Pair Programming and Remote Pairing

Total estimated time: 1-3 hours

Top level table of contents

Table of Contents

Intro

Students often haven't heard of pair programming when they first start learning development. It certainly seems to counter the image of the rainmaker developer who hacks away on his own into the wee hours of the morning. For a long time, pairing was primarily the domain of a dedicated but minority set of programmers who followed the Extreme Programming (XP) and the debate continues about whether and how to use pairing for production code. More recently, however, the use of pairing as purely a learning tool has exploded and it's much harder to argue against the benefits and evidence of its effectiveness for that purpose.

Pair programming is when two programmers work on the exact same code base. Often, in the real world, it is done by simply passing the keyboard and mouse back and forth when it's time to switch. Typically, one person (the "driver") writes the code while the other person (the "navigator") watches, thinks several steps ahead, and catches typos and bugs.

In production settings, this results in better code and helps developers stay focused. Despite tasking two developers to produce one code base, the gains in quality and speed make up for the loss of two separate builders. But the two developers will also be constantly learning from each other, constantly thinking through problems together, and constantly staying engaged with the task at hand. That means two developers pairing together can potentially learn many times faster and more effectively than they would on their own.

This curriculum is designed specifically to benefit from and encourage the use of pair programming as a learning tool among peers. The projects won't be easy and a single sneaky bug can frustrate you for hours on your own. But with a pair, you're often able to get past the issue in a fraction of the time. Any time when you're not quite 100% confident of your knowledge but want to dive into a

project anyway (which we encourage!), having a pair sitting either right next to you or 1,000 miles away using a screenshare can be invaluable.

You'll end up working with a lot of different types of people over the course of your learning. Sometimes it will go really well and other times you'll have to find patience. But, overall, your learning experience will be greatly enriched by the chance to both learn from and teach to your peers along the way.

Read (briefly) about pairing on the Extreme Programming website.

Why Pair?

The major benefits of pairing to learn are:

- Learn by teaching, the most effective method
- Get through bugs faster
- Ask questions of a real human
- Get realtime feedback on the code you're writing
- It forces you to focus and remove distractions

Things that are difficult about pairing:

- There's an element of personality fit between partners
- You need to get used to thinking out loud
- You need to get used to teaching instead of just grabbing the keyboard and doing things

Some Articles:

- Why Pair Programming Works
- Pair Programming Considered Extremely Beneficial
- Pair Programming: When and Why

Additional Resources

An older writeup on pairing from Dr. Laurie Williams of NC State University has a rich bibliography of research on pairing.

How do we Pair?

Tools

For best pairing, we recommend using two very easy tools: Skype and Screenhero. Skype lets you chat with other users for free. Screenhero is like Skype for screen sharing and lets you interact remotely with the other person's computer. Both tools are just one-click-to-call, and using them together will allow you to talk while you're working together on the same code on one of your computers... remotely!

Combining being able to talk with someone (over Skype) and being able to see and interact with the same computer (over Screenhero) is what makes pairing work.

You could do it using just chat but it's not close to the same experience. It's probably not necessary (and a waste of bandwidth) to have a video feed going at the same time either.

Setting up Skype Download the free client at Skype.com, set up your user details, and "call" your partner.

Setting up ScreenHero Setting up ScreenHero: Download the free client at Screenhero.com, set up your account, and "call" your partner to start sharing your screen and allowing your partner to interact with it. You'll see two mice and you'll both be able to use your keyboards to interact with the shared computer. Cool!

Alternative Technologies (experimental) Sometimes a program just won't work properly so you need to go to Plan B. In our case, we usually recommend Google Hangouts because it has both audio and screensharing/remoting tools baked into one. The bandwidth usage isn't quite as efficient as the Skype/Screenhero combination but should work fine for your purpose. You'll need to sign up for Google Plus and add the extension to the Chrome browser to get it to work.

This tutorial describes setting up a hangout session, though they don't get into the actual remote pairing side of things. **Basically, just click this link to open a new hangout once you've got the plugin installed.**

We'll be using the feature called Remote Desktop, which lets your partner use your computer like maybe you've experienced before with tech support.

Another alternative, as described in this blog post, is Apple's native Screen Sharing app (combined with Skype).

Resources Here's a Google Hangout tutorial (video) and another on using Hangouts to screen share. We will be using Remote Desktop Sharing so check out this how-to video to familiarize yourself with it.

Here's an older video on How To Use Google Hangouts.

Do you have a favorite screensharing or remote pairing technology? Email us or fork this repo, add your changes, then submit a pull request.

Once you are able to talk to each other and you can both work on one of your screens, it's time to get coding!

Techniques

There are a few different techniques for how to manage the actual process of pairing. If you stick with Time Swapping and use an actual online timer, you can't go wrong. It's the best defense against keyboard hogs.

- **Time swapping:** Spend 25 minutes as the driver, then swap roles. Try using a free online timer like this one set to 25 minutes.
- **Ping Pong:** If you're using a test-driven approach, one person writes the failing test then the other person makes it pass. Swap roles for the next test or batch of tests.

Some tips:

- Think out loud as much as you can – give each other a chance to hear your thoughts.
- No distractions! Emailing, texting, etc. should all be turned off while pairing.
- As navigator, wait 10 seconds before pointing out a typo... give the driver a chance to correct on their own

After You've Finished

When you're done with your session, you've got code sitting on someone's computer that you'd both like to have. If you're using git to save your work and push to a github repo, just make sure the repo is public and both of you will have access to it or be able to fork it as necessary (a good incentive to start using git...).

If you aren't using git yet, just have the student with the code on their hard drive email out the files or a .zip archive of them to make sure both people have it.

Additional Resources

- WikiHow has a solid entry explaining how to pair program effectively.
- Check out this Slideshare about pairing
- To see real pairing sessions in action, check out these recordings

A Sample Remote Pairing Session

If all of that seemed a bit confusing, here's a step-by-step workflow for remote pairing.

1. Find and call your partner on Skype
2. Find and "call" your partner on Screenhero and share your screen
3. Get your pomodoro timer fired up
4. Come up with your battle plan for the project
5. [If using git] Create the public Github repo for the project
6. Code! Switch! Code! Switch! Repeat n times!
7. Push your code to the Github repository or email it to the partner whose computer wasn't used
8. Schedule another session!

TODO: video of a sample pairing session

Finish

You should now have the simple tools installed to allow for remote pair programming and a good idea of how to use them. Get excited! That's a small but important step forward... now you've got the power to pair with anyone on any project. All that's left is to start learning and get started on some projects!

Additional Resources

Check out this video talk about pairing professionally at Pivotal Labs from Joe Moore.

Have any other pairing instructions, tips, tricks, or helpful sections? Is there anything about pairing, whether remotely or not, that you've found to be difficult or surprising that we should address here? Email us or fork this repo, add your changes, then submit a pull request.