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.