Level Up Your Engineering Career with Mentorship, Pairing, and AI

Hana Harencarova
Software Developer at GitHub
@hharen

What you'll hear today

1. 🤝 Mentorship
2. 🍐 Pair programming
3. 🤖 AI for learning

👋 Hi, I'm Hana

Helvetic Ruby

🍵

🤝Mentorship: Learn Faster by Learning Together

???

Technical topics - start

- Ruby/Rails to help build my first projects
- Onboarding to a new codebase
- Implementing specific feature
- Different parts of Rails/Ruby stack

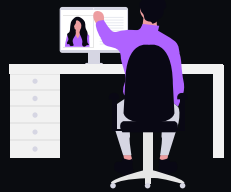
Technical mentoring - advanced

- Special topics like:
  - Elegant and performant code
  - Rails/Ruby source code
  - Observability and incident investigations
  - New language(s)
  - Contributing to OSS

Career mentoring

- Understanding the field
- Understanding the company
- Reflections and feedback
- Career options
- Promotions
- Visibility

List of topics for mentees










Mentor & Mentee

- Setting up sessions for success
  - topics
  - expectations
  - time frame
- Preparing for sessions to maximise value

How to be a good mentor

- Check with your mentee on the topics
- Give contained tasks/readings
- Bring your insights

How to be a good mentee

- Prepare
  - Do readings/tasks
  - Write down questions
- Setup - even if just 5 minutes
  - Dev setup
  - Screens to share

How to find a mentor?

🤝

Why to mentor?

You can do both!!!

Mentorship vs. Sponsorship

Sponsoring

	Mentoring	Sponsoring
Promotion	Talk them through career ladders and expectations	Create visibility for them, advocate for them
More impactful projects	Tell them where to find projects in a roadmap	Bring up their name when planning work
Attending conference on a tight budget	Tell them about posibities to get support	Connect them with organisers or find a way to get them a ticket
Speak at a conference	Go over their proposal, rehearse a talk with them	Suggest them as a speaker

Pairing

Progression

- Learning about the codebase
- Starting new work
- Debugging tricky problems
- Discussing used approach
- A PR walk-through

AI and learning

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

1. Re-reading books and notes
2. Recalling things from memory
3. Mini-testing
4. Underlining
5. Multitasking
6. Solving different problems

Learning

- Overview and bigger picture

Overview and bigger picture
Visual Flow / Call Graph

"Generate a call graph or flow diagram (in text or mermaid) showing how control flows through this code: what gets called, in what order, and under what conditions. Label branches and outcomes."

Overview and bigger picture
Understanding a Complex Function

"Rewrite this function in plain English. Describe the intent, inputs, outputs, main branches, error cases, and what side effects occur. Give me a simplified mental model of what this function is doing."

Overview and bigger picture
All in one

"Analyse the following code and give me a high-level overview of what it does. Then list every major execution path, including conditionals, function calls, and side effects. Present the flow as a clear step-by-step outline or diagram so I can understand the overall behaviour quickly."

Learning

- Overview and bigger picture
- **Getting feedback fast**

Getting feedback fast

“Review the following method. Give concise, high-impact feedback focused on:

1. **Readability** – is the intent clear? how to simplify?
2. **Performance** – any inefficiencies or unnecessary work?
3. **Elegance / Cleanliness** – idiomatic patterns, best practices, cleaner alternatives.
4. **Refactoring opportunities** – how to make it shorter, clearer, or more maintainable.
5. **Edge cases / pitfalls** – anything that might break.
6. A better alternative implementation, if appropriate.

Be direct, specific, and practical. Provide code examples for improvements.
Here is the method:”

Learning

- Overview and bigger picture
- Getting feedback fast
- **Focused attention (learning)**

Learning

- Overview and bigger picture
- Getting feedback fast
- Focused attention (learning)
- **Repetition**

To sum up, what tasks are good for AI to learn?

Happy mentoring, pairing, learning

👀 links

- hharen.com/talks/mpai
  - mentorship, pairing, AI
- Your first Ruby friend (continues in 2026)
  - mentorship program

What's one thing you want to try?

Attributions

- Google calendar icon - Google Inc., Public domain, via Wikimedia Commons
- Undraw image - undraw.co/illustrations
- Rails logo - Jamie Dihiansan weblog.rubyonrails.org/2016/1/19/new-rails-identity/2, CC0, via Wikimedia Commons










Thank you! Let's stay in touch 😊

Hana Harencarova

hharen