Practical AI Productivity Tips For The Too Busy Developer

Slides at scottsauber.com


@scottsauber

Audience

- Looking for ideas how to best leverage AI
- Been too busy shipping value to pay attention
- This’ll be a 100-200 level talk
- More than just “AI in your IDE”
- This an all day workshop, but we have 60 mins

Poll

- Who’s using AI tools? Every day?
- Copilot?
- Claude Code?
- Codex?
- Cursor?
- Windsurf?
- Something else?

Poll

- How many people AI is writing 99%+ of their code?
- Most of their code?

Agenda

- Am I still going to have a job?
- Define all the AI Buzzwords like Context, Instructions, Skills, and more
- Talk about tools like Copilot and Claude Code
- Show some demos
- Practical Tips about where AI fits and where it doesn’t

Goals

- Give you ideas you can start using today
- Where AI tooling is going in the future

Who am I?

- Director of Engineering at Lean TECHniques
- Microsoft MVP
- Dometrain Author
- Redgate Community Ambassador
- Co-organizer of Iowa .NET User Group

Am I going to be out of a job?

History Lesson (2000s – Present)

- What set teams apart was their commitment to modern engineering practices and modern work practices – proven by DORA
- Fast feedback
- Working in small batches
- Learning Culture
- Automated Tests
- CI/CD
- Cloud
- Database Automation
- Monitoring and Observability
- And more…

Which ones are still relevant in an AI-dominated world?

History Lesson (2000s – Present)

- What set teams apart was their commitment to modern engineering practices and modern work practices
- Fast feedback ✅
- Working in small batches ✅
- Learning Culture ✅
- Automated Tests ✅
- CI/CD ✅
- Cloud ✅
- Database Automation ✅
- Monitoring and Observability ✅
- And more…

The Fundamentals Remain The Same

Jevon's Paradox

- Economic Principle of increasing the the efficiency of a resource actually leads to more use, not less
- More fuel efficient engines, means cheaper travel, which encourages people to drive, which means more fuel consumption
- Faster Internet speeds has led to more internet usage not less (ie streaming)
- If history repeats itself - becoming more efficient with creating software will actually result in MORE software, not less
- Building faster changes the Build vs Buy equation
- We all still have jobs!

“SaaS is Dead.”

