How to Make Your Code Reviewer Fall in Love with You

Overview

A cover image and series of cartoons for a follow-up article to [“How to to Code Reviews Like a Human.”](#)

The cover image should be

Deadline

Late October, but flexible.

Rights

In exchange for final payment, Michael Lynch assumes full copyright of the images.

Level of Detail

For the cover image, I’d like to stick to the level of detail we’ve had for the last few covers.

For the in-article illustrations, I’d like them to be simpler but still a little bit more detailed than the original “How to do Code Reviews Like a Human.” This is a good level of detail:

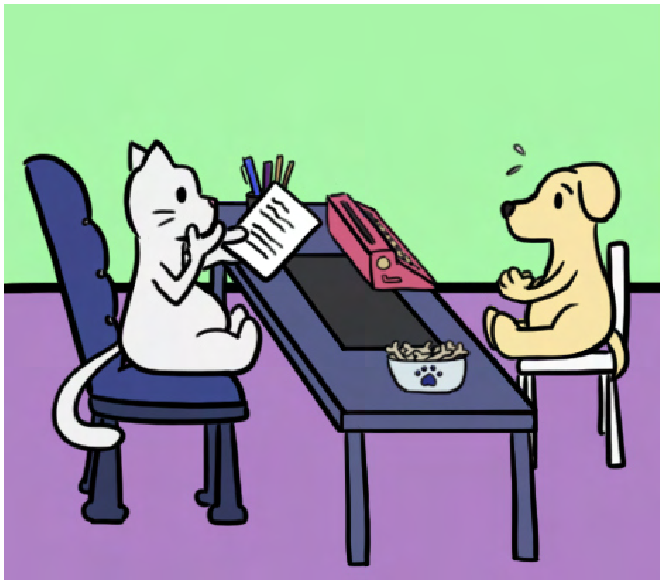
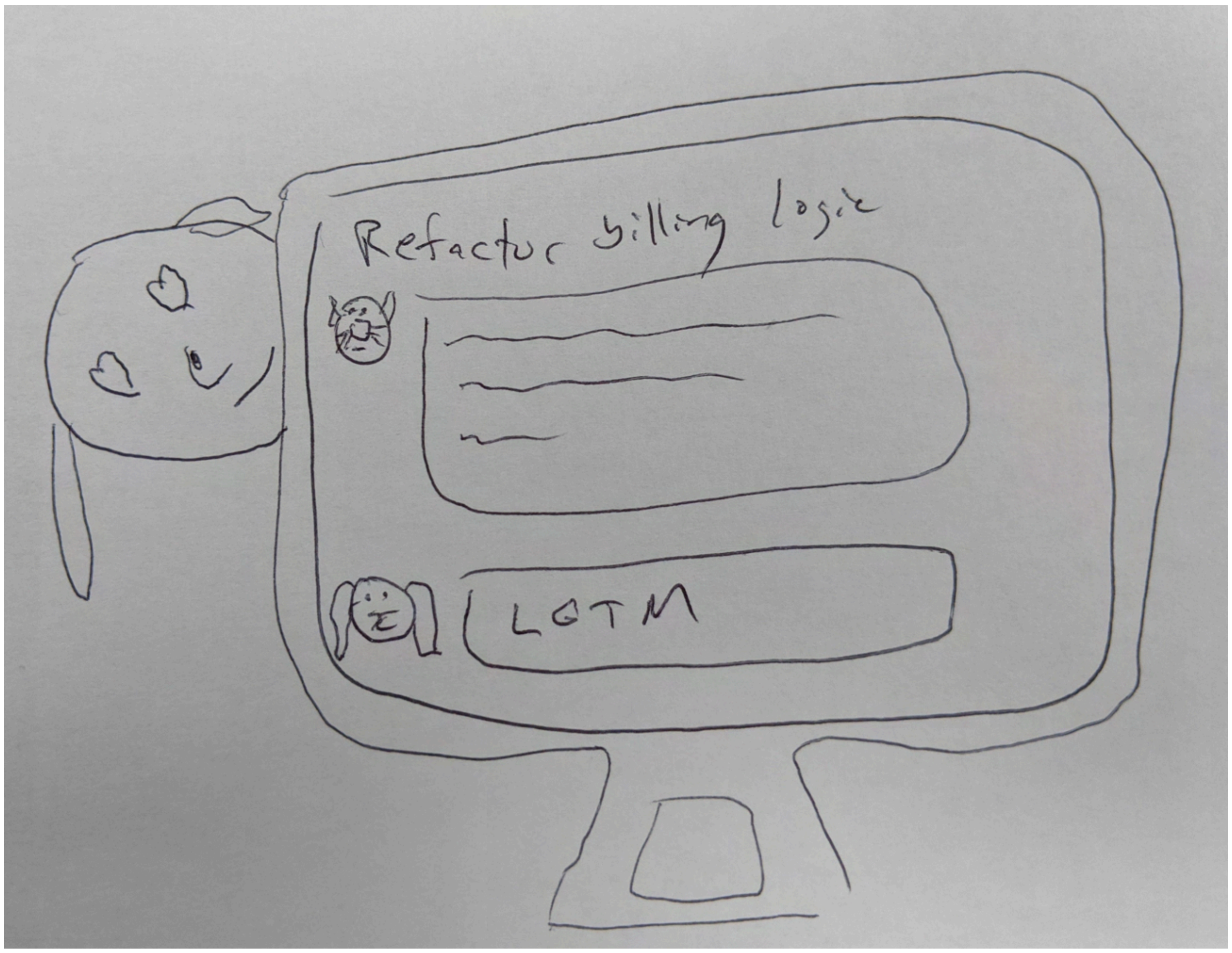


