**1. Use Google very aggressively.**You need to know how to phrase search queries, review other developers’ code, and apply it to the problem that you’re trying to solve.

**2. Maintain an obnoxious amount of stick-to-itiveness.**Veteran programmers embrace the experience of being a beginner with new technologies and teach themselves on the fly.

**3. Acknowledge that micro-decisions matter.**For example, things like naming variables, calling functions, naming CSS properties, using a hash vs using an array, and other seemingly small things can have a big impact.

**4. Acknowledge that most major decisions don’t matter that much.**Experienced developers roll with the punches and avoid the flamewars that other developers often find themselves in. They’re quite zen-like.

**5. Always use the right tool for the job.**There are so many different open source libraries, tools, and frameworks out there. Experienced programmers [**know what to use**](http://blog.thefirehoseproject.com/posts/nodejs-vs-rails/?utm_source=quora&utm_medium=quora_ken_answer&utm_content=29_behaviors_unstoppable_programmer&utm_campaign=quora_answers) for each problem that they encounter.

**6. Understand that code is cheap.**You need to be ok withscrapping hundreds of lines of code to do things in a different way.

**7. Evaluate technologies based on all their merits.** For example, I’ve been[bullish on Elixir](https://medium.com/@kenmazaika/why-im-betting-on-elixir-7c8f847b58#.ozz8sow6w). It has wonderful syntax, an amazing community, and a bright future. But it’s so new that if you want to actually build complex features, you’ll have a harder time finding open source technology to make your life easy. You need to take all of these factors into account.

**8. Say “I don’t know.”**There’s no quicker way to waste your time as a developer than to refuse to acknowledge what you don’t know.

**9. Always analyze the clues found in error messages.**Traditional education has taught us that failure is bad. Error messages are often associated with failure. However, good programmers know that these messages are actually clues that lead you down the path to the right solution.

**10. Know the difference between a premature optimization and a show-stopping optimization that NEEDS to happen.** They know the right time to write code that’s a bit more confusing, but will be faster.

**11. Take responsibility for your mistakes.**Mistakes will happen. Especially when working on a team. It’s a waste of time to pass off the blame because typically there are multiple parties at fault.

**12. Become a power-user of your development tools.**If you spend a good percentage of your time coding in a specific environment, you should know exactly how to control it.

**13. Know how to use Vim (at least a little bit).**You need to be capable of at least muddling your way forward using this text editor.

**14. Never accept freelance work in an unfamiliar technology.**A big part of freelancing is estimating how long things will take. Don’t put yourself in an awkward spot by projecting time for something you don’t already know.

**15. Don’t count the number of hours.**Unstoppable programmers spend their time doing deep work (as opposed to shallow work), and they understand that the number of hours spent working [**don’t actually matter.**](http://blog.thefirehoseproject.com/posts/10000-hours-to-learn-to-code/?utm_source=quora&utm_medium=quora_ken_answer&utm_content=29_behaviors_unstoppable_programmer&utm_campaign=quora_answers)

**16. Absorb massive criticism with ease.**You need to develop the ability to react rationally and logically when your code gets torn apart.

**17. Pair program with people with more experience.**There’s no faster way to[**learn to code**](http://www.thefirehoseproject.com/?utm_source=quora&utm_medium=quora_ken_answer&utm_content=29_behaviors_unstoppable_programmer&utm_campaign=quora_answers).

**18. Always code review your own work first.**Before you issue a pull request on Github, you should review the code and pick it apart as if it was written by someone else.

**19. Recognize that the hard part of freelancing isn’t writing the code. It’s everything else.**Sales, marketing, customer support, quality assurance, and product management will all take up a lot of time.

**20. Identify and resolve bigger issues.**The best programmers think beyond the immediate issue at hand and know how to deal with problems in a way that leads to more long-term solutions.

**21. Dive into large open source projects to bring your features to life.**Impossible is nothing if you know how to monkey patch a solution.

**22. Skip a lot of meetings.**Your company pays you to write code. Not to talk about writing code. When meetings get out of hand, it’s ok to start skipping them. People will appreciate your time more if you start doing this.

**23**. **Know when it’s time to give back.**There comes a time when you need to start giving back to junior developers, just like your mentors did with you.

**24. Be capable of writing bad code.**Sometimes, it’s ok to become a [Duct Tape Programmer.](http://www.joelonsoftware.com/items/2009/09/23.html)Over time, you need to figure out when it’s acceptable to take a shortcut and when it is absolutely needed. This is one of the most difficult skills to learn.

**25. Let other people know that you’re working late without being a jerk.**If you’re the last one in the office, just send an email to someone with a quick update. People generally notice the timestamp.

**26. Act as a leader, not a boss.**Bosses are people who have people work for them. Leaders are people who other people follow. Be a leader.

**27. Go play foosball.** In the long run, building relationships with other developers (and people in other roles) will be more valuable than shipping a feature in a tighter window.

**28. Learn under pressure.**You need to know how to navigate a situation in which a system has gone down and you’re responsible for getting it back up. Even if you don’t know exactly how to solve the problem.

**29. “Move fast and**[**break things**](http://startupquote.com/post/1624569753)**.”**Don’t let perfect be the enemy of good. Mistakes are often the best learning opportunities. So don’t treat your mistakes as failures. Instead, treat them as learnable moments and know that fighting through them is critical to your growth as a programmer.