- Satya Nadella, Microsoft CEO

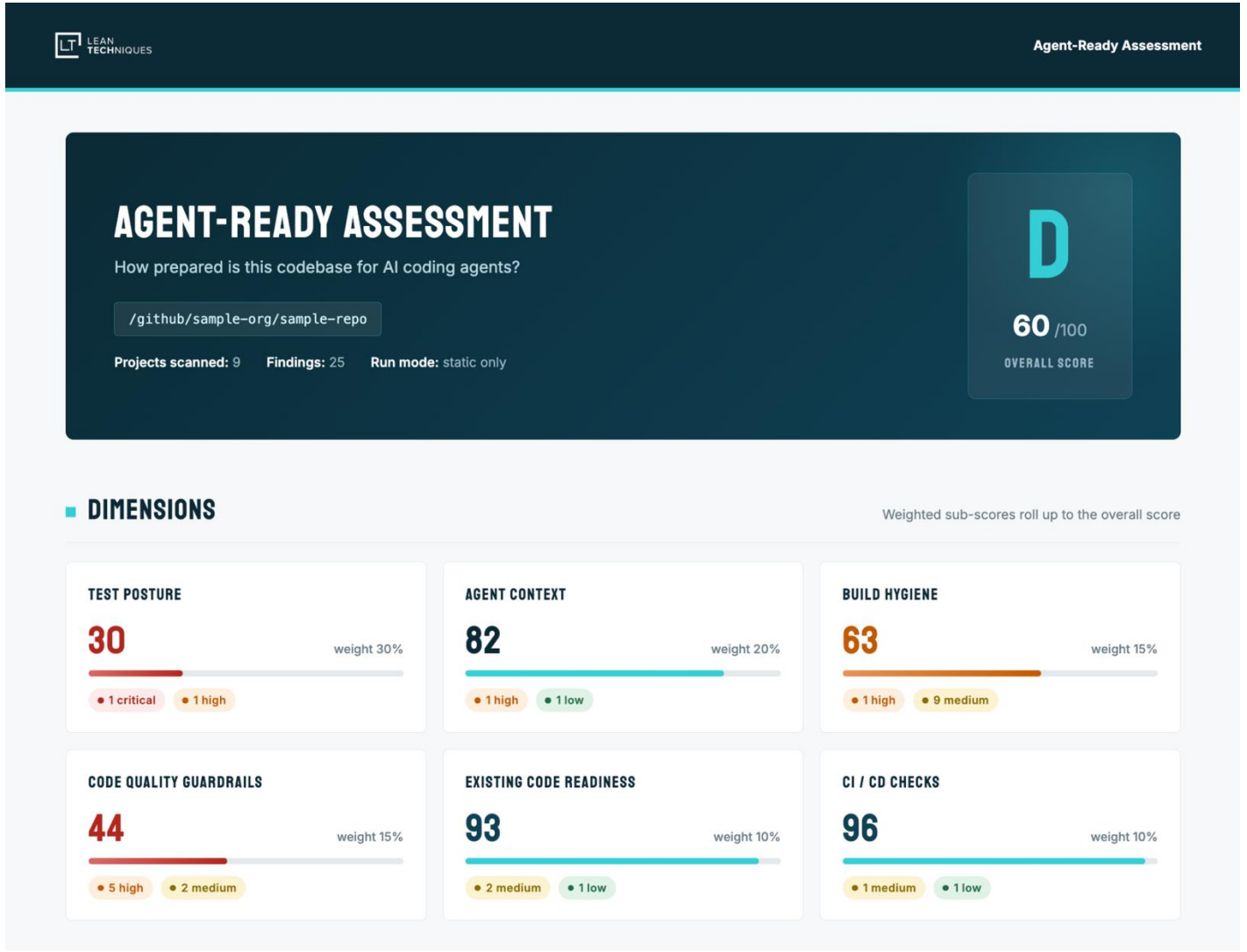
These tools are the worst they will ever be today.