Image #1: Cover Photo



We’re seeing a computer monitor from the cat’s perspective.

On the monitor, the title is “Refactor billing logic.” There is text next to a cat avatar that’s illegible. Below it, the dog has just approved a code change with an “LGTM” stamp (LGTM=”looks good to me”).

Behind the monitor, we see the dog peering out from behind its own monitor. The dog has cartoon hearts over his eyes and/or floating around its head because it has fallen in love with the cat based on reviewing the cat’s code. The dog peering over should be 80% cute, 20% creepy.

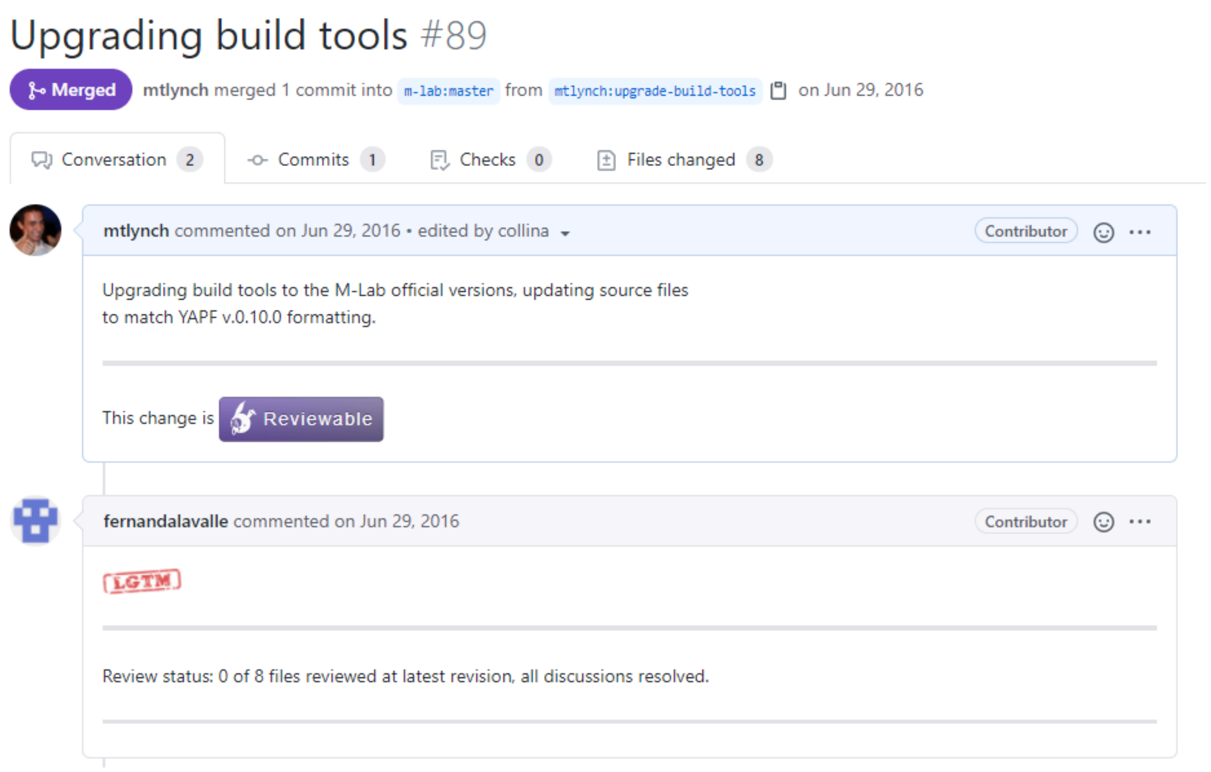
Can we do two versions? One with the text:

- How to Make Your Code Reviewer Fall in Love with You

And one with no title text. The one with title text should be 1200x628 px, but the title-free one can be different dimensions to crop out unneeded space.

Sidenote: The cover image strategy I've settled on is to display the version with text when I share the article on social media so that the title stands out more. I'll embed the other version in the article itself, where it doesn't need a prominent title because the title's already there.

Reference Images


Upgrading build tools #89

Merged mtlynch merged 1 commit into m-lab:master from mtlynch:upgrade-build-tools on Jun 29, 2016

mtlynch commented on Jun 29, 2016 • edited by collina

Upgrading build tools to the M-Lab official versions, updating source files to match YAPF v.0.10.0 formatting.

This change is Reviewable

fernandalavalle commented on Jun 29, 2016

Review status: 0 of 8 files reviewed at latest revision, all discussions resolved.

The monitor should show a screen that looks like a simplified version of Github's code review screen.

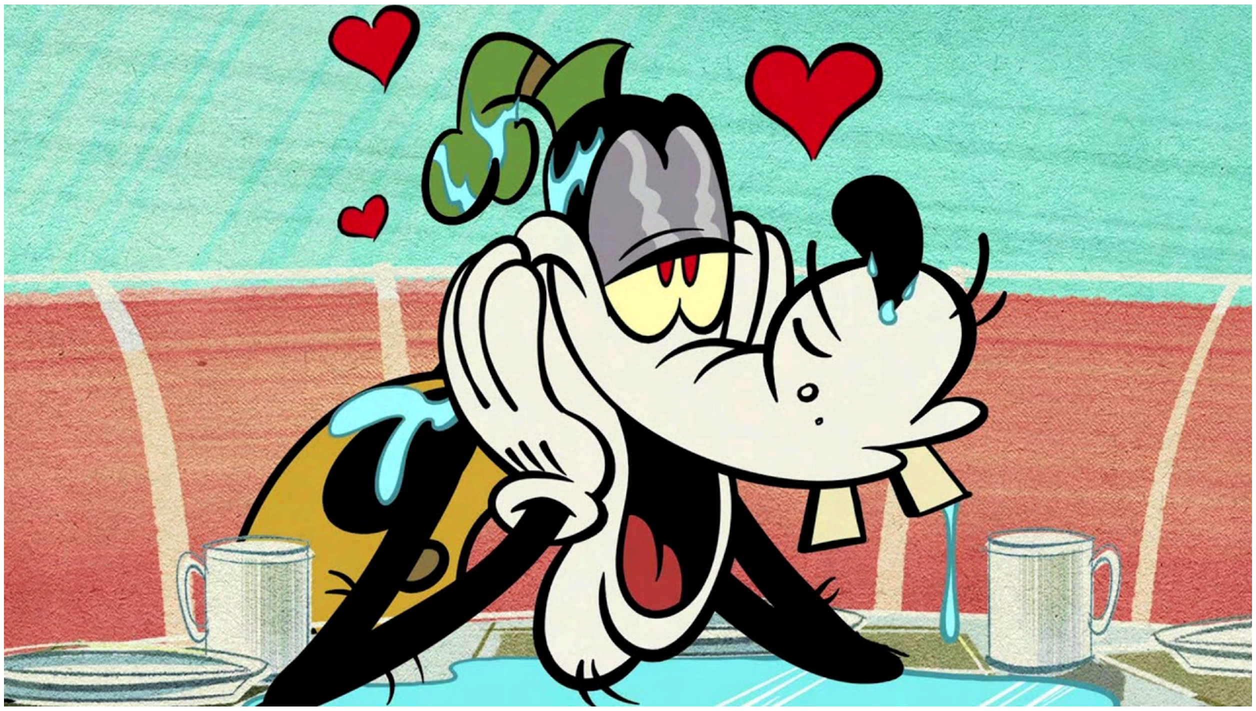
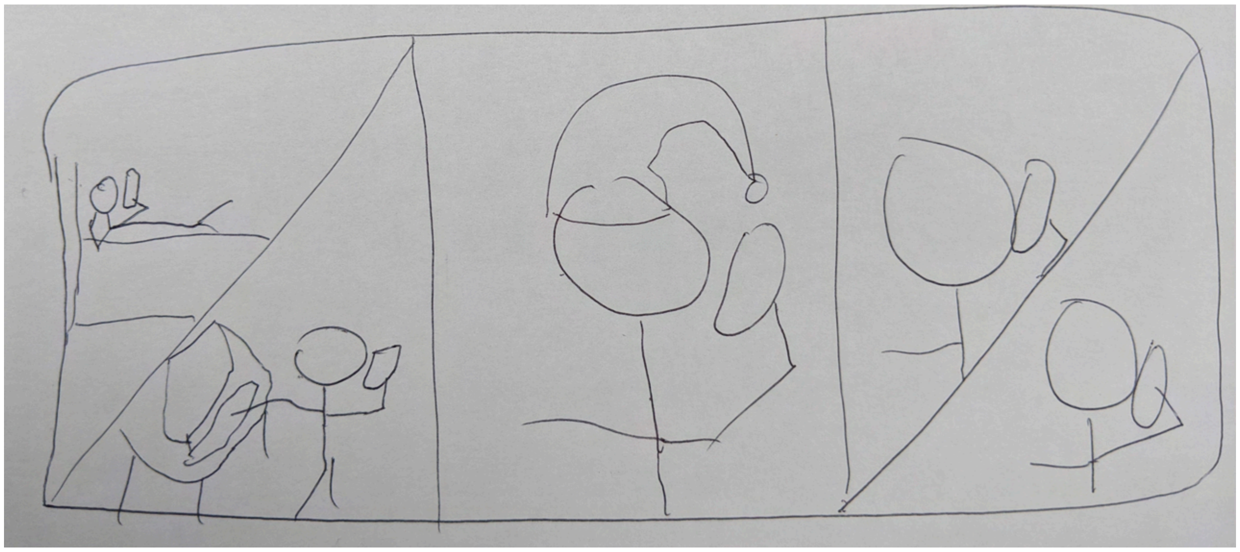

A dog in love (but ours should look less gross).

Image #2: Late night question

Size: Flexible



Frame 1
The frame is split (I used diagonal, but I'm not married to that).

In the first half, the dog is waking up in the middle of the night to answer a phone call. The dog looks sleepy and confused. The dog is wearing the same style pajama hat that we used in, "What I've Been Doing Since Quitting My Job."

The cat is in an office somewhere far away where it appears to be normal business hours.

> Dog: Hello?
> Cat: When you wrote `bill.py` six years ago, why'd you make t=6?

Frame 2
The dog looks excited.

