

# How to pair with a junior developer

Source: <https://tuple.app/pair-programming-guide/how-to-pair-with-a-junior-developer>

## Prioritise learning over productivity

Yes, you could probably complete this task faster on your own.

If you've chosen to pair on it, it's because you are wisely prioritising your junior's future productivity. Over the long term, levelling up a team member is worth more than completing today's task.

You're likely to learn from this experience as well. If you take your pair's questions seriously, you might find yourself discovering holes in your knowledge, or uncovering assumptions you didn't know you had.

If you're under too much pressure to prioritise this learning, don't pair.

## Understand they're probably nervous

Most people find it intimidating to pair with someone with more experience.

This can happen even if you're extremely friendly, or have known each other a long time.

Assume your pair is at least a bit nervous.

## Realise fear disrupts thinking

A nervous person is primed for physical action, not careful logic.

When it takes your pair a little longer to spot the typo, decipher the error, or respond to your question, cut them some extra slack.

## Don't make it worse

It's extremely easy to make things worse with a careless word, sigh, or body language.

If you get frustrated or bored, it's your duty to hide this fact as best you can. If you can't hide it, suggest a break.

Try not to drum your fingers on the desk.

## Share the driving

Shoot for an even division of driving responsibilities.

When you drive, don't move too fast. Your pair won't want to tell you that they're not keeping up. Consider using the mouse even if you know a shortcut so they can follow better.

You should constantly be narrating your actions and sharing your thought processes. Teach them how you think about problems.

## Don't over-optimize

Don't go nuts telling them how to accomplish tasks faster (via editor shortcuts, shell aliases, and the like).

Spend most of the session observing them patiently and share only the tip or two most likely to provide a payoff.

## **Avoid definitive language**

Don't say someone should "always" or "never" do something. Share the trade-offs of each approach.

Avoid justifying a decision by saying it's a best practice. Talk about the elements make it a good fit for your situation.

## **Ask questions**

- "How about we extract a method?"
- "Can we find a better way to do this?"
- "Should we take a break?"

## **Admit, and lead by example**

Admit when you don't know. Show them how to look it up.

Admit when you're stuck. Show them how you get unblocked.

Admit when you're tired. Show them the worthiness of breaks.