

How education makes your design system click

by Ashley Zhu

Your design system probably isn't failing because of missing components or messy libraries. It's failing because people don't feel confident using it.



Your design system probably isn't failing because of missing components or messy libraries. It's failing because people don't feel confident using it.

We launch systems like products, with big announcements and polished documentation. Yet we keep battling the same questions every day, limited adoption, and (maybe) still a touch of skepticism about why the system even matters.

That's not a system failure. It's a failure in education.

This article explores how to design education that sticks and creates that 'magic moment' to turn users into confident advocates — the moment your system moves from optional to essential.

When confidence becomes the blocker

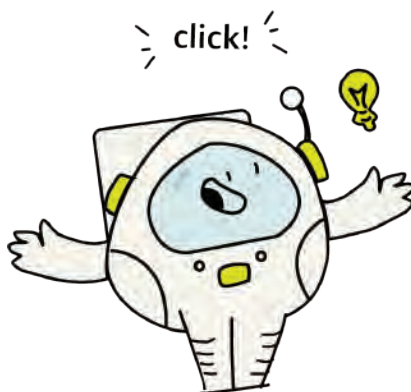
29

During a round of user-testing interviews, a designer said something I'll never forget:

"I think I'm using the button correctly... but I'm not confident about it."

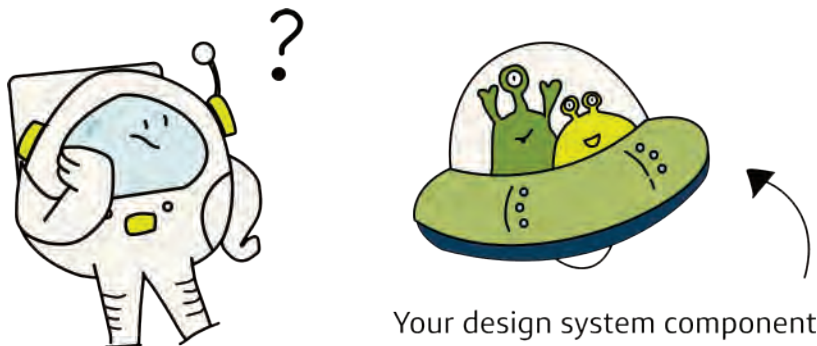
And so they hesitated to use the component at all. They knew what it did, but weren't sure they were doing it right. Applying it in their own work felt almost alien.

That's when it hit me: A design system isn't successful when it ships; it's successful when it sticks. Successful adoption comes



from teaching system fluency — helping people understand not only why it matters, but also how it works for them.

The spark that turns awareness into fluency is what I call the click moment: when someone goes from “I don’t get it” to “Ohhh... I get it! I love this. I know how to use it.”



Your design system component

Why design systems stall

Design system education tends to hit against two main pain points:

1. **Unstructured learning for consumers.** Endless docs to scroll through, property panels to sort, lecture after lecture. People drown in information before they even start building — reluctant to engage because learning the system feels optional, overwhelming, and without a clear path forward.
2. **Unsustainable teaching for the system.** The same questions appear again and again, because information isn’t sticking. The system team repeatedly becomes the catch-all for every question and design decision, owning too much and burning out.

Both challenges share the same root cause: We’re fighting against how people actually learn.

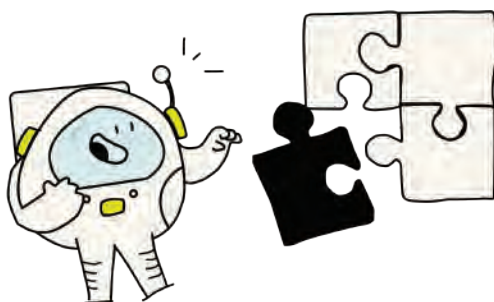
Research shows we forget 70% of what we read or hear within a day, and our working memory only holds a few ideas at once (Source: *Neurolaunch, Ebbinghaus Psychology*). People don’t learn deeply

by reading docs; they learn by doing, repeating, and connecting in *context*.

Dump a 100-page text about buttons on someone, and mostly it feels overwhelming, forgettable, and exhausting. That's because our working memory can only handle 3-5 things at once (*Source: NLM, Cognitive Load Theory*). People don't want a manual; they want movement and real-life examples. They want to feel empowered. Design systems can't succeed just because they have components. They succeed because their community feels confident using them.

So what if we treated our design system as a learning system? What if we didn't just launch design systems like products, but also designed education like a product?

Confidence
is the missing
component



31

Design education as a product

Education has users. It has journeys. It has outcomes. It needs feedback loops and iteration. And just like any great product, it requires a scalable, sustainable strategy for both system creators and consumers.

