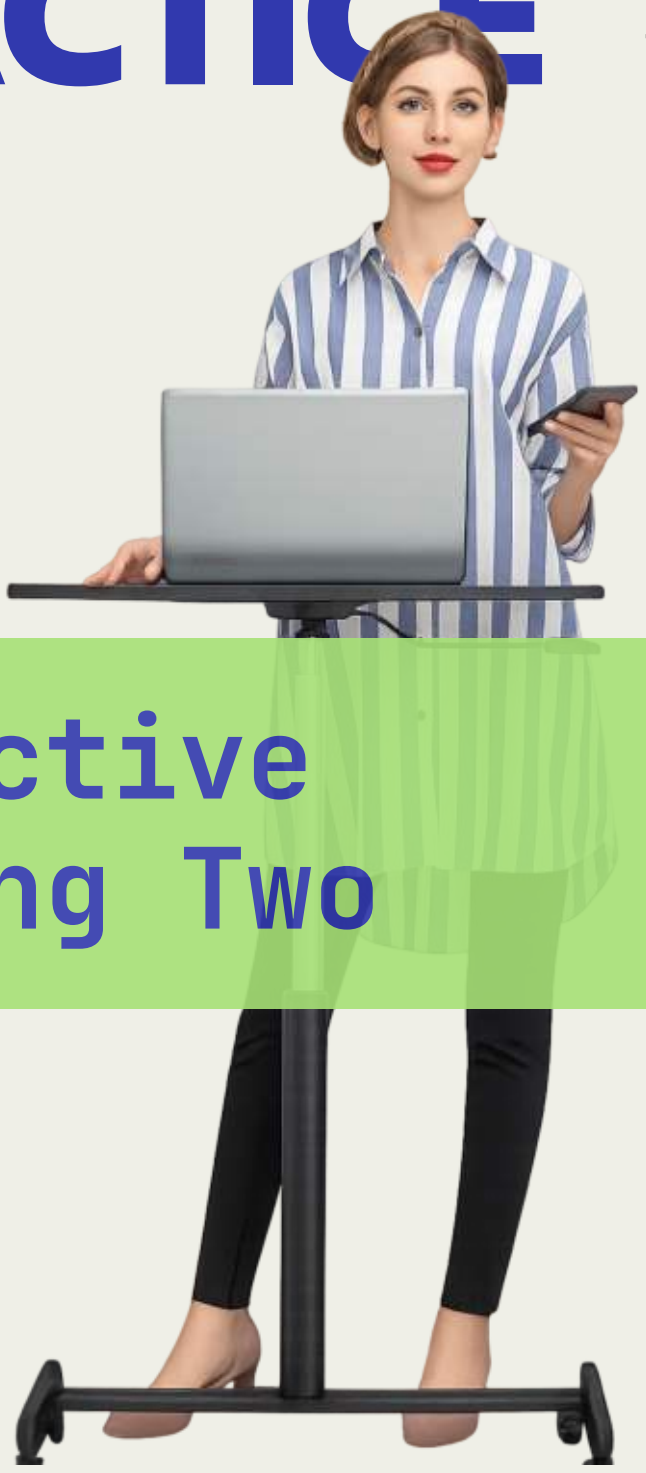
PIDP: 3260

PROFESSIONAL PRACTICE

Instruct

Author: Yuri Tricys
Date: August 23th, 2025



Reflective Writing Two

Objective

Chapter 16 of Stephen Brookfield's book, "The Skillful Teacher", titled: "Understanding student's resistance to learning", lists multiple reasons teachers may encounter student resistance to learning.

Some of the headings in his chapter include:

- Poor self-image as learners
- Fear of the unknown
- A normal rhythm of learning
- A disjunction of learning and teaching styles
- Apparent irrelevance of the learning activity
- Level of required learning is inappropriate
- Fear of looking foolish in public
- Cultural suicide
- Lack of clarity in teachers' instructions
- Students' dislike of teachers
- Going too far, too fast

These headings are each followed by explanations of how they may prompt students to resist learning. The quote below is from the last heading, 'Going Too Far, Too Fast.'

> *"[...] I know an error I've made multiple times over is to push too far, too fast. Because I find critical thinking so intriguing and pleasurable, it comes easily to me. Consequently, I can easily mistime my efforts to get students to apply it to their own reasoning and actions."*
>
> *(Brookfield, 2015, P. 224)*

The quote addresses the pitfall of accelerating student learning beyond their readiness, particularly in critically challenging material.