Alright let’s talk AI Tools

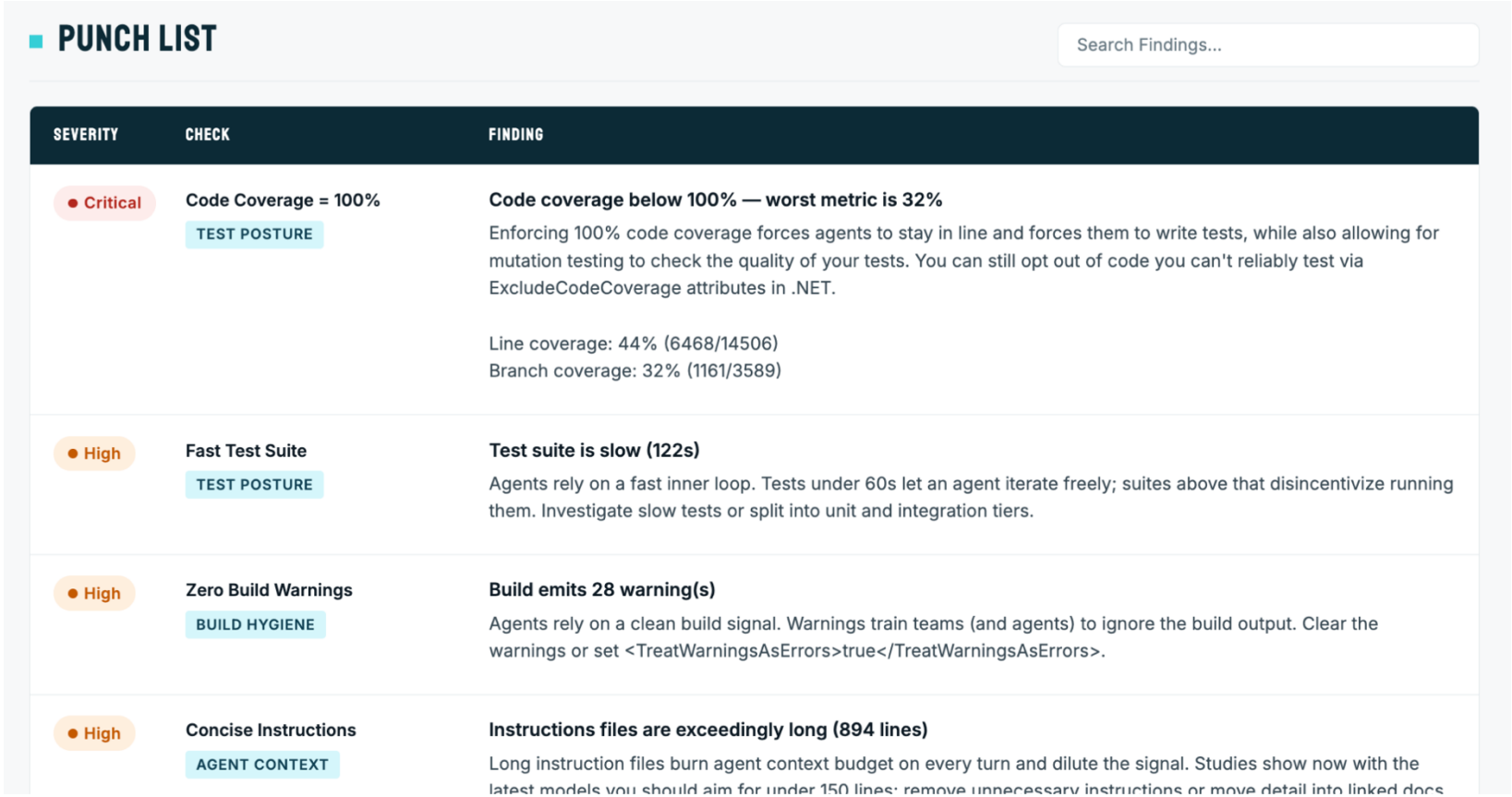
AI is an Amplifier

- Good practices – move faster at high quality
- Bad practices – move faster at lower quality
- How do you verify your app works? Compile, Tests, Lint, Format. Are you doing those things already?
- Do you have judgement to know when AI went off the rails?
- Your PR output might 2-10x when using AI correctly
- Do your PRs sit for days without reviewing already?
- Do you rubberstamp PRs?
- You still own the code – not AI

Agent Ready Assessment



Agent Ready Assessment



PUNCH LIST

Search Findings...

SEVERITY	CHECK	FINDING
Critical	Code Coverage = 100% TEST POSTURE	**Code coverage below 100% — worst metric is 32%** Enforcing 100% code coverage forces agents to stay in line and forces them to write tests, while also allowing for mutation testing to check the quality of your tests. You can still opt out of code you can't reliably test via ExcludeCodeCoverage attributes in .NET. Line coverage: 44% (6468/14506) Branch coverage: 32% (1161/3589)
High	Fast Test Suite TEST POSTURE	**Test suite is slow (122s)** Agents rely on a fast inner loop. Tests under 60s let an agent iterate freely; suites above that disincentivize running them. Investigate slow tests or split into unit and integration tiers.
High	Zero Build Warnings BUILD HYGIENE	**Build emits 28 warning(s)** Agents rely on a clean build signal. Warnings train teams (and agents) to ignore the build output. Clear the warnings or set <TreatWarningsAsErrors>true</TreatWarningsAsErrors>.
High	Concise Instructions AGENT CONTEXT	**Instructions files are exceedingly long (894 lines)** Long instruction files burn agent context budget on every turn and dilute the signal. Studies show now with the latest models you should aim for under 150 lines; remove unnecessary instructions or move detail into linked docs