When we design for education, we design for confidence. We create the conditions for that click moment, when the system goes from "something I've heard I should use" to "something I want to use."

Here's a simple framework to help your system click into motion: **Invite. Involve. Inspire.**

Invite

Invite says: You belong here. The system is for you. We'll meet you where you are.

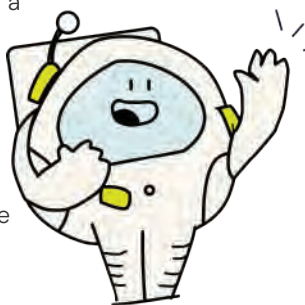
Give a warm welcome.

From day one, make the system approachable. Joining a company and being expected to design with a system can feel intimidating. Embed system training into new-hire onboarding so new teammates view it as the default way we work, not as an extra tool.

Invite also means making the system human and fun. Show who's behind it through names, stories, and real examples. No one should feel silly asking a question. Position your team as allies, not gatekeepers. When people feel welcomed, they stay curious.

Meet people in the tools they use.

Why? When learning happens inside Figma or code, it feels a little less side-of-desk to learn the system, and more essential to their work. Psychologists call this "context-dependent memory": Learn in the tool, remember in the tool (*Source: ScienceDirect, Context-Dependent Memory*). Try adding documentation tips inside Figma libraries, such as sandboxes, property descriptions, or notes that link to quick videos. Meet people where they are.



Offer different paths for different skills.

Designers, engineers, and PMs don't need the same thing, and they all come in with different levels. By tailoring learning paths, you reduce overload and increase relevance. Motivation spikes when learning feels personal, and their path is clear.

Involve

Involve means: Let's try it together.

I've loved to dance my whole life. Yet while I can

pick up a few things by watching, I learn best by stepping onto the floor. Design systems are no different. Learning sticks when it's active, hands-on, and repeatable.

Build rhythm in action.

You don't learn choreography by reading the steps; you learn it by doing. Many users need to build new experiences with the system, even when they haven't seen direct equivalents before. That's where panic sets in. Host design-alongs or code-a-thons where designers or engineers work side by side with the system team. Learning through doing together turns education into community.



Create a design system habit.

Education isn't one big event; it's a series of small steps. Share a quick tip of the week, a fun challenge, or a short demo in Slack or Teams. Spaced repetition works: When people revisit learnings regularly, retention skyrockets. Small doses, spaced out, beat a giant info dump. These micro-repetitions are what turn knowledge into instinct.

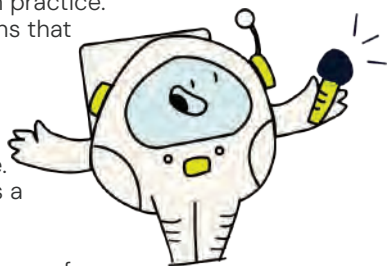
33

Build playgrounds, not lectures.

Create Figma "playgrounds" where designers can explore, make mistakes, and learn through practice. Offer self-paced, hands-on system lessons that move from beginner to advanced.

Inspire

Inspire is where learning becomes culture. No one team can do it alone. Education is a community effort.



Spotlight real stories. Celebrate creative uses of the system. Let engineers, designers, and PMs show how they solved problems with shared principles. When people see their peers implementing, say, design tokens with the system, trust grows much faster than any system campaign ever could. Partner with onboarding teams, tools teams, and system champions across the org. The goal isn't fewer questions, but more people answering them.

What does that look like in practice? It's ambassadors running their own workshops, engineers volunteering to join because they see the value, and conversations flowing both ways — not just consumers adopting, but contributing back. Education has created system advocates, not dependents.

And here's the magic part: Inspire creates momentum. When people feel confident, they feel proud. They share, they lead, and systems thinking takes off. That's the final step that turns the system from a tool into the way we love to work.

Why this works

People don't learn through docs alone — they learn through practice, confidence, and community.

- When learning is active, it sticks.
- When confidence grows, adoption scales.
- When a community takes ownership, the system sustains itself.

The best design systems are social systems. They thrive on shared energy, not top-down enforcement. You'll know it's working when peer-led sessions multiply, repetitive questions fade, and designers start teaching others without being asked. That's when adoption becomes momentum. It's the difference between "Leadership told us to use this" and "Hey, this is awesome, let's try it."

So how do you create that click moment in your own system?

Start small: Create a mini-tutorial in Figma or code for one component. Post a "tip of the week" in your team channel. Highlight a peer who used the system beautifully. Ask for feedback, and act on it.

Each action sparks another. Each moment of confidence builds momentum. Until one day, your system isn't just used. It's loved.

That's the moment it clicks. Your design system doesn't just work: It takes flight.

“
**When we
design for
education,
we design for
confidence.**
”

