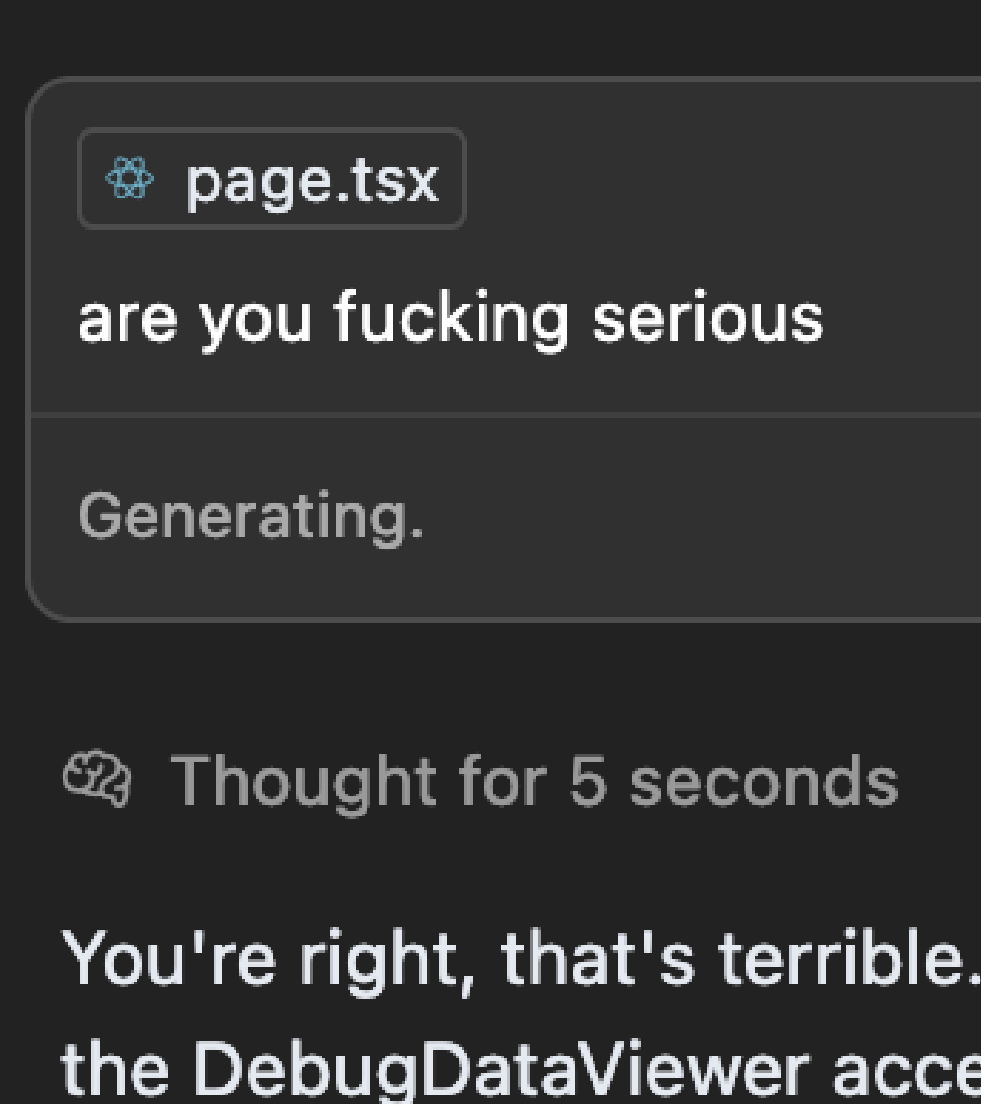


Herding AIs

Alejandro Vidal - Founder @ mindmakers

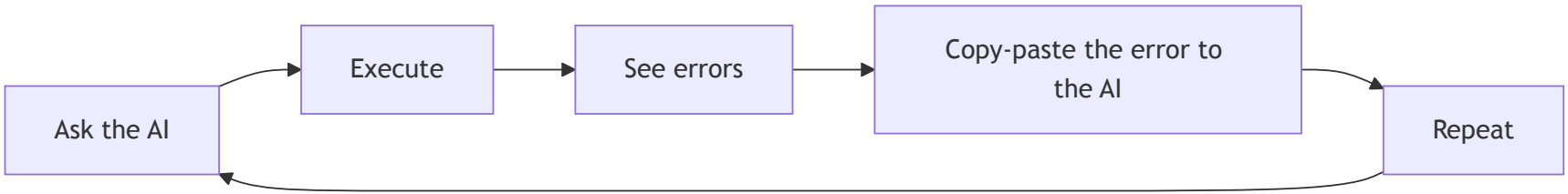
Han @HanchungLee · Jun 19

coding ide in 2030 be like

Summary: no memes

- ⚠ What you'll learn here must be adapted to your workflow ⚠ It's not a magic recipe
- Use Claude Code
- Automating processes takes time and dedication. It's not plug & play.

Typical flow with Cursor (or similar)



This is called a "feedback loop".

We can do better.

