

Every User's Plea

“Why can’t it Just Work?” That’s the plea of every user who confronts tech that they can’t immediately figure out. Whether they’re surfing the web for directions, enduring permahold on a support line, or just banging their heads in frustration, that’s what they’re thinking.

I agree with them. Today’s tech can Just Work, and it should. This book will show you how.

Where We Came From

The builders of the earliest computer programs didn’t care about making them easy to use. A computer’s time was enormously expensive, much more so than a user’s time. Forcing the human user to memorize complicated commands and type them into a dumb terminal, instead of using computer power to provide a menu listing them, made economic sense.

The relative costs are now reversed, but almost everyone in the industry older than about 30 grew up in that type of environment. It can't help but shape our thinking today, no matter how hard we try to leave it behind. Think of your older relatives who grew up in the Great Depression of the 1930s, who even today can't bear to throw away a sock with only one hole in it.

Like driving a car in the early twentieth century, early users expected computers to be a pain in the butt, and we were rarely disappointed. Almost all users were programmers themselves. Few of them felt the need, or often even the desire, to make things easier. We accepted the difficulties—the rationed computer time, the arcane commands, the awful documentation (if any even existed)—as those motoring pioneers accepted hand-cranked engines and constant tire punctures.

It was the best anyone had. We were happy to get our important computing jobs (tabulating the census, cracking enemy codes) done at all, as they were happy to stop shoveling horse manure out of the barn every day. We liked fiddling with our programs, using them in ways their designers never intended, as the early motorists liked tinkering with their engines. If someone had told Henry Ford that his Model T needed a cup holder, he'd have laughed in that person's face.¹

There was a feeling in those days that making programs easy to use was just plain wrong. If a program was hard to write, it *should* be hard to use so that only those who had proven themselves worthy through intellectual struggle could benefit from the programmer's effort. I remember, with surprising fondness even today, the pride I felt on discovering that the command to print a document on the first major computer system I ever used² wasn't Print or P but, rather, the letter Q, since you were placing the document in a queue to be printed. I had learned a magic word. I was becoming one of the elect. I was Smart!

¹ My recently retired Honda minivan contained 17 cupholders for 8 seating positions, so I could fill it with two-fisted drinkers and still have one cupholder left over.

²A DEC PDP-10, freshman year in college, fall of 1975.

But as hardware got smaller and cheaper, computers moved from the air-conditioned glass rooms attended by high priests, to the workbenches of geeky hobbyists, then to the desktops of individual employees, and today to the pockets of over half of humanity. So, developers had to start building tech that users could actually use.

These old attitudes still bedevil the industry. In early 2026, I attended a talk on AI. During discussion, I pointed out that an important operation was difficult—not just a little difficult but tonsillectomy-through-the-rectum difficult. The speaker replied, “Yeah, well, you just have to learn to use the tool.” That took me back to my freshman year in college, half a century ago. “It’s not P to print; it’s Q. Mind your Ps and Qs, that’s how I remember it.” Today, in 2026? Seriously?

Yes. Far too often. It’s wrong. It’s bad. More to the point, we can’t make money that way. To succeed in today’s brutal marketplace, we need to make our tech—our apps, websites, programs and devices—Just Work. Here’s how we’ll do it.

Fundamental Principles

Every programmer thinks he knows exactly what users want. He uses a computer all day, every day, so he ought to know. He says to himself, “If I design a program that I like, the users will love it.” Wrong! Unless he’s writing programs for the use of burned-out computer geeks, his user is not him. I tell my students to engrave on their hearts, along with “garbage in, garbage out,” Platt’s First, Last, and Only Law of Tech Design, the Golden Rule, the Ground Truth, the One True Axiom:

Know Thy Users, for They Are Not Thee.

Of course, knowing a principle is one thing. Following it is trickier. These three corollaries will guide you in putting that fundamental law into practice.

First corollary: The tech you build has zero value in and of itself. You build tech for one reason and one reason only: to make the user of your tech happier or more productive than she would be without it. The only value that your tech ever has or ever will have is the degree to which it increases the happiness or productivity of its users in some way. It is extremely rare that customers are willing to pay for tech that decreases their overall happiness or productivity, although that's far too often what they end up with.

Second corollary: Tech increases the happiness or productivity of users in one of two ways. It can help a user solve a specific problem—writing an email message or an article, buying groceries or an airplane ticket, viewing a bank statement and paying bills. The user's only goal in these cases is to finish the task and get on with her life, or at least on to the next task. The faster your tech converts a task from not-finished to finished, the happier the user will be. Completing a task in 10 minutes is better than 11 minutes, and 9 minutes is better than 10. Examples of this type of application are word processors and online bill payment apps.