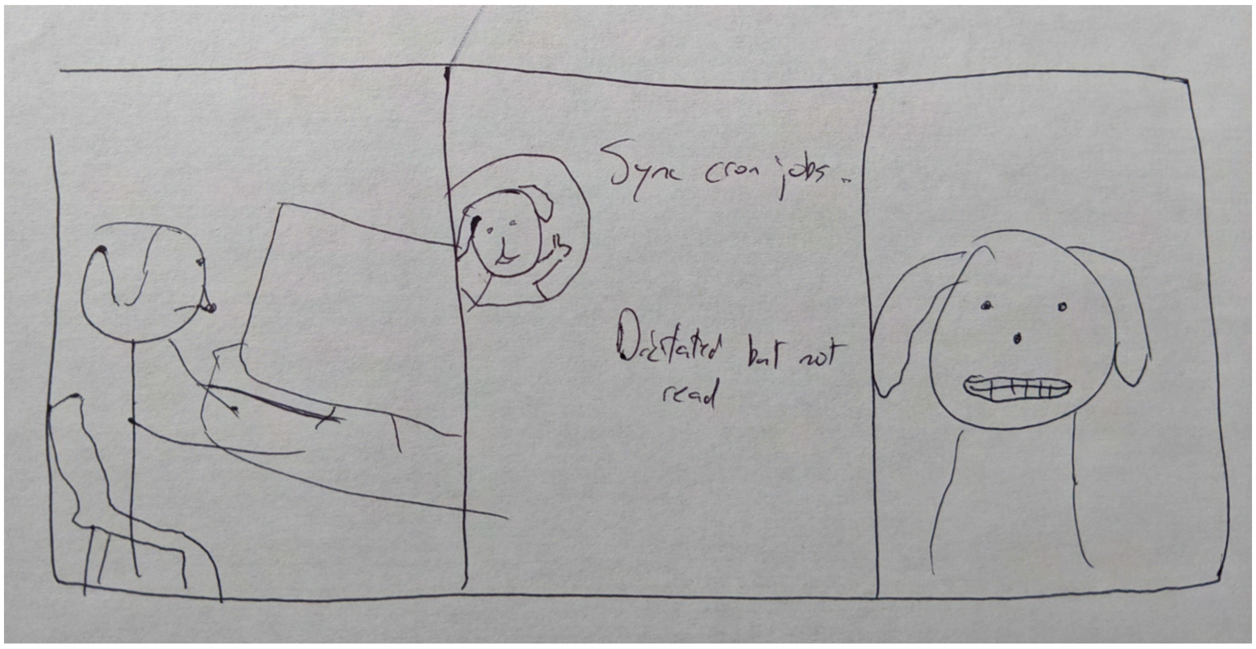
> Dog: I'm glad you called! It's because sales tax is 6%.
> Cat: (through the phone) Of course!

Frame 3

> Dog: This is a good way to communicate implementation choices.

Both are nodding and smiling.

Image #3: What idiot wrote this?



Frame 1
Dog is reading code on their computer. The code is illegible but has the general “shape” of code.

Dog: What idiot wrote this?

Frame 2
Computer screen shows a change request that reads:

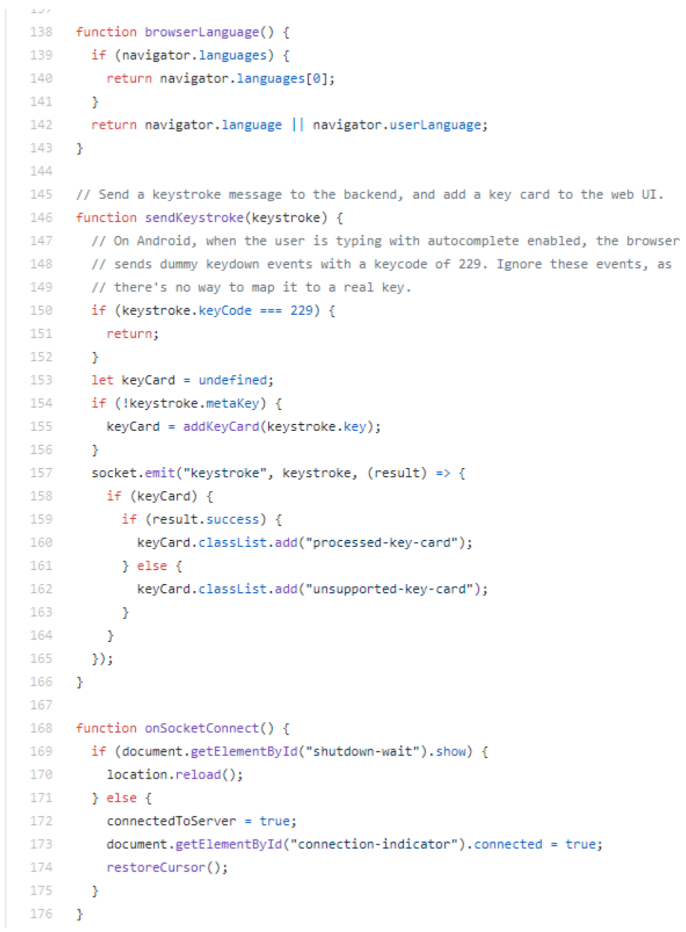
Sync cron jobs to the lunar cycle

Dictated but not read

The avatar next to the change shows the dog giving a thumbs up

Frame 3
Dog is grimacing as it realizes that it was the one responsible.

Reference images

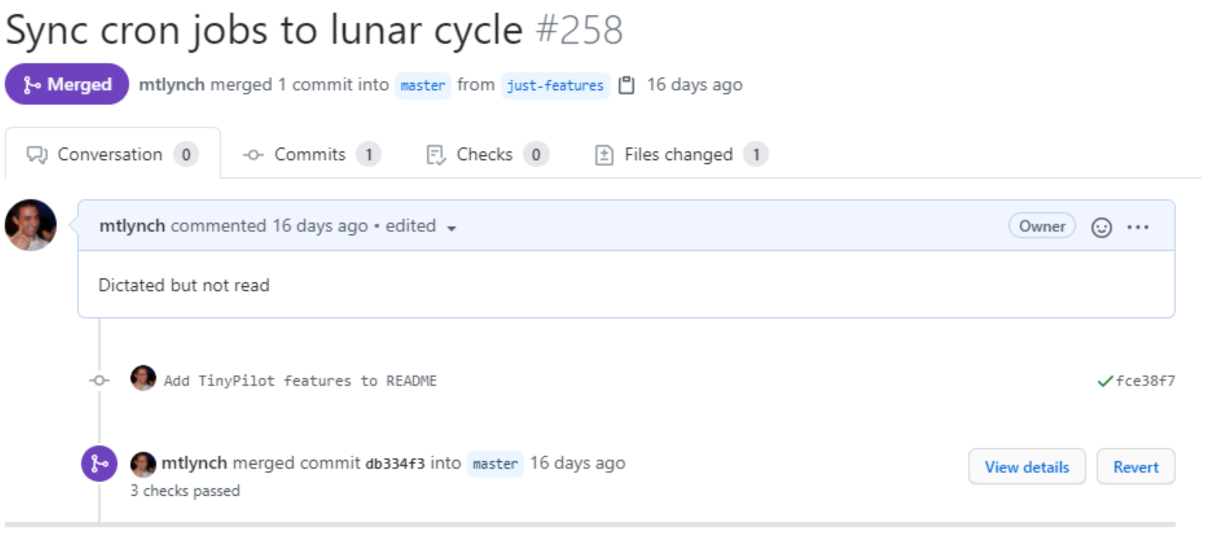


```
function browserLanguage() {
  if (navigator.languages) {
    return navigator.languages[0];
  }
  return navigator.language || navigator.userLanguage;
}

// Send a keystroke message to the backend, and add a key card to the web UI.
function sendKeystroke(keystroke) {
  // On Android, when the user is typing with autocomplete enabled, the browser
  // sends dummy keydown events with a keycode of 229. Ignore these events, as
  // there's no way to map it to a real key.
  if (keystroke.keyCode === 229) {
    return;
  }
  let keyCard = undefined;
  if (!keystroke.metaKey) {
    keyCard = addKeyCard(keystroke.key);
  }
  socket.emit("keystroke", keystroke, (result) => {
    if (keyCard) {
      if (result.success) {
        keyCard.classList.add("processed-key-card");
      } else {
        keyCard.classList.add("unsupported-key-card");
      }
    }
  });
}

function onSocketConnect() {
  if (document.getElementById("shutdown-wait").show) {
    location.reload();
  } else {
    connectedToServer = true;
    document.getElementById("connection-indicator").connected = true;
    restoreCursor();
  }
}
```

The screen of code in frame 1 should look kind of like this, with indentation and curly braces that look like code, but the code itself should be illegible.