Where I’m at today

- AI writes 99% of my code
- I review every line – I’m still responsible
- Speed is 10x faster, Quality is Higher – forced me to automate more
- Running 2-4 agents at once
- Mostly use the CLI – you should start here – starting to use Copilot App (when using Copilot)

Large Language Models

- Engine trained on massive amounts of text to learn patterns of meaning
- Literally just predicts one token at a time based on probabilities
- Best Models is a misnomer – it's tradeoffs
- Accuracy vs Cost vs Speed
- Opus is the "best" model for one-shotting code in my experience, but it's the most* expensive and often slow (* until Fable)
- Haiku responds very quickly, but isn't as deep
- Sonnet tries to balance these two

Context

- The information an LLM is allowed to consider for a request
- LLMs have a limited Context Window
- What makes up Context:
- Conversation History
- Code in your repo
- User Prompt
- Instructions
- Clearing context is key – when one task is done, /clear
- Compacting can also lead to hallucinations

AI Dev Tools

- Unifies LLMs with codebases to provide the most context to get the best results
- GitHub Copilot (most ubiquitous)
- Claude Code (best AI dev tool)
  - Preview Feb ‘25, GA May ‘25
- Cursor (best autocomplete)
- Lots of others
- Best == my opinions and experiences
- Autocomplete is nice… but you shouldn’t be using it anymore