The second way in which your tech can increase a user's happiness is by putting the user into a pleasurable state, which she wants to maintain as long as possible, or at least as long as it remains pleasurable. Games fall into this category, as do media player apps such as Spotify. My favorite example of this type of application is Zoom and its predecessors, with which my daughters made video calls to my parents, starting at ages 5 and 3 and continuing to this day 20 years later.

Third corollary: In neither of these cases do users want to think about the tech they are using. At all. Ever. Even a little bit. In the former case, they want to think about the problem they are solving: the wording of the document they are writing or whether they have enough money to pay their bills and which unpaid creditor would hurt them the most. They don't want their tech to distract them from the problem they're thinking about. If they want to goof off, they'll bring up Solitaire, not admire the dancing paper clip in a word processor. In the latter case, users want to enter that pleasurable state as quickly as they can and stay there

as long as they can. Anything that delays the start of their fix, or distracts them from it while they're enjoying it, is even less welcome than the interruption of a work task. My parents want to see and talk with and gush over their grandchildren, and my children their grandparents. Any attention that the program diverts from this purpose to itself is a negative and must be minimized.

To summarize these three corollaries, users don't care about your tech, in and of itself. Never have, never will. Your mother might, because you built it and she loves you, and then again she might not; but no one else does. Users care only about their own productivity or their own pleasure. Every single user, of every single piece of tech, every single time, wants what Don Norman calls *The Invisible Computer*, in his landmark book of that title. In other words, they want their tech to Just Work, without their having to think about it.

Make It Just Work™

So how about this novel idea: why don't we make our tech Just Work? Users shouldn't have to read a manual (remember those?) or watch a training video. It is the designers' job to build tech that explains itself. That makes its functions obvious. That speaks the users' language, instead of making users learn its language. That provides good default settings. And so on.

Here's the first of many examples in this book. It's an oldie but a goodie—an early example of a tech operation done right, then wrong, and then even more right.

Early Example of Tech Just Working, Then Not, and Then *Really* Just Working

Travel with me back in time, back to when the earth was still cooling off and the last few dinosaurs tottered around, wondering where all their friends had gone. Back before the internet existed,

back when companies charged for individual long-distance calls, back when I had more hair.

If you wanted to phone someone (we didn't differentiate between voice and text, because the latter didn't exist yet), you needed their number. These were written down in large paper directories, popularly called "the phone book," (Figure 1.1) sent to phone subscribers by their carrier.³ If you didn't have the book for your desired callee's location, you would dial directory assistance, 411 locally or 1-(area code)-555-1212 if outside your local calling area.



Figure 1.1: A paper telephone directory, "the phone book"

Source: arti om/Adobe Stock Photos

AT&T's directory assistance was originally simple and easy. They'd ask, "What city?" then "What listing?" You'd specify these, and the automated voice would say, "The number you requested is [whatever]. Please make a note of it." If you stayed on the line, it would repeat the number so that you could be sure you'd

³At the time, it was rare to have a choice of carrier, so we referred to them as simply "the phone company." See Lily Tomlin's hysterical video at <https://vimeo.com/355556831>: "We don't care. We don't have to. We're the phone company."

written it down correctly. Simple. Easy. Impossible to screw up. It Just Worked, by the standards of the time.

Then AT&T added the capability of automatically dialing the number that they'd just found for you. They'd say, "The number you requested, [whatever], can be automatically dialed for an extra charge of 50 cents. Press 1 to accept and 2 to decline." The simple operation of finding the number was as simple as ever, and the newer, more powerful feature was available to those willing to pay for it. Anyone who didn't care about the new feature could simply hang up after writing down the number, which they'd be doing then anyway.

Then some high-ranking idiot had an awful idea. The last time I used AT&T directory assistance on my land line, before telling me the number, it said, "The number you requested can be automatically dialed for an extra charge of 50 cents. Press 1 to accept and 2 to decline." It wouldn't give me the number until I entered my choice. I had to take the phone away from my ear and visually reacquire the keypad (which gets harder after age 45 or so). I then had to put down the pencil I was holding in my other hand to write down the number when they gave it to me, press the correct button, and then pick up the pencil again and put the phone back to my ear. Only then would it tell me that the number was [whatever]. The new, powerful operation was possible, but the simple operation was no longer simple. That's Not Just Working, by any standards, then or now. I'll add a fourth corollary to those earlier: when something Just Works, for %*&# sake, Don't Break It.

My cell carrier, Verizon, on the other hand, thought carefully about their users and raised Just Working to new heights. Verizon realized that almost everyone calls directory assistance because he wants to call someone immediately, so why not just do that? I was driving around with my wife, and we wanted to visit the New England Aquarium, but were they open now? They had a website, but this was before internet-capable phones. I dialed directory assistance from my dumb cell phone, asking for the aquarium's number. The automated voice said, "The number is

(617) 942-6960. I'll connect you now." It happened automatically, without any action or even thought from me. The new number stayed on my phone's recently dialed list so that I could add it to my contact list if I wanted. Any requesters who didn't want to call the provided number immediately, who only wanted to write it down for later access, could simply hang up, which they'd be doing then anyway. Simple things remained simple. Powerful and convenient things became simple, too, especially important while driving. This design is as good as AT&T's is bad. Verizon made this operation Just Work, by the more stringent standards of that day, and raised the bar for everybody else.⁴

So What Do We Do?