Brookfield acknowledges his own tendency to rush students into critical thinking exercises because he finds the process inherently stimulating, creating a mismatch between his enthusiasm and their capacity to engage. This concept parallels web development instruction, where pushing students too quickly into advanced tooling (e.g., React, Node.js), or complex frameworks, without mastering foundational skills (HTML/CSS, JavaScript syntax) risks confusion and disengagement.

What caught my attention is the disconnect between the instructor's expertise and the learner's vulnerability – highlighting how assumptions about "natural" enthusiasm can undermine effective teaching.

Reflective

I chose this quote because I secretly suspect it mirrors challenges in teaching web development, a field where staggered progress is crucial.

Programming is inherently difficult for several reasons, including:

1. **Abstract Concepts** (e.g., closures, asynchronous programming, modularity) require time to internalize without real-world analogs.
2. **Cumulative Structure**: Skipping CSS layout logic, or naming conventions, before teaching Flexbox or CSS Grid can lead to partial understanding and bad habits.
3. **Varying Baselines**: Beginners in coding often lack familiarity with logic structures or computational thinking, making rapid-fire lessons inaccessible.
4. **Mastering Tool Use**: Configuring tools takes time away from understanding material — yet it's an essential skill for productivity.
5. **Conceptual Leaps**: Moving from basic syntax to patterns, like MVC (Model-View-Controller) and OOP (object-oriented programming), demands significant cognitive restructuring.
6. **Hidden Dependencies**: Understanding *why* a framework does something requires context about underlying systems (HTTP, browser rendering) often glossed over.
7. **Tool Overload**: The sheer volume of tools boxes can distract from actual learning goals, leading to cognitive fatigue (e.g., 20 frameworks to choose from).

Like Brookfield, I romanticize the excitement of my area of expertise because I love it, but can neglect the immense amount of gradual scaffolding needed.

Programming isn't just a cognitive shift, it's technical overload. Students require time to adapt to new languages, tools, and mindsets.

Interpretive

The core message is that learning is non-linear — students need space to process, fail, and revisit concepts. Translating this to web development, the practice of throwing everything at them all at once, to see what sticks, could be revisted. Rushing into sophisticated libraries or APIs before solidifying basics like base language principles, build pipeline practices, or even responsive design may leave learners feeling overwhelmed and incompetent.

My instinct is always that faster coverage equals better preparation, but I can easily overlook the *cognitive load* novice programmers endure, not to mention other distractions students have to deal with in their day-to-day lives.

Brookfield's reflection is helpful while I reshape my view: effective instruction prioritizes incremental mastery alongside *the bigger picture*, not after it.

Decisional

To apply this insight:

1. **Map Cognitive Steps**: Break projects into modular tasks (e.g., teaching form validation in vanilla JavaScript before integrating it into a React app).
2. **Formative Feedback**: Use low-stakes checkpoints (e.g., code quizzes, peer debugging sessions) to align pacing with class comprehension.
3. **Differentiated Pathways**: Offer tiered labs (beginner, intermediate, advanced) for topics like state management using React or Next context, ensuring no one is left behind or bored.

4. **Normalize Struggle**: Share my own early coding failures (e.g., battling CSS specificity) to destigmatize slowness as a learning phase.
5. **Scaffold Tools**: Introduce frameworks only after core syntax and problem-solving patterns are practiced, much like building critical thinking through iterative debate rather than instant analysis.

By respecting the **"rhythm of learning"** Brookfield describes, my teaching can evolve from pushing learners toward technical outcomes to guiding them through the vulnerabilities of becoming proficient – a shift that prioritizes depth over speed.

References

Brookfield, S. D. (2015). The skillful teacher. On technique, trust, and responsiveness in the classroom. Jossey-Bass: John Wiley and Sons Inc.

AI Models Used In This Report: The ideas, structure, writing, and editing in this paper were performed by the author. Various AI models were used to research, collect, and verify data, format arguments, and grammatically structure content. Models used include: Qwen 2.5 235b a22b. Meta Llama 3.1 405b Instruct. MistralAI Devstral Small 2505, GPT-5, GPT-Image-1, Google Gemma 3 27b