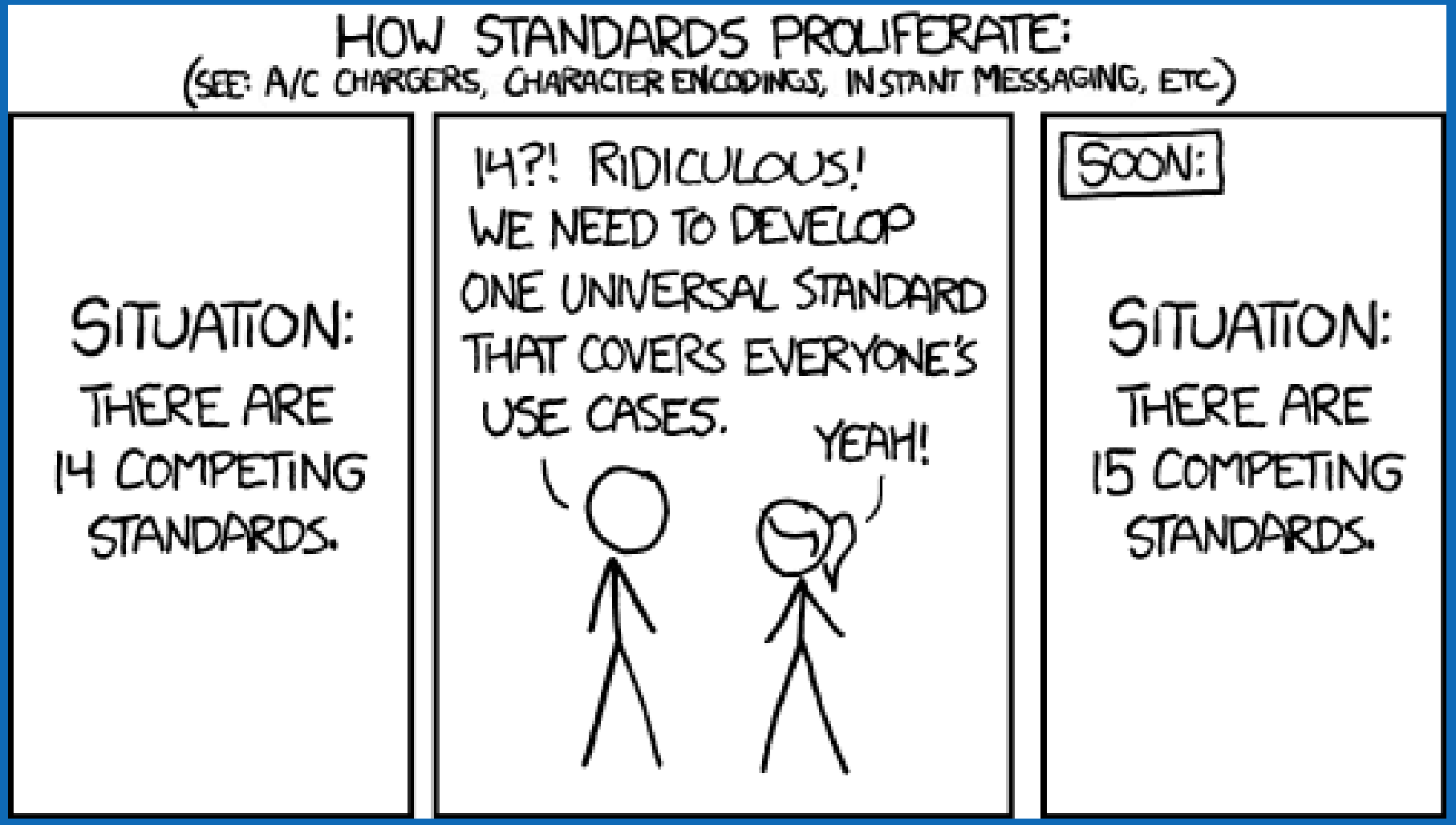
Instructions

- Set of Markdown files to define your team's/organization's rules
- "Don't use Arrange, Act, Assert comments in tests"
- Tip: anything the agent does wrong? Add it to your Instructions
  - This is key
- Used to be you should tell AI about your standards, your folder structure, etc
- Nowadays – delete everything except Don'ts per Boris Cherny


INSTRUCTIONS
INSTRUCTIONS EVERYWHERE
imgflip.com

Instructions Everywhere

- Every tool now respects AGENTS.md in the root – except Claude Code
- .cursorrules, .cursor/rules
- CLAUDE.md
- GEMINI.md
- .github/instructions/copilot-instructions.md


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

Skills

- Rather than loading all those Instructions into context every time, what if we just did it selectively as needed?
- Also can be used to automate workflows
- .github/skills/<skill name>/SKILL.md
- .claude/skills/<skill name>/SKILL.md
  - Recognized by Copilot too
- .agent/skills/<skill name>/SKILL.md
- Just became a thing in October 2025
- https://github.com/anthropics/skills

Skills

skills > frontend-design > SKILL.md

```
---
name: frontend-design
description: Create distinctive, production-grade frontend interfaces with high design quality. Use this skill when the user asks to build web components, pages, artifacts, posters, or applications (examples include websites, landing pages, dashboards, React components, HTML/CSS layouts, or when styling/beautifying any web UI). Generates creative, polished code and UI design that avoids generic AI aesthetics.
license: Complete terms in LICENSE.txt
---

This skill guides creation of distinctive, production-grade frontend interfaces that avoid generic "AI slop" aesthetics. Implement real working code with exceptional attention to aesthetic details and creative choices.

The user provides frontend requirements: a component, page, application, or interface to build. They may include context about the purpose, audience, or technical constraints.

## Design Thinking

Before coding, understand the context and commit to a BOLD aesthetic direction:
- **Purpose**: What problem does this interface solve? Who uses it?
```

Skills vs Commands

- Both automate tasks or workflows
- Commands can only be invoked by a user by /command-name
- Skills can be invoked by a user (by /skill-name) **or an agent**
- In general – prefer Skills

