Pair Programming

QA

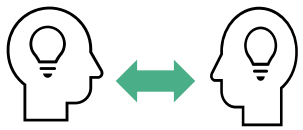
Learn. To Change.

What is Pair Programming?

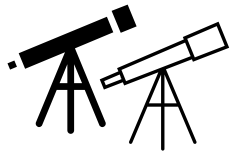
- Two developers work together on one project
- One developer codes while the other reviews
- Co-responsibility for outputs
- Developers can switch roles when needed
- Effective for identifying bugs and design problems, and maintaining coding standards

Benefits of pair programming

- **Increased productivity**
- **Increased confidence**
- **Cross-learning**
  - Reduced training costs and time

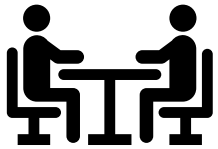


- **Multiple points of view**
  - Faster issue resolution
  - Fewer obstacles



Benefits of pair programming

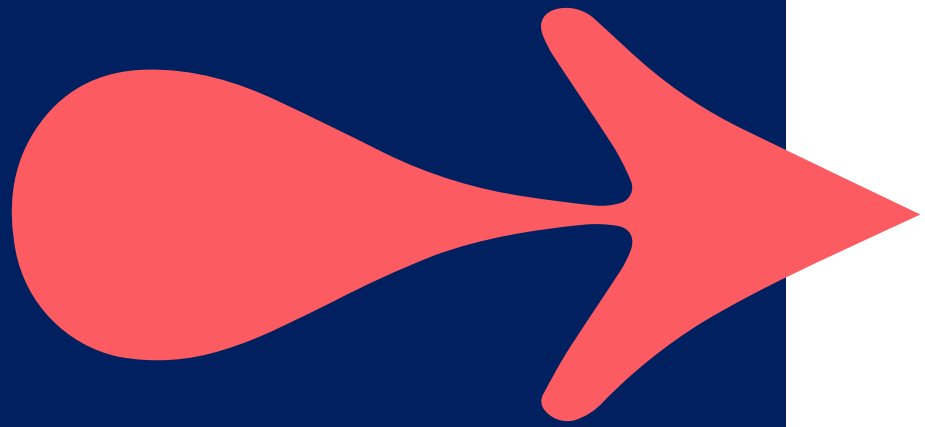
- **Continuous code reviews**
  - Give feedback and reduce bugs more quickly



- **Shared responsibility and trust**
  - Both equally own the code and its quality
  - Feel safe sharing ideas, asking questions, making mistakes.
  - Backups if a developer is on annual leave or falls ill

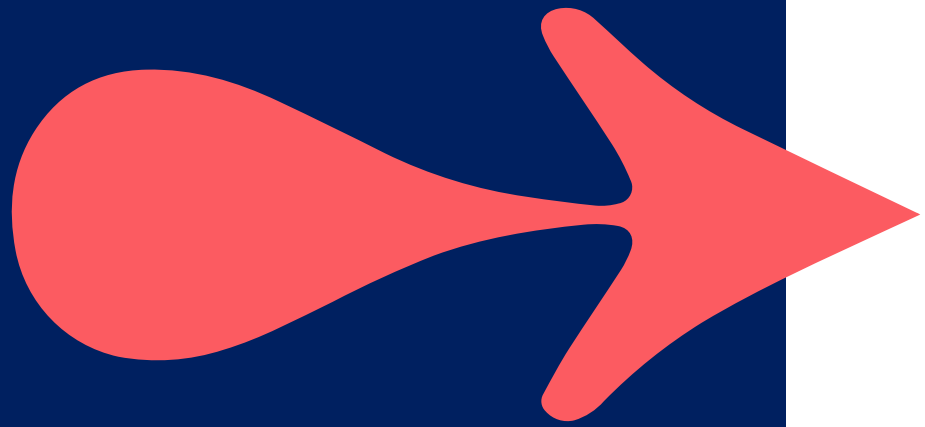
Types of Pairing

- Driver-Navigator
- Backseat navigator
- Tour guide
- Ping-pong
- Cross-functional
- Distributed



Pair programming types

- Expert-Expert
- Expert-Novice
- Novice-Novice
- Developer-Operations engineer



Driver-Navigator (most common style)

It is like when you negotiate a car trip through unfamiliar territory.

Driver

handles typing, save and load files, and other implementation issues.

Navigator looks at other issues like

checking for mistakes.

does the code fit with the designed architecture?

are we duplicating code which exists elsewhere?

are we in a blind alley (code going nowhere)?

Navigator provides a more strategic and high-level guidance

Backseat Navigator (Novice & expert)

Driver
Takes care of typing code and saving files

Navigator dictates tactical instructions like:

- When to create a class/method or a new file.
- What to name a file or unit test or even a variable.

Navigator exerts more control (line-by-line critique)

Tour Guide (Expert and new Novice)

It is like when you take a trip conducted by local tour guide.

The ***driver*** (the expert) writes the code and explains what they're doing,
the ***passenger*** (the novice) can follow and learn.

It might seem like the passenger is just watching, but their role is active

They should take notes and try to understand what's happening.

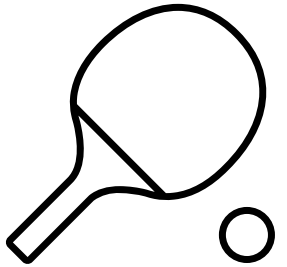
You can also switch roles! Let the novice try coding, even if they make mistakes.
The expert watches, then gives feedback.

Pair programming types…

Ping-Pong: (good for TDD)

The first programmer writes a failing test and the other writes the code to pass it.

- Encourages Test-Driven Development (TDD).
- Keeps both developers engaged.
- Balances contributions and learning.
- Helps catch bugs early.
- It is fun!



Pair programming types…

Cross-functional

- **For example, the developer works with a hardware engineer**
- Allows more time to work alone
- Ideal for developing new systems

Pair programming types…

- **Distributed**
  - **members of the team are in different geographical locations**
  - Developers work together via a collaborative real-time editor, shared desktop or remote pair programming plugin
  - More tiring than traditional pair programming (more hours)
  - Tools used include *Mikogo*, Trellis and *Yuuguu*

Lab

Use one of the Pair Programming methods
Choose just one solution in your group and share code.

Only C# and Java starter projects are provided

Afterwards, we'll discuss...

Which pair programming method(s) you tried

What went well

What didn't go so well

QA

Learn. To Change.