Published in JavaScript Scene · Follow

Eric Elliott · Follow

Apr 25, 2020 · 6 min read

How to Learn to Code

10 Tips to 10x Your Coding Skills

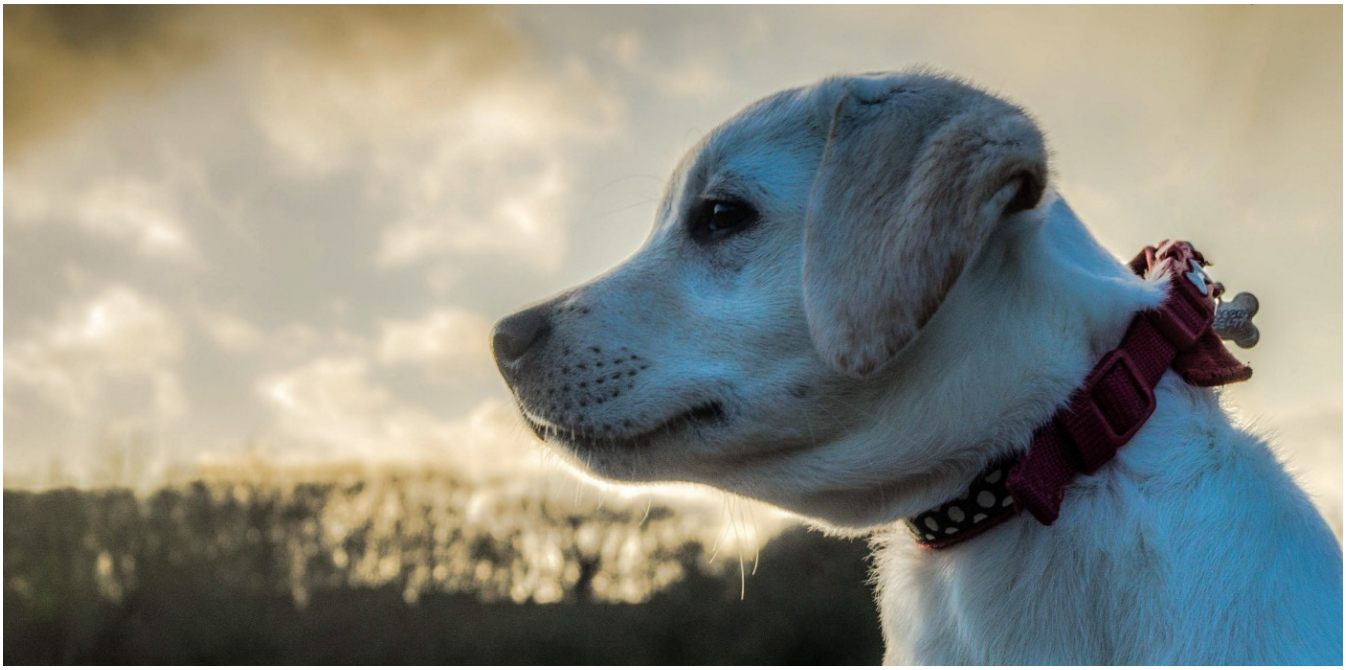

Photo: Phil Dolby — Her First Ever Walk (CC BY 2.0)

Software development is an incredibly rewarding skill that can be extremely valuable. It's remote-work friendly, and no matter where you live in the world, if you get good enough, you can qualify for great paying work ranging from $100k/year — $200k+/year (USD). Some of the highest-paid JavaScript developers make close to $500k/year. But to qualify for those great salaries, you have to get undeniably good at what you do.

Even if you're already a professional software developer, you need to learn how to learn to code. Choosing a career in software development is choosing a path of lifelong learning.

with little or no coding background learn new concepts more than 10 times faster than others who may have 10+ year's experience in the craft. The secret is, you can, too.

There are a handful of learning secrets that can put you on a rocket to mastery of the craft.

1. Code

The best way to learn to code is to code. Jump into a development environment, and write some code. If you're reading a book or blog post and you encounter a code example, type it out in a code editor and try to make it work. Once you get it working, play with it. Change things up. Try to think of other ways to apply it, or other things you can do with the same technique. *Play with the code*.

Book smarts will only get you so far. The best *learning* will come from *doing*.

2. Drive

The best way to get great at something is to do it. A lot. You need to be motivated and determined to learn. One way to get motivated is to give yourself the time and patience to gain some mastery. You don't need to be an expert right away. It's like learning a musical instrument. You can't sit down at a piano and immediately be the next Debussy, but you can master the C major scale in your first sitting.