Skills

- /grill-me <abc-123>
- /grill-me Add a button to this page

```
---
name: grill-me
description: Interview the user relentlessly about a plan or design until reaching shared understanding, resolving
each branch of the decision tree. Use when user wants to stress-test a plan, get grilled on their design, or mentions
"grill me".
---

Interview me relentlessly about every aspect of this plan until we reach a shared understanding. Walk down each
branch of the design tree, resolving dependencies between decisions one-by-one. For each question, provide your
recommended answer.

Ask the questions one at a time.

If a question can be answered by exploring the codebase, explore the codebase instead.
```

Starter Workflow

- 👨 Switch to Plan Mode
- 👨 /grill-me <Jira ID or Task>
- 👨 Answer any questions
- 👨 Review and Approve AI generated Plan
- 🤖 AI executes the code
- 👨 Reviews what it did

Demo

Introducing /ship-it

1. 👨 /grill-me on a GH issue, Jira issue, or just free text
2. 👨 Approve plan and go on auto/autopilot mode
3. 🤖 Create a new worktree
4. 🤖 Implement the plan
5. 🤖 Run the verify script til it passes (formatting, linting, tests, etc)
6. 🤖 Run /code-review and fix anything you have a lot of confidence in. If it's a code style, back it up with reduced cyclomatic complexity or another metric that proves the code is better
7. ❓👨 Present anything you're unsure about for approval (rare maybe 2% of the time)
8. 🤖 Run /security-review and fix anything you have a lot of confidence in. If you're unsure then write a test to prove there's an issue
9. ❓👨 Present anything you're unsure about for approval (rare maybe 1% of the time)
10. 🤖 Open a Draft PR in my particular format using the /concise-pr skill
11. 🤖 Watch CI and fix any build errors. If a test is flaky, root cause and fix it
12. 🤖 Watch for the Copilot Code Review and fix any comments
13. ❓👨 Present any comments you're unsure about for my review (uncommon, maybe 25% of the time).
14. 🤖 Notify me when PR is green with no comments and ready for me to review
15. 🤖 Mark PR ready for review (move out of Draft)
16. 👨 Approve or request changes

Demo

Top 10 AI Tips Today

1. Give AI a way to validate itself - compile, tests, format, lint, acceptance criteria, code quality, etc
2. Get good with the CLI first then graduate up (Copilot App is the best desktop UI imo)
3. Start with Plan Mode by default - graduate out of it when you feel comfortable
4. Auto Mode via `claude –enable-auto-mode` or autopilot in Copilot
5. Skills
6. CLIs over MCP servers - GitHub, Atlassian, Azure, etc
7. Be aggressive clearing context, <200K tokens is the Smart Zone… >200K is the Dumb Zone
8. Use AI to do code reviews - in GitHub and locally
9. Keep your AGENTS.md lean - cross cutting things, how to verify your app, not your code structure, delegate specific concerns to Skills
10. Keep an 👀 on AI

𝕏 @scottsauber

How do I stay up to date?

- Follow people on Twitter / LinkedIn
- @bcherny – Boris Cherny who created Claude Code
- @claudedevs – Claude Code Devs
- @_evan_boyle – Evan Boyle, Copilot
- @leeerob – Lee Robinson, Cursor
- @code – VS Code
- @housecor – Cory House, consultant
- @scottsauber – a humble peasant

Takeaways

- Ideas on where AI is and where it’s going
- Ideas you can start using literally today
- Now’s the time to jump in, because the productivity is real
- “Change has never been this fast but will never be this slow again.”

Resources

- This slide deck at https://scottsauber.com
- If you're interested in us 2x-ing your speed and uplifting your quality with AI Agents – talk to me

Questions?

ssauber@leantechniques.com

Add me on LinkedIn:

@scottsauber

@scottsauber.com

Slides at scottsauber.com

Thanks!

Add me on LinkedIn:

Slides at scottsauber.com