Developer Lite

This information is based on my past 12+ years of experience in Software Development Industry. I have gone through different stages in my career starting from Trainee Software Developer till Program Manager.

I'm not going to dictate any of the points, but all the practices listed here contributed a lot in my software development career, so if you think they make some sense for you then try to adopt few.

What is Practice?

I would say:

- Practice is a habit.
- Practice is a routine.
- Practice does not need to remember.
- Practice comes by practicing.
- Practice needs dedication and commitment.

There are millions of examples which you think about practice. I can list few for your understanding.

There could be a driver but would you assume him an efficient driver if he is driving at a speed of 20 miles per hours and meeting with accidents so frequently and bringing lots of scratches in the car on a daily basis?

Software development is also not different than other skills like shooting, writing or driving. To become a successful software developer, you need lot of practice, dedication and commitment.

Code Reading

Rule 1- Keep Continue Reading Existing Software Source Code

Let me ask you few basic questions before we start with one of the most important best practices required for a software developer.

- Do you read movie magazines?
- Do you read newspapers?
- Do you read roadside advertisements?
- Do you read junk written here and there?
- Do you just read?

Definitely your answer will be positive but if I ask you one more question in the series:

Do you read Software Source Code?

Only few software developers will have positive answer because reading and understanding an existing software source code is the most boring task. If you are one of them who feels reading software source code is a boring task, then you are missing one of the most important best practices, which a software developer should have in his/her life.

So, if you want to write a good software code, then how it will be possible for you to write a good source code without reading tons of source codes? Even if you will write something, then how would you and know which the best is?

Reading source code written by others gives you opportunity to criticize the mistakes done in writing that code. You will be able to identify the mistakes other software developers have done in their source code which you should not repeat.

There are many attributes of software codes (indentation, comments, history header, function structure, etc.), which you will learn by reading existing code, specially, a code written by well-experienced software developers.

Spend some time in reading others' source code and I'm sure you would be able to write BEAUTIFUL source code in few days or few weeks and you will be able to fix the mistakes, which you were doing so far in writing the source code.

Documentation is the Key

Rule 2 - Complete your documents before next step

I had passed out my masters in Computer & Application and I was so passionate to write source code even without completely understanding and documenting the requirements. Design document and test cases documentation were nowhere in the software development life cycle ....there was direct jump to the coding.

At later stages I found myself in big trouble and soon I realized Documentation is the Key to become successful software developer, tester or architect.
Before you start developing small or big software, you should have answer for the following questions:

- Where is the Requirements Specification?
- Where is the Impact Analysis Document?
- Where is the Design Document?
- Have you documented all the assumptions, limitations properly?
- Have you done review of all the documents?
- Did you get sign off on all the documents from all the stakeholders?

What you learn today, prepares you for tomorrow!!!

So, again it is one of the best practices to have documentation as much as possible. Few important documents, which will prepare you for future are:

- Design Approaches
- Tips and Tricks

• Special functions, commands and instructions
• Lessons learnt
• Peculiar situations
• Debugging methods
• Best Practices
• Anything which can help you in future

Keeping documents electronically does not cost you. So let's start maintaining required documentation.

Follow the Standards

Rule 3 - Follow the defined standards, don't create it

Most of the standard software organizations maintain their coding standards. These standards would have been set up by well-experienced software developers after spending years with software development.

This is equivalent to following footsteps of great people left behind them. If your organization does not have any standard, then I would suggest to search on internet for coding standards off different programming languages and you will find many.

A coding standard would fix the rules about various important attributes of the code, few are listed below:

• File Naming convention
• Function & Module Naming convention
• Variable Naming convention
• History, Indentation, Comments
• Readability guidelines
• List of do's and don'ts But once defined, start following the defined standard instead of creating or changing them every day.

Write to be Reviewed

Rule 4 - Code should be written to be reviewed

While writing your software code, keep in mind that someone is going to review your code and you will have to face criticism about one or more of the following points but not limited to:

- Bad coding
- Not following standard
- Not keeping performance in mind
- History, Indentation, Comments are not appropriate.
- Readability is poor
- Open files are not closed
- Allocated memory has not been released
- Too many global variables.
- Too much hard coding.
- Poor error handling.
- No modularity.
- Repeated code.

Keep all the above-mentioned points in your mind while coding and stop them before they jump in your source code. Once you are done with your coding, go for a self-review at least once.

I'm sure, a self-review would help you in removing 95% problems yourself.

Once you are completely done with your coding and self-review, request your peer for a code review.

I would strongly recommend to accept review comments happily and should be thankful to your code reviewers about the comments.

Same time, it is never good to criticize any source