




# Pairing: when, why, how

---

Anna Headley  
she/her/hers  
Princeton University Library



# Reasons to pair

---

- Knowledge transfer
  - "We both know the codebase, but have strengths in different areas. Having both of us was helpful."
- Greater insulation from distractions / fewer rabbit holes
  - "It was easier to police the part of me that says `that would work but is there anything that might be better`"
- Knowledge distribution / avoiding silos
  - "Pairing on new functionality ensures more than one person is an expert in it later."

# Benefits of pairing as practice

---

- Stronger relationships within our team
  - "I feel happier working this way" "It's less lonely"
- Higher quality code
  - "We made faster decisions. We made better decisions." "I feel more confident about the decisions we made."
  - "Y'all's code from when you were pairing was merged without any change requests."
  - "Yes, we should write a test for that" and other good habits reinforced by positive peer pressure.
  - Studies find lower error rates in code produced by pairs.

# Start pairing

---

1. Select a task!
2. Set an intention. A reason for the pairing session guides the way you'll communicate.
  - Onboarding
  - Design / code quality
  - Productivity mode
  - Code review
3. Set up your environment/s

# Set up environments

---

- Turn on absolute line numbers in your editor. This aids communication.
- Turn off relative line numbers. They make it hard to reference code.
- Use github's multi-author syntax on your commits
  - ``Co-authored-by: Anna Headley <anna.headley@gmail.com>``
  - Add these to your `.gitmessage` configuration for easy access

# Mechanics of pairing

---

Decide who will drive and who will navigate.

- Driver: Your job is to type.
- Navigator: You keep a slightly higher-level mindset. Keep track of the next few things the driver will be doing to help with transitions. Look out for bugs and edge cases. Ask questions.

Use any screen-sharing program to share the driver's screen.

Switch roles periodically.

# Talk a lot

---

When pairing you are working together, not next to one another. Your pair should know what you are thinking at all times, because you're thinking aloud. Your pair should know what you are doing, because you are looking at something together and narrating your navigation (e.g. "let's go back to the test") or because you have told them something you'll do in parallel (e.g. "I'm going to check that syntax real quick").

# Assess as you go

---

- Check yourself. Are you behaving generously? Are you being patient? Are there power dynamics that you can ease?
- Speak up. Tell your pair what you need to make it work. Slower pace? Answers to questions?



# Potential vectors of power dynamics

---

- junior - senior
- "wrong" background - "right" background
- learning developer - teaching developer
- feminine - masculine
- people of color - white folks
- women & other genders - men
- national origin
- native language
- external signs of religion
- editor preference
- programming language
- location in stack

source: <https://twitter.com/sarahmei/status/990968833547497472>

# Wrap up

---

- Do a small retro for the pair session, especially if this is your first time pairing with one another, or it's been a while since you've done a retro together.
- If there's unfinished work, decide whether to pair again or who will finish it. Try to avoid having one person work more on the issue if you'll continue working together.

# Introducing pairing to your team

---

Some people on the team want to try more pairing, but everyone is used to working solo most of the time.

Informal introduction requires people to ask one another if they are available / want to pair. It can be hard to time these requests, and they can feel awkward to make.

Figure out within your team how to introduce formalities to ensure success. Evaluate your practice periodically as a team.

# A formal pairing mechanism

---

Example: During our Friday wrap-up meeting, we will ask if anyone wants to pair next week, and match people up. Those people will pair 3 afternoons, or similar. Other details can be arranged at this time, such as what kinds of issues to work on.

- Ensure no one pairs all day every day.
- Ensure people get some consistency, and pair with a variety of colleagues.
- Ensure people who don't feel their personality / work style is compatible with pairing don't have to do it.

# Further Resources / Notes

---

- <https://tuple.app/pair-programming-guide/the-case-for-pair-programming> - Covers much of this content with some great tips and research summaries.
- <https://www.thoughtworks.com/insights/blog/pairing-are-you-doing-it-wrong> - Presents some variations on pairing, e.g. "mobbing" and "supported soloing".