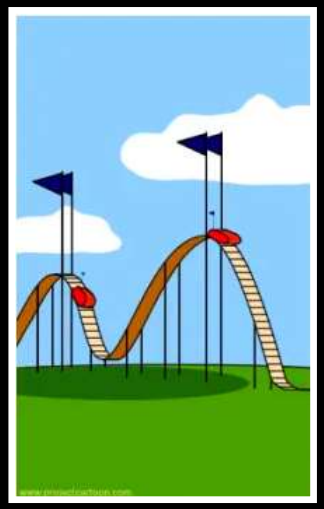
"At Facebook, we have an internally-developed web-based tool to aid the code review process. Once an engineer has prepared a change, she submits it to this tool, which will notify the person or people she has asked to review the change, along with others that may be interested in the change -- such as people who have worked on a function that got changed.

At this point, the reviewers can make comments, ask questions, request changes, or accept the changes. ***If changes are requested, the submitter must submit a new version of the change to be reviewed.*** All versions submitted are retained, so reviewers can compare the change to the original, or just changes from the last version they reviewed. Once a change has been submitted, the engineer can merge her change into the main source tree for deployment to the site during the next weekly push, or earlier if the change warrants quicker release."

Ryan McElroy, Software Engineer, Meta

What are we reviewing?

- Verification: are we building the system right?
- Validation: are we building the right system?



What the PR included

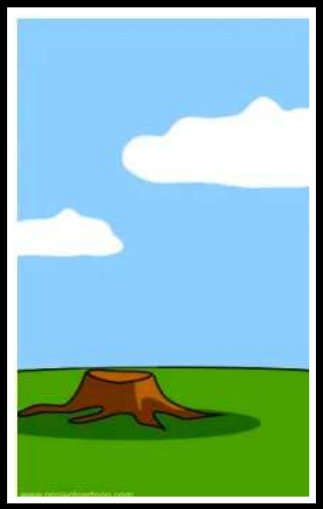
What the customer really needed

What are we reviewing?

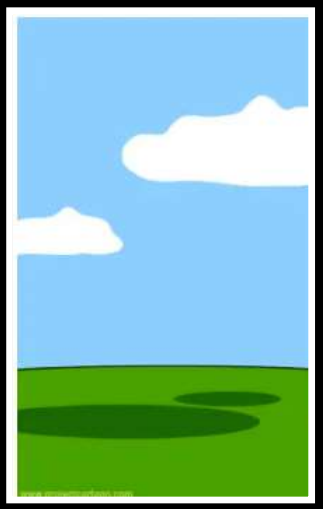
- Verification: are we building the system right?
- Validation: are we building the right system?

- Presence of good properties?
- Absence of bad properties?

- Identifying errors?
- Confidence in the absence of errors?



How much the tests covered



How the code was documented

What are we reviewing?

- Verification: are we building the system right?
- Validation: are we building the right system?

- Presence of good properties?
- Absence of bad properties?

- Identifying errors?
- Confidence in the absence of errors?

- Robust? Safe? Secure? Available? Reliable?
- Understandable? Modifiable?
- Cost-effective?
- Usable?

Leverage code review checklists

Consider if -

- The code is well-designed.
- The functionality is good for the users of the code.
- Any UI changes are sensible and look good.
- Any parallel programming is done safely.
- The code isn't more complex than it needs to be.
- The developer isn't implementing things they *might* need in the future but don't know they need now.
- Code has appropriate unit tests.
- Tests are well-designed.
- The developer used clear names for everything.
- Comments are clear and useful, and mostly explain *why* instead of *what*.
- Code is appropriately documented (generally in g3doc).
- The code conforms to your style guides.

Make sure to **review every line of code** you've been asked to review, look at the context, make sure you're **improving code health**, and **compliment developers** on good things that they do.

Resource: Google What to look for in a code review

https://google.github.io/eng-practices/review/reviewer/looking-for.html

Are there other benefits of code reviews?

- Praise
- Mentoring, teaching
- Learning
- Comradery, bonding

Code Review Exercise

```
public class Account {
    double principal,rate; int daysActive,accountType;
    public static final int STANDARD = 0, BUDGET = 1,
                            PREMIUM = 2, PREMIUM_PLUS = 3;

    public static double calculateFee(Account[] accounts)
    {
        double totalFee = 0.0;
        Account account;
        for (int i=0;i<accounts.length;i++) {
            account=accounts[i];
            if(account.accountType == Account.PREMIUM ||
               account.accountType == Account.PREMIUM_PLUS)
                totalFee += .0125 * (   // 1.25% broker's fee
                        account.principal * Math.pow(account.rate,
                        (account.daysActive / 365.25))
                        - account.principal);   // interest-principal
        }
        return totalFee;
    }
}
```

Code Review Exercise

Let’s review the review

https://github.com/zhuzhiyu-1962988/CSE403/pull/3

“At Yelp we use review-board. An engineer works on a branch and commits the code to their own branch. The reviewer then goes through the diff, adds inline comments on review board and sends them back. ***The reviews are meant to be a dialogue***, so typically comment threads result from the feedback. Once the reviewer's questions and concerns are all addressed they'll click "Ship It!" and the author will merge it with the main branch for deployment the same day.”

Alan Fineberg, Software Engineer, Yelp

Making a Good Pull Request

Think like a reviewer

- Use descriptive but concise title and summary
- Describe context, rationale, and alternatives considered
- Link to relevant resources (specs, issues/bug tracker, previous PR)
- Provide screenshots/recordings for UI changes

Resources

How to write the perfect pull request
https://github.blog/2015-01-21-how-to-write-the-perfect-pull-request/

Writing good CL descriptions
https://google.github.io/eng-practices/review/developer/cl-descriptions.html

Agree on a plan

How and where?

- Online/electronic
- In-person meeting
  - Best to prepare beforehand: artifact is distributed in advance
  - Preparation is critical and usually identifies more defects than the meeting

Who participates?

- One other developer
- A group of developers (especially for a design review)

What is reviewed?

- A specification
- A coherent module (e.g., checklist-style “inspection")
- An entire component ("holistic review”)
- A single code commit or PR ("incremental review”)

Agree on a plan

When:

- What's the expected review time frame?

Who submits after approval?

- LGTM and "auto-submit" -> reviewer submits
- Approval plus comments -> author submits

Agree on a plan for CSE 403

How and where?
- ***Online/electronic***

Who participates?
- ***One other developer***

What is reviewed?
- ***A single code commit or PR ("incremental review")***

What's the expected review time?

Who submits after approval?

Project requirements

Team Policy

“Can someone review this PR please? Thanks”

A message that you’ll likely see in Slack

YES

WHO?

WHEN?

Your Action Items

- Figure out code review logistics with the team
- Start using pull requests and doing code reviews
- Enforce code review through branch protection (may not be available with free github)
- Automate the code review process with CI checks
- Lean on coding guidelines for format disagreements
- Keep feedback constructive


Questions?