Someone has to speak for the silent majority of users who don't give a flying fish about technology for its own sake, who just want to get their work done so that they can get on with their lives, or into their pleasurable state. I try to fill this role at every design review I attend. "You're like the guys who design the drills that they sell at Home Depot," I tell the programmers. "You're arguing

⁴It is surprisingly difficult to find the Aquarium's phone number on their website today. Their home page does not contain one. You have to scroll all the way to the bottom to find a small link labeled "Contact Us." When you click that, you get another page that contains a contact message form. You have to scroll all the way to the bottom of this page to find the "Phone Numbers and Addresses" section, and you have to click the + sign to open. You then have to pick out the specific department you want, of which the first two are "Corporate Partnership" and "Development." You could try calling "Gift Shop" or "Membership," and they'd probably tell you, but there's no "General Information" or "Visit Information" phone number. The usual search engines produce a phone number, but when you call it, the recorded message doesn't even mention the Aquarium's name. It just says, "Thank you for calling. Our offices are currently closed. Our hours are. . . ." They really don't want to talk to you on the phone. Maybe providing people to answer it costs too much. In fairness, I have to say that their hours are prominently posted on their home page. Viewing them there Just Works better than talking to a human being.

over the internal details of drills—ball bearings versus roller bearings versus air bearings—with each of you claiming that’s what your customer wants more than anything in the world. Wrong. The customer doesn’t care about your drill for its own sake, not one tiny bit. Never has, never will. He doesn’t go to Home Depot because he wants a drill. He goes to Home Depot because he wants holes. He’d much prefer buying a box of holes if he could, without having to touch a drill.⁵ Your drill is a necessary evil in your user’s quest for holes. Now ask yourself, and answer truthfully: “What kind of hole does your user really want, and how is your tech going to get him better holes, faster, for less money?” I wrote this book to help you get there.

Here’s What’s Here

Nothing teaches like a vivid example, either of good design or bad. I’ll show you plenty of both in this book. Together we’ll critically review apps and websites that Just Work or Don’t, sometimes both in the same package. You will see how making our tech Just Work requires us to challenge our long-held thought processes and attitudes. We’ll rethink some long-held assumptions and rediscover principles that we’ve forgotten. Here are the topics we’ll be examining together:

Chapter 1, “Every User’s Plea”: You’ve read this far, so you already know what’s here. And if you skipped over it to get here, tough noogies; go back and read it.

Chapter 2, “Artificial Intelligence”: AI is sweeping the computing world now, but it’s still in its toddlerhood. Like any toddler, it tests limits and throws tantrums, trying to figure out how the world works. I discuss what’s hype and what’s real and where you can dive deeper without the hype.

⁵Remember Ringo in the Beatles’ cartoon movie *Yellow Submarine*? “I’ve got a hole in my pocket....” See <https://youtu.be/EK0R272RjXI>.

Chapter 3, “Going Mobile”: More than half of humanity carries a smartphone in their pocket every day. About 15 years old today, this segment of industry is in its adolescence, struggling to mature into adulthood. We’ll check in on its progress.

Chapter 4, “Tangled in the Web”: About 30 years old now, the world wide web is an adult in its prime, usually more or less comporting itself as it should. We will examine what has survived from its early days and what has been cast aside.

Chapter 5, “Keep Me Safe”: Any time our software accesses anything outside our platform, we encounter attacks from bad guys. Developers do their best to minimize this risk and mitigate any damage. But our users are human and often won’t cooperate with security operations in the ways that developers demand. This chapter shows how making our security measures Just Work will provide more genuine safety.

Chapter 6, “Who’s Looking at You?”: All users, when asked, will say that privacy is important to them. But they won’t cooperate with privacy-enhancing measures, if they require even the tiniest inconvenience. Making them Just Work will lead to more genuine privacy as well.

Chapter 7, “The Generation Gap”: The world’s population is getting older. Good tech can help manage the challenges of aging and improve the lives of seniors. It needs to Just Work even more than for other users.

Chapter 8, “That’s Just Magic”: Some apps are so easy to use that they seem like magic, even among other Just Working apps. This chapter discusses what sets these apps apart.

Chapter 9, “Doing Something About It”: Now that the first eight chapters have convinced you that your tech needs to Just Work, this chapter will show you how to start. I describe five practical procedures for your design process. Follow them to your benefit; violate them at your peril.

Chapter 10, “Epilogue and Benediction”: In which I say goodbye.

And now, dear readers, let’s get on with it.