AI fails…

- When the prompt is poorly designed (short, under-defined prompts)
- When it has no feedback loop (just like humans)
- Because it lacks long-term planning

** If we solve these problems: what we can do is incredible **

v0: Learning to use AI in the terminal

```
claude --dangerously-skip-permissions -p "Make a joke and write it to JOKE.md"
```

Long-term planning: v1

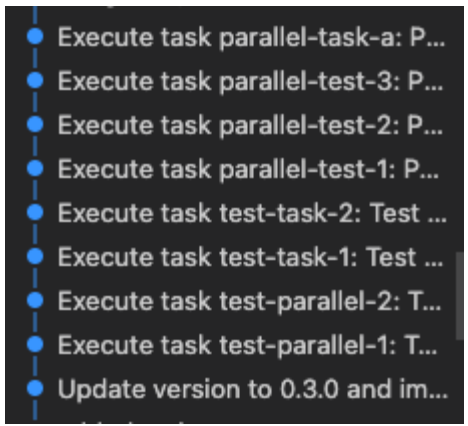
Making task lists…

We can do it with absurdly basic scripting:

```
alias superclaude='claude --dangerously-skip-permissions'
superclaude -p "Make a joke and write it to JOKE.md" && \
superclaude -p "Translate JOKE.md to Spanish, French and German in JOKE_SPANISH.md, JOKE_FRENCH.md and JOKE_GERMAN.md"
```

Long-term planning: v2

Use branches please. Don’t be like me:




```
git checkout -b "feat/add-jokes" && \
superclaude -p "Make a joke and write it to JOKE.md" && \
git commit -a -m "feat: add jokes" &&
superclaude -p "Translate JOKE.md to Spanish, French and German in JOKE_SPANISH.md, JOKE_FRENCH.md and JOKE_GERMAN.md" &
git commit -a -m "feat: add jokes translations"
```

Better prompting + Long-term planning: v3

AI (despite what people think) doesn’t hallucinate much. It’s usually an under-definition problem.

If you ask: "Build me a SaaS to book appointments for a health clinic" there are **millions of ways** to solve it.

```
superclaude
# Chat with it to define what we want and have it write it to a TASKS.md
```

```
while true; do
  superclaude -p "Take the first task, implement it, mark it as done and commit the changes"
done
```

Long-term planning: v4

We can improve it:

```
while grep -r "\[ \]" TASKS.md >/dev/null 2>&1; do
  superclaude -p "Take the first task, implement it, mark it as done and commit the changes. If something goes wrong or
done
```

But… bash scripts and that’s it?

Yes, iterate. Don’t try to implement everything at once. The natural flow is usually:

- cursor
- cursor + claude code
- cursor + claude code + bash scripts
- cursor + claude code + bash scripts + CI/CD
- cursor + claude code + bash scripts + CI/CD + **custom agents**
- cursor + claude code + bash scripts + CI/CD + custom agents + **massive parallelization**

Massive parallelization: v5

And if we define the dependencies...

We can parallelize the tasks:

```
superclaude -p "Create a DESIGN_SYSTEM.md file following the style of the existing components" && \
#  && != &
superclaude -p "Make a dropdown component" & \
superclaude -p "Make a button component" & \
superclaude -p "Make a input component" & \
superclaude -p "Make a checkbox component"
# ...
```

Feedback loop

Option A: In the prompt.

```
claude -p "Implement a checkbox component. After you finish, run `npm run lint` and fix any errors."
```

Option B: Sequential multi-agent.

```
claude -p "Implement a checkbox component and make a commit" && \
claude -p "Read the last commit and run `npm run lint`. Implement a test and fix any errors if any."
```

Option C: Implement some logic:

```
claude -p "Implement a checkbox component and make a commit"
if ! npm run lint; then
  superclaude -p "Fix the linting errors of the last commit"
fi
```

Feedback loop II: Testing

- TDD/BDD is **super important for developing with AI**
- In fact, paradoxically, source code in small teams will start becoming more complex in order to parallelize tasks and make them easier for AI.

Feedback loop II: Testing

Recommendations:

- Set up both unit and end-to-end testing from the start.
- Use the Playwright MCP so the AI can test better:
- Ask the AI to do a "manual" test with the MCP and then **write the test**

Feedback loop III: Multimodal

You can do the same with front-end design if you have multimodal models:

- Implement (with AI)
- Screenshot of the component (in different states) using Storybook
- Iterate on the errors.

Good prompting

- Most example prompts are very under-defined.
  - (# of possible implementations or functional definitions)
- Claude Code with "plan mode" is an excellent example

When things get out of hand…

- You'll sleep less
- You'll start doing side projects
- You'll build your own orchestrator
- You'll build MCPs for your use cases
- **Demo!**

`whoami`

- Founder @ mindmakers
- I help teams adopt augmented development

hello@mindmake.rs

Other interesting things

- opencode: Claude Code but generic (works with other vendors)
- Claude Code GitHub actions (async work)
- OpenAI Codex (on OpenAI for teams): async work on branches.
  - Codex is also the name of the official OpenAI CLI (also supports Google Gemini and others)