Likewise, you're not going to sit down and immediately crank out the next Instagram, TikTok or Fortnite.

As you begin to master each small lesson, you'll realize you can do this. You can get good at this. You can start to see your goal begin to materialize, and you'll be more motivated to drive toward that goal.

Keep at it.

3. Focus

I've seen a lot of developers try to master everything all at once and get nowhere, fast. Their progress slows to an excruciatingly glacial crawl rather than a gold medal sprint.

If you want to learn something quickly, you can't have your attention scattered

something else.

I tell people all the time, concentrate on one language full time for at least a year before you branch out and learn another language. Decades ago, it used to be that a typical software developer would actually need to learn many languages in the course of their career to stay competitive in the field.

While it's still true that learning more than one language can teach you different ways of seeing things, and even deepen your understanding of your primary language, these days a single language (JavaScript) can get you through the majority of your career.

Tip from a hiring manager: *The skills you specialize in are your most valuable skills. If you commit to being a lifelong generalist bouncing from language to language, you'll put an artificial ceiling on your mastery and earning potential.*

4. Read

Many of the most useful insights available to software developers come from books. There are lots of good YouTube videos and courses online, but books are the standard bearers of software development culture and knowledge. In particular, I've found the following books extremely valuable:

- Eloquent JavaScript
- Composing Software (Disclaimer: I wrote this one. The printed version is available on Amazon.)
- Code Complete: 2nd Edition
- Clean Code: A Handbook for Agile Software Craftsmanship
- Test Driven Development By Example
- Refactoring: Improving the Design of Existing Code 2nd Edition

5. Review

If you want to move a new concept from a familiar-sounding idea into long-term memory, reviewing a topic is your friend. The mistake most learners make is that they

days in a row, and your chances of committing the learning to long-term memory increase dramatically.

6. Mix Mediums

Some people learn best by reading, others by watching videos, but if you mix it up — watch a video, then do some reading, then practice with some interactive code sessions, you'll repeat the concepts from multiple angles, and multiple examples. You'll naturally drill some review, and get some practice in while you're at it.

7. Build Projects

Learning the concept doesn't mean you'll know how to use it in a real app. Once you've been coding with exercises for a few weeks, it'll be time to build something of your own. Need an idea? Instead of the ubiquitous todo app, try implementing The Rejection App.

8. Value Principles Over Frameworks and Languages

Frameworks and APIs change fast. Software design principles are evergreen. Learn principles that translate across language barriers.

Examples:

- "A small change in requirements should lead to only a small change in implementation." (Paraphrased from "A Practical Handbook for Software Development")
- Do One Thing (DOT) — Simplified from Doug McIlroy's "Do One Thing and Do It Well (DOTADIW)" — a function should have one job. It should not fetch data AND process data AND draw to the screen. It should only fetch data. Or only process data. Or only draw to the screen. (Time to split your React components into smaller parts!)
- "Program to an interface, not an implementation." — Gang of Four, "Design Patterns"
- "Favor object composition over class inheritance." — Gang of Four, "Design Patterns"

  

- You Aren't Gonna Need It (YAGNI) — Don't write code for something that isn't actually required, yet.

9. Share, Document, and Mentor

> *"Dr. Hoenikker used to say that any scientist who couldn't explain to an eight-year-old what he was doing was a charlatan." ~ Kurt Vonnegut — Cat's Cradle*

Learning how to code is just part of the equation. When you're collaborating with other developers, your code will be reviewed by other people, and they will sometimes challenge your choices. As you try to explain yourself, you may find that you didn't understand well enough to defend your position. Practice explaining, documenting, and teaching the concepts to your coworkers and other collaborators on your projects.

10. Practice, practice, practice!

Anybody who's ever learned an acquired skill can attest, practice is key. But to get better you can't just practice the concepts you already know. You need to challenge yourself and extend beyond the realm of what is familiar. If you constantly practice at the edge of your current abilities, you will excel.

The book, "Peak: The New Science of Expertise" delves into the study of deliberate practice and offers a wealth of insights that you can apply in your daily life to get better at practice. I strongly recommend reading it so that you can make your practice time and side-projects more productive.

Next Steps

Ready to compress year's worth of learning into a few months? DevAnywhere.io is a 1:1 software development mentorship program that pairs you live with an expert mentor in weekly video calls where you'll learn by coding while your mentor customizes your learning journey just for you and holds you accountable for your progress.

***Eric Elliott** is a tech product and platform advisor, author of "Composing Software",*

He enjoys a remote lifestyle with the most beautiful woman in the world.