Mindful Communication In Code Reviews

By Amy Ciavolino, presenter notes are at the bottom.

What is mindful communication?

Mindful communication means to listen and speak with compassion, kindness and awareness.

This is a simple definition I like. It's a high level and easier to keep in mind day to day. This includes yourself and others!

General Guidelines

- Put yourself in the other person's shoes.
- Think about how it would feel to hear what you to say.
- Listen to hear what's actually being said.
- Let go of what you think the outcome should be.

Why did they do what they did? Is the tone and wording something you would feel good hearing? People do a lot of filtered listening, try to really hear what someone is saying. Remember you might not have the best answer. ***You might have to fake it till you make it on this one.***

Mindful Communication *In Code Reviews*

But this is all easier said than done and this talk is about how to apply these to code reviews, so let's talk about code reviews! I'm going to go through some tips for reviewers, and some for reviewees.

Tips for everyone

Take care of yourself

Giving kind and considered feedback takes time and energy!

If you're hangry, tired, in a hurry, have a lot of meetings, about to leave for vacation, etc. *don't review code or send out a review*.

Fix those things first.

Take care of yourself or you can't take care of others.

Tips for reviewer

Don't jump to conclusions

Assume the author did what they did for a reason.

Instead of telling the reviewee they did something wrong, try

`"What about this other option? It might make xyz better!"`

`"What happens in abc situation?"`

Put yourself in their shoes, when you make a code change you likely consider a few options, and go with one that fits the needs. They likely did the same thing. This hits on "Let go of what you think the outcome should be." You might not have the right answer. You might not have all the information you need to understand why the author took the approach they did.

Ask open questions

"This part is a little confusing, could you walk me through what it's doing?"

"I'm having a hard time following this section, how does it work?"

"What other options did you consider for this part?"

Listen! Don't assume. If you think something is 100% off the wall, consider that you respect your co-workers, and ask them to explain their thinking. The answer might surprise you, or you might learn something, or get a glimpse into someone else's thought process.

Nit: leave this out

"Nit: ..."

"This is a nitpick ..."

"Tiny nit: ..."

"This is slightly nit-picky, but..."

If it's not worth saying that much about, then maybe it's not worth saying at all.

Why minimize your own feedback? Either its important, in which case just say it, or it's not so leave it out. We have linters. How do you feel when someone nit-picks your code? Not great, right?

Call out if things could be fixed later

"This could cause x problem later, but I think it's fine to fix in the next iteration"

"Can we make a ticket to come back to this section? It might cause a bug down the road!"

So you've left out the nits, but there are still things you think could be improved but aren't strictly necessary or are stylistics difference, let the reviewee know you don't mind if they're updated in a later or follow up PR. You might not be aware of a deadline for the team or that there are a lot of changes still coming.

Be Biased Towards Approving

Our work is ittertive so don't be a gate keeper. Try to aprove it even if you disagree with the style or approach.

There's a weird power dynamic in code reviews, the reviewee *must* get your or someone else's approval. Don't abuse this power, unless you're worried a change will cause major problems, approve it. "request changes"/blocking on Github: my opinion on it is that you should basically never do that. If you are that worried about something go talk to the person! Lastly, this might be different if this is a vary large change or includes an architectural decision

Give example code

Particularly when reviewing code for someone more junior, give a specific example in the review comment.

```
"This section might be more clear if you switch the order
like this:
if (true) {
    return "this is an example";
}"
```

This is kindness. Saves them time, you already thought through it so why make them figure it out on their own.

Link to documentation

Shows them where to find the answer next time!

End with "what do you think?"

Honor the person whose code you're reviewing. Ultimately they will be making and owning the change. If you have a more fundamental or big piece of feedback, end the comment with `"What do you think?"`.

Goes back to letting go of they out come, maybe you don't have all the context, maybe the author already thought about the approach you're suggesting and there's issues with it you missed. Hopefully it doesn't get here, which leads me into tips for the reviewee.

Tips for reviewee

Small iterative changes

It is so much easier to review small changes. The smaller the better.

This is kindness and awareness, how you'd like to be treated.

Link to documentation

Makes it easy for the reviewer to dig in if they need to and provides important context for a small changes.

Have you ever recived a one line change and had no idea what it meant? A link to the docs would be so helpful in these cases.

Assume the reviewer has the best intent

Even if a reviewer followed all the tips so far, they'll make mistakes or get careless. Try to keep in mind we're on the same team and they likely had good intentions behind whatever feedback was given.

It's easy to get defensive, but they aren't trying to be mean. Unless they are, in which case that's harassment and you should speak to your manager/HR. As the author you have to let go of the out come also.

Think about the readability of your diff

Your diff itself is a kind of communication. Make sure it's well written!

Think about the readability of your diff

- Don't refactor in addition to new changes.
- Check that your PR doesn't have unnecessary changes.
- Make sure it doesn't include unrelated files.
- Each PR should be one logical unit of work.

Get feedback earlier and often

Having a quick chat about some implementation options with someone who'll review your code is a great way to save stress when the review comes around.

Provide or ask for history about how the decision was made. If you weren't part of conversations about the approach, maybe get some history before suggesting a different one.

This sounds like hard work!

It is hard work

I said you had to take care of yourself!

…but it also gets easier

You can build up muscle memory of listening and giving kind feedback and then it'll become natural.

It might feel uncomfortable in the process. Learning new things is uncomfortable. Again, you might have to fake it till you make it. You might have to stop and take a deep breath.

Thanks! 👋

I'm on twitter: @imightbeamy
You can find the slides at http://amy.tech