Pair Programming

Why & How

What is it?

- ➔ Two people
- ➔ Collaborating on one code source
- ➔ To collectively produce one program

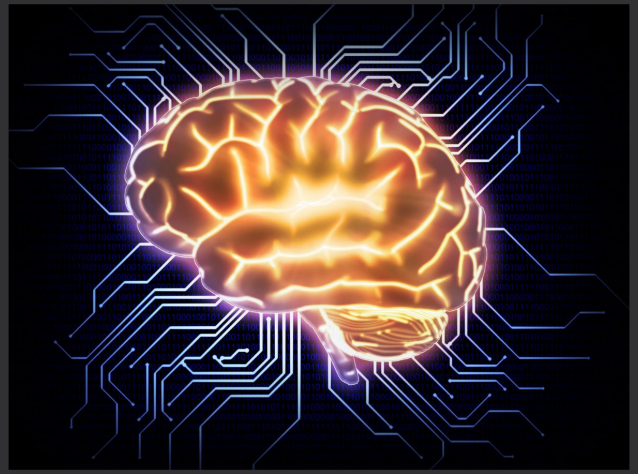
(that's it!)

Questions

Advocates for pair programming might say....

Why?

- ➔ Increased specialization requires more collaboration
  - Different people have different knowledge and strengths
  - Two people problem solving leads to better solutions
- ➔ Distributing cognitive load
  - Easier to perform individual functions
  - Easier to catch mistakes and produce high-quality code
- ➔ Not tempted to take short-cuts
- ➔ Not distracted by other tasks



Why?

- ➔ Coding is communication with your machine
- ➔ Development is communication with your colleagues
- ➔ The most important thing your code does is communicate.
- ➔ Many developers struggle to articulate their problems and talk to others about their code.
- ➔ Pair programming forces you to practice coding vocabulary and talk through your problems.

How: *The Ingredients*

- Two...
  - living, breathing, awesome humans
  - unique experiences and perspectives
  - levels of coding expertise
- One...
  - GitHub repo
  - code base
  - person typing at a time

How: *The Roles*

- ➔ “Driver”
  - Using keyboard and mouse
  - Managing the editor
- ➔ “Navigator”
  - Watching, advising, thinking
  - Responsible for higher-level input to the code
  - Catching errors, bugs, typos
- ➔ Talk to each other constantly!
- ➔ Both must understand Why and How

How: *Do We Communicate?*

Scenario:

I'm driving and I feel like my navigator is not engaged.

How: *Do We Communicate?*

Don't:

Get angry, give up and do separate work.

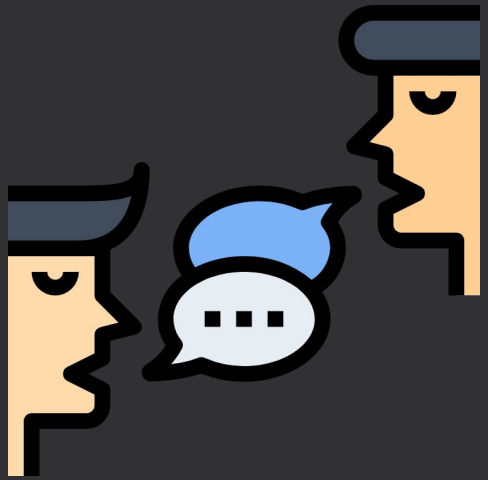
Action:

Suggest taking a break. When you return from your break, take 5 minutes to regroup and refine the plan of action. Together, write down your goal and the plan to accomplish this goal. Hold each other accountable to stick to the plan.

How: *Do We Communicate?*

Scenario:

My partner just made a suggestion and I know their idea won't work. I want to go in another direction.



How: *Do We Communicate?*

Don't:

Say, "No, that won't work."

Action:

Set a time box (ex. 5 minutes) and try it. There is only one way to find out.

Or, if you have a good grasp of the concept, explain your thought process. Communicate the pros and cons to each approach.

How: *Do We Communicate?*

Scenario:

I am very new to coding and my partner seems to know way more than me. I feel like I am just following along and not learning for myself.

How: *Do We Communicate?*

Don't:

Sit quietly and feel defeated.

Action:

Be strong enough to put yourself in the vulnerable position of asking for help. Ask your partner to slow down and talk through the process. Ask for clarification on code you don't understand and take notes of confusing points to review later.

How: *Do We Communicate?*

Scenario:

I have more coding experience than my partner. I feel like they are not contributing and I am left to do all the work.

How: *Do We Communicate?*

Don't:

Take on the challenge alone.

Action:

Remember, there is more to being a developer than being great at coding. Employers want people who are team players and good communicators. Look at this as an opportunity to improve your skills by helping and teaching your teammate.

How : *The Process*

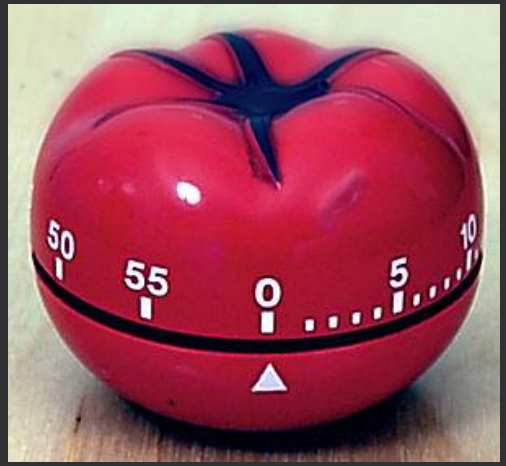
➔ Switch roles every 10-20 minutes

- OR -

➔ Pomodoro method

- Decide on the task to be done
- Set timer to 25 minutes
- Work on the task until the timer rings
- Take short break: 3–5 minutes
- Switch roles

https://timer.town/
https://double-trouble.wielo.co/
https://agility.jahed.dev/
https://tomato-timer.com/



How : *The Process*

- ➔ Collaboration Tools: GitHub!
- ➔ Each pairing team will create a repository on GitHub
- ➔ Each participant can use whatever development tools are most comfortable
- ➔ The driver will share his/her screen and have “hands on the keyboard”
- ➔ The navigator will talk through the coding process
- ➔ When it is time to switch, the driver will push the code to the Github repo
- ➔ The new driver will pull from the GitHub repo and screen share
- ➔ If you are not sharing your screen, you should not be coding