Pair Programming Anti-Patterns

Source: https://tuple.app/pair-programming-guide/antipatterns

It's possible to pair well simply by avoiding pairing poorly.

Stay away from these common mistakes and you'll up your chances of success.

For navigators

Leaping on errors too quickly

Give your driver a chance to notice their own syntax errors and typos.

Constantly pointing out small errors hurts flow. Yours *and* theirs. It may also make your pair self-conscious.

Remember: your job is to consider the bigger picture, not to point out misspelled words as soon as you spot them.

Giving low-level instructions

If you have a suggestion for the driver, communicate it at the highest level of abstraction they'll understand.

If you find yourself dictating code (or worse, individual keystrokes), stop and see if you can communicate your idea at a higher level.

If that fails, ask to drive for a bit to get your idea sketched out.

Not bringing a keyboard

Bring your own keyboard to every pairing session. Plug it in before you start.

This makes swapping roles easier and allows you to show rather than tell when words fail.

Having your own mouse is nice too, but not as essential. (It's easy to ask someone to click on something, harder to get them to type many characters.)

For drivers

Driving too fast

If you're highly proficient with your editor, it's easy to move fast enough to lose even experienced navigators.

Unless you're sure your pair is keeping up, don't manipulate code quite as fast as you're able.

It helps if you dictate what you're doing.

Allowing a checked-out navigator

It's easy to lose your navigator's attention by moving too fast, or doing things they don't quite understand.

If you get the sense that your pair's attention is drifting, stop and sync up.

A bad question: "You understand this, right?"

A good question: "Which part of this is hardest to follow?"

Pairing should involve constant two-way communication. If you or your navigator has gone quiet, stop and check in.

Unequal screen access

Sit so that the monitor is **between** the two of you. Make sure both of you can see it equally well (consider bumping up font sizes).

If one person is tucked off to the side, it will create a subconscious unequal hierarchy.

A pair is a unit. Neither of you is more important.

Not taking breaks

Pairing is draining. Even more so than normal programming.

A nice way to ensure you take adequate breaks is to employ the Pomodoro Technique. Consider agreeing on preferred work and break lengths with your pair before you start.

Listening without hearing

It's hard to listen and type at the same time.

If your navigator is making a suggestion, consider taking your hands off the keyboard. Even better: turn and make eye contact.

For both

Allowing unproductive distractions

Before you start pairing, disable all notifications (on your computer *and* phone).

A pairing session should be interrupted by exactly zero Slack notifications or text messages. If one slips through, apologise and disable future ones.

Don't leave your email open on another monitor.

(You should do the above even when you're not pairing. The quickest way to improve programming productivity is to reduce interruptions.)

Not swapping roles

Driving and navigating are draining for different reasons.

Swapping roles lets you rest the tired parts of your brain and activate the idle ones.

Swapping drivers is a great way to energise a pairing session that's losing steam. Consider setting a timer to indicate every time it's time to switch.

Forgetting it's a skill

Pair programming is a skill which must be learned.

You will not be good at it at first, but consistent practice will yield improvements.

Don't give up after a difficult first experience. Don't assume experienced developers are automatically good pairing partners. Don't expect to be good without practice.

Consider reflecting with your pair or asking for feedback after each session. What could have been better?