Your First Pairing Session

Source: https://tuple.app/pair-programming-guide/your-first-pairing-session

First: **relax**. You got this.

Pairing is nothing more than two people helping each other on a task. You've probably "accidentally" paired with a colleague when facing a tricky bug or new area of the codebase.

That said, it's common to be a bit nervous in the beginning. Thus, the biggest enemy of your first pairing session is usually nerves.

Here are some tips to keep things low-stress.

Try it with someone nice

Ideally, your first session will take place with an experienced pair. But it's even more important that they be kind.

Optimise for nice.

Do it when you're fresh

Shoot for first thing in the morning, when caffeine and motivation levels are high.

Take extra care to avoid immediately after lunch or very late in the day.

Get the ergonomics right

- Sit directly next to each other, with the monitor equidistant between you.
- Plug in one keyboard per person.
- Bump up the text size a little.
- Use an editor/IDE that you both know reasonably well.

Pick an easy task

Once you've got some experience under your belt, you'll find that pairing truly shines when the problems get harder.

But for this first session, try to pair on something that doesn't feel intimidating. Something you're already pretty sure you know how to accomplish is good.

Maybe don't call it pairing

The word "pairing" carries some baggage with it.

If the idea intimidates you or your partner, consider not even using the p-word.

Try "can I get a second set of eyes on this?"

Try mob programming ("mobbing")

Some folks find mob programming less intimidating than a two-person pairing session.

Here's what you do:

1. Get a group of three or more devs together.
2. Put some code up on a TV or very large monitor.
3. One person drives while the group navigates.
4. Switch drivers very frequently. Every ten minutes is good.
5. Marvel at the quality of code you produced, how much you learned from each other, and how much fun it was.

Timebox it

It's a good idea to start with shorter sessions. 45-60 minutes is about right.

Pairing requires above-average focus and communication, which can wear you out surprisingly quickly.

With time, your stamina will grow, but keep your sessions short at first.

Wait to draw conclusions

After your first session, you'll begin to form an opinion about pair programming.

Try to avoid this.

Pairing partners vary widely in quality and compatibility. Try to pair with five different people before you decide if pairing is right for you.

Use a template

Want some further guidance on your first session?

Check out our pairing session template for ideas.

A Pairing Session Template

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

1. ☐ Agree on the high-level goal out loud.
2. ☐ Break the work into a handful of tasks and prioritise them.
3. ☐ Decide your driver/navigator swapping strategy.
4. ☐ Configure git to share credit.
5. ☐ Eliminate distractions.
6. ☐ Work.
7. ☐ Analyse the session with a mini retro.

Agree on the high-level goal out loud

State out loud what you hope to accomplish at a high level.

You wouldn't think it'd be possible for two people to start pairing without agreement about where they're headed, but it's surprisingly easy.

Break the work into a handful of tasks (and prioritise them)

It's worth trying to break your high-level goal into a handful of smaller steps.

This has a number of benefits:

- It makes the goal less intimidating.
- You'll spot dead ends and pitfalls more easily.
- You can sort your task list by priority.
- You're more likely to notice that accomplishing task C would make B easier, and reorder appropriately.
- You can decide on a task based on your current energy levels.
- It gives you a clear place to put new tasks you think of while working.

Some folks like to write each task on its own index card. The stack of them lives in front of the navigator. Each card can be a nice home for notes or ideas to bring up when there is a break in the action.

Decide what will trigger a driver/navigator swap

Unless you already know what works best for you, I strongly recommend the Pomodoro Technique:

1. Code for 25 minutes.
2. Take a 5 minute break.
3. Switch drivers.

Other pair programming styles exist if you wish to try them.

Configure git to share credit

If two of you work on some code, both your names should appear on the commit.

Here's a handy guide to configuring git appropriately.

Bonus: GitHub understands this natively and will give you both credit for the commit.

A few tools exist to make this even easier:

- git pair
- git duet
- git-together

Eliminate distractions

Show respect for your pair and the work you're about to do.

- Don't bring your phone. Silence it if you do.
- Disable notifications on the machine you're using to pair.
- Close email/Slack/Twitter/IRC. Never keep something distracting on a second monitor.

Work

Do the work!

Don't forget:

- *When navigating*: ask questions rather than making demands.
- *When driving*: dictate what you're doing and why.
- Err on the side of over-communication.
- Take lots of breaks.
- Swap roles frequently.
- Do the simplest thing that could possibly work (for now).
- Avoid these pairing anti-patterns.

Perform a mini retro

Spend a few minutes after your session reflecting on the experience.

First, discuss what went well.

Then, consider what would make the next session 1% better.

Possible areas for improvement:

- **Focus**: did distractions sneak in?
- **Communication**: were there long stretches of no talking?
- **Pacing**: did the session feel like a grind at any point?
- **Division of responsibility**: did you split the work up well?
- **Code quality**: was your end-product high-quality?

Pair Programming Styles

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

Driver/Navigator

Probably the most common type of pairing. This is a good place to start if you're new.

The driver types the code and stays focused on the current task.

The navigator thinks ahead, ponders edge cases, spots bugs, suggests refactorings, asks good questions, stays zoomed out.

Don't forget to switch roles regularly.

To see this style in action, check out Pairing On Elixir, Phoenix, and Elm.

Ping Pong

This is a great option if you and your pair practice TDD.

You'll definitely want two keyboards for this one (or a remote pairing app like Tuple).

Here's the flow, looped until you're done:

1. Person 1 writes a failing test.
2. Person 2 makes it pass.
3. Person 2 writes a failing test.
4. Person 1 makes it pass.

This is a great option because the work is naturally divided, and there's no need to practice the discipline of swapping drivers often.

Three recommendations:

1. Sometimes it will take much longer to write the test than the code that makes it passes, or vice-versa. Don't worry about it. Time spent typing will even out over time.
2. Try to write a *minimal* failing test, and the *smallest* amount of code that will make it pass. This is easy to forget in the moment.
3. Don't forget to spend some time refactoring when things are green.

Want to see this style in action? Check out TDD Ping-Pong Pairing.

Strong Style

This seems like madness, but we're including it here for completeness:

> *The golden rule for this style of pairing is: "For an idea to go from your head into the computer it MUST go through someone else's hands"*

This article has more details.

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?

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, its 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-optimise

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.