Pocket Worthy Stories to fuel your mind.

**I’m a Developer. I Won’t Teach My Kids to Code, and Neither Should You.**

**Programming requires much more than the cut and dry language taught in children’s books.**

[Slate](https://slate.com/?utm_source=pocket)

* Joe Morgan

A picture containing table, orange, sitting, bed

Description automatically generated

*Illustration by Natalie Matthews-Ramo*.

On a recent late-night formula run, I passed by a large display of books about teaching children to code. I have seen these books around, but never such a large display directed toward elementary-aged children. These books are part of a flood of resources—summer coding camps, after-school code clubs, apps designed to teach kindergarteners the rudiments of JavaScript—aimed at equipping children with future-proof skills.

It’s easy to see why parents push coding on their children. What better way to prepare our kids for a future ruled by software than by training them how to build it? If everything is going to be automated, it’s much safer to be the one doing the automating. And if learning to code is good, then learning earlier is better. But while these products may teach kids specific coding languages, they actually have very little to do with the work of creating software.

A former co-worker of mine was trained at a coding boot camp with the motto “Coding Is the New Literacy.” That sentiment is at the heart of all the programming books and games. The description in one popular book says starting coding early is “essential to prepare kids for the future.” This gives the impression that not teaching kids to code is somehow equivalent to not teaching them to read.

That is, of course, ridiculous. Coding is not the new literacy. While most parents are literate and know to read to their kids, most are not programmers and have no idea what kind of skills a programmer needs. Coding books for kids present coding as a set of problems with “correct” solutions. And if your children can just master the syntax, they’ll be able to make things quickly and easily. But that is not the way programming works. Programming is messy. Programming is a mix of creativity and determination. Being a developer is about more than syntax, and certain skills can only be taught to the very young.

Early in my career, I wrote some code to configure and run a group of remote servers. The code worked great. At least that’s what I thought until about 18 hours later, when my phone dinged in the middle of the night telling me a group of the servers had failed. Staggering from bed to my laptop, I ran the code again to replace the broken servers. Hours later, a different group failed.

There wasn’t a syntax problem. If there had been, the servers would never have been built in the first place. The problem was much deeper. Isolating and solving it took several weeks and many nights of interrupted sleep.

Coding is like that. Try something. See if it works. Try again. If a problem was straightforward, it would be automated or at least solved with some open-source code. All that’s left is the difficult task of creating something unique. There are no books that teach you how to solve a problem no one has seen before. This is why I don’t want my kids to learn syntax. I want them to learn to solve problems, to dive deep into an issue, to be creative. So how do we teach that?

One day, my son was concerned that a chair of his was wobbly. We looked at it and he helped me isolate the problem: One of the screws was loose. I found one of our many leftover hex wrenches and showed him how to screw it back in. After that, he was curious what would happen if he screwed the other way, which he did until the screw came out. We ended up taking the chair all the way apart and putting it back together a couple of times, often mismatching pieces, before he was satisfied the job was finished. Try something. See how it works. Try again.

Of course, getting something working is just the first step of building software. The next step is to make code clear, reusable, and neat. Once, early in my career, I wrote a feature and gave it to a senior developer for review. He took one look at my sloppy spacing, mismatched lines, and erratic naming conventions and just said, “Do it again.” It was working. The syntax was valid. It was still wrong. Good coders don’t just get something to work. They want it to be good.

That feeling of quality is the hardest thing for many developers to master. Well-designed code feels good to work with, and ugly code will make developers involuntarily cringe. The best developers learn to fuse abstract logic with the sensitivity of an artist. Learning to trust that aesthetic feeling is as much a part of development as any algorithm or coding pattern.

My wife and I recently made sugar cookies with our son. Every time we mixed some ingredients we would pause and look at the dough and talk about the texture and color. Was it smooth? Did we get all the parts mixed evenly? As we rolled out the dough, my son felt the surface and watched as my wife showed him how to get everything even and thin. The hardest part, though, was cutting out the shapes. Like all kids, he instinctively pushed the cutter right in the middle of the rolled-out dough, and every time we would try to explain how to place shapes next to each other in order to maximize each roll.

Every step—precisely measuring ingredients, gauging mixed dough for smoothness and consistency, placing precision cuts to minimize waste—taught him something about quality. It’s hard to teach the difference between merely executing steps, such as following a recipe, and doing something well. It can only be passed on through feel and experience. And every time you involve your kids when you work on something you value, you are teaching them how to do things well. You are preparing them to write code.

But you’re not only teaching them that. You’re teaching them the world is full of interesting things to discover. You’re showing them how to be passionate and look for that ephemeral sense of quality in everything they do. The best part is that even if they don’t become coders—most shouldn’t and won’t—the same skills can be used in nearly any career, in every hobby, in every life. When we force kids to learn syntax, we reinforce the idea that if something is not a blatantly employable skill, it’s not valuable. Adults can learn syntax. Only kids can learn to embrace curiosity.

***Joe Morgan*** *writes a lot of code and is the author of* [*Simplifying JavaScript*](https://www.amazon.com/dp/1680502883/?tag=slatmaga-20)*.*