Reviewing Code

Code reviews mean a lot of different things to different people as companies will do things differently, but there is some basic etiquette and guiding principles – whether we are having our code reviewed, or reviewing anyone else's:

https://github.com/thoughtbot/guides/tree/main/code-review

Watched this video in our breakout rooms:

https://www.youtube.com/watch?v=PJjmw9TRB7s

My notes on the video:

- Code reviews aren't just to catch bugs (but they are there to catch bugs too!)
- They are a way to get better at your job
- Chief benefits:
    - Knowledge transfer
    - Increased Team Awareness
    - Finding alternative Solutions
- Code review is the discipline of discussing your code with your peers
- Like so many other things, it's all about communication!
- 2 key rules of engagement:
    - As an author: **Provide sufficient context**
        - "If content is king, then context is God" (Gary Vaynerchuck – when talking about something completely unrelated to code reviews!)
        - Context is: "what problem are you solving?"
        - "Use type column first in multi-column indexes" as a pull request/comment does not give you the "why" it was done?
        - Don't just answer with "Fixes #1337" or reference a ticket!
        - Then gave an example with 2 paragraphs of context that explained why and referenced PostgreSQL docs as well as MySQL docs
        - These were docs quoted in the context rather than just linked, which means they live on in the commits and are date stamped
        - Even if you are just refactoring – provide the context on why!
    - As a reviewer: **Ask questions rather than making demands** ("Ask, don't tell")
        - Ask questions rather than make demands
        - Remember it can be difficult to communicate tone of voice online
        - "Extract a service to reduce some of this duplication" is a command. Owner might just ignore you and doesn't give any credit that the author may have already considered it and decided against it!

        - "What do you think about extracting a service to reduce some of this duplication?"
        - Try to open up a conversation:
            - "What do you think about…?"
            - "Did you consider…?"
            - "Can you clarify…?"
        - You can then go on to describe your suggestion for improvement if needed
        - Be careful that questions don't become silent judgments! -
            - Don't use "Why didn't you just…?"
            - Is "Why didn't you…?" any better? Probably not.
        - So be positive with your questions! (Socratic method)
- Conflict
    - Conflict is good to drive change, but needs to be healthy!
    - Agree how to handle conflict in code reviews (make sure the whole team is on board)
    - Remember not to argue about trade-offs (make it about quality rather than just the way you would have done it)
- What to review
    - 10 minutes is a long time to spend on a review, so changes should be small so they can be reviewed quickly
        - Single responsibility principle (S from SOLID acronym) – does it do one job
        - Naming – 2 hard problems in coding – naming things and cache invalidation (and timezones!) – make it semantic!
        - Complexity – are there areas of the code where it looks complex (and break out the "Can you clarify?" question
        - Test coverage – looking for bugs in code review is important but it doesn't give you Quality Assurance, that's what tests are for!
        - Duplication – DRY – Don't repeat yourself
        - Then specific things to the job you're working on, or that play to your strengths
- What about style?
    - Codebases should be clean (like a kitchen – everything in its place etc.)
    - People can feel negatively about people who comment on style, but it is important! To avoid problems:
        - Agree a style guide as a team
        - Outsource it (putting it in to auto linter or a bot)
- Benefits of a Strong Code Review Culture:
    - Better code (by being able to discuss and understand code)
    - Better developers (by taking best from each developer)
    - Team Ownership (it's not "yours" or "mine", it's "ours")

    - Healthy debate – you can agree to disagree! Don't silently seethe!
- Reviewing is another kind of pairing! (Although often asynchronous as not normally in real time!)
    - Doesn't have to be async, but can make the request have to stand on its own feet
- Do code reviews before Quality Assurance (QA) – try to make their life boring!
- Junior devs should review Senior devs code as well!