Sync cron jobs to lunar cycle #258

Merged mtlynch merged 1 commit into master from just-features 16 days ago

Conversation 0 Commits 1 Checks 0 Files changed 1

mtlynch commented 16 days ago • edited Owner

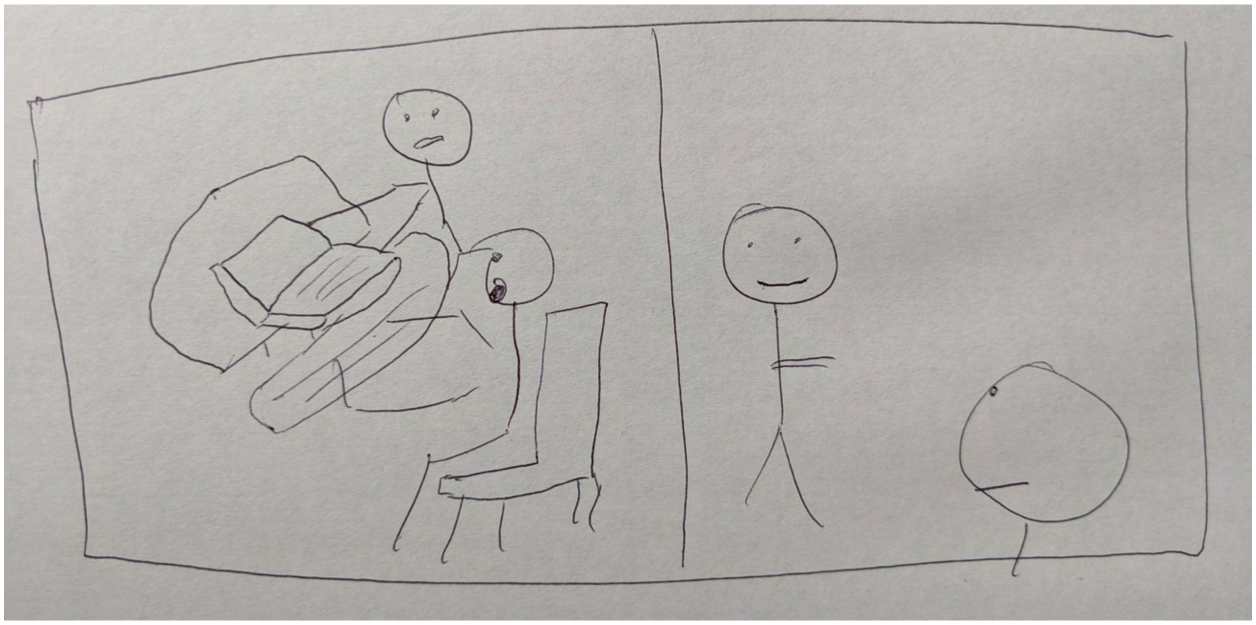
Dictated but not read

Add TinyPilot features to README ✓fce38f7

mtlynch merged commit db334f3 into master 16 days ago
3 checks passed

The change history screen in frame 2 should look like a simplified Github change history screen with no reviewer.

Image #4: Don't bother the compiler



Frame 1
The cat is working at their computer, when the dog suddenly shoves a laptop in front of their screen. The cat seems startled. The dog looks oblivious and helpless.

Dog: Can you verify that my code syntax is correct?

If it’s possible to show legible text on the dog’s laptop screen, it should be:

```
if (true)) {
  printf("Your code compiled!\n");
}
```

(the imbalanced parentheses are intentional))

But it can just be illegible text on both screens if it’s hard to fit legible text.

Frame 2

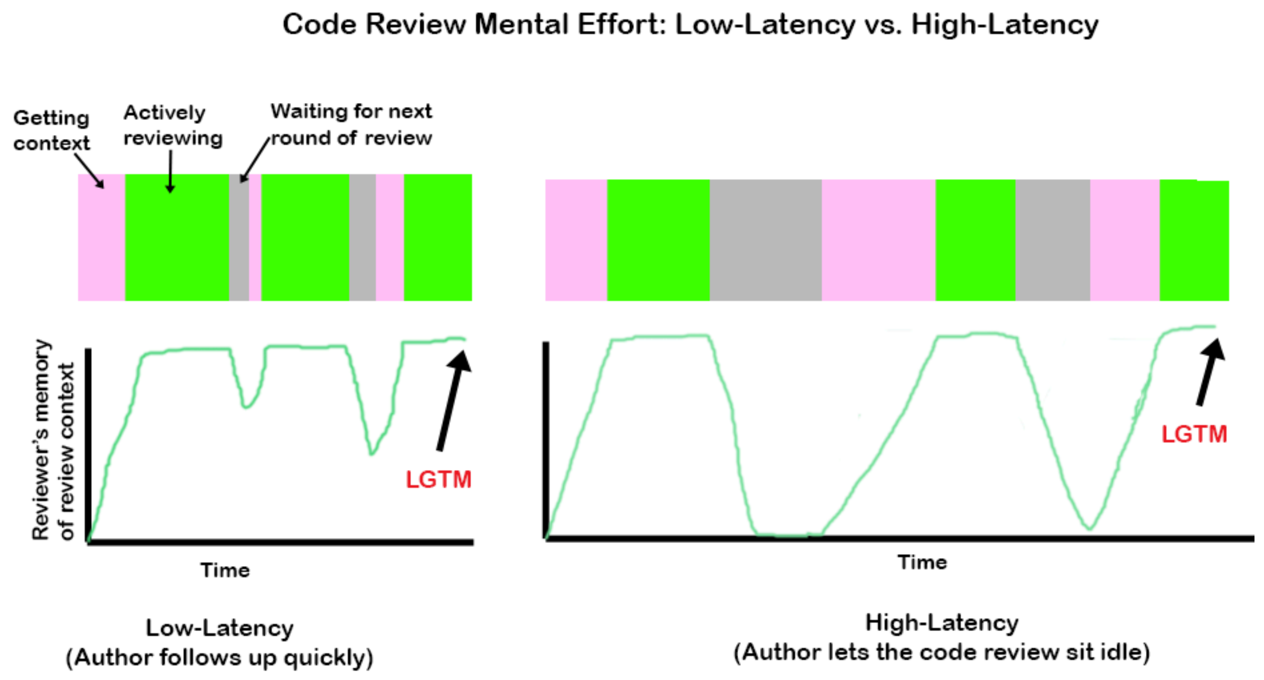
Dog: I’d ask the compiler, but I don’t want to waste its time.

The dog is self-satisfied, as if it’s doing the responsible thing in protecting a precious resource. The cat is frowning, annoyed at the interruption.

I’m not sure how to frame this. I want the focus to be on both their faces, but I’m not sure how to get them both in focus if the cat is sitting, so they’re kind of at different heights. Maybe the dog has sat down on the cat’s desk?

The important thing is just bringing attention to their faces. If we can work in more jokes of the dog obliviously intruding on the cat’s space, that’s a bonus (e.g., sitting on the cat’s computer mouse, knocking over other stuff on the cat’s desk).

Image #5: Total review time


Code Review Mental Effort: Low-Latency vs. High-Latency
Getting context
Actively reviewing
Waiting for next round of review
Reviewer's memory of review context
Time
LGTM
Low-Latency
(Author follows up quickly)
Time
LGTM
High-Latency
(Author lets the code review sit idle)

This one's a little abstract, and I'm open to suggestions on how to improve it.

The idea is that during a code review, there are rounds of feedback where the reviewer gives notes, then waits for the author to send edits, then reviews again, and so on under the reviewer approves (LGTM).

When the author leaves long gaps between rounds of review, the reviewer forgets more about the context, so they have to spend more time remembering before they start the next round of review. The point I'm trying to get across is that leaving gaps degrades efficiency.

The green blocks (actively reviewing) should be identical on both the low-latency side (left) and the high-latency side (right). In other words, the time the reviewer spends reviewing doesn't change - the only change is the time to get context.

The colors are all placeholder, and I'm not married to any of them. But I do want to suggest positive and negative in the blocks:

- Waiting for next round of review: Neutral color, like gray.
- Actively reviewing: This is a positive action, so positive color.
- Getting context: This isn't necessarily negative, but it's friction, so some color to indicate undesirable.

I made the color blocks separate from the line graph, but it might work better superimposed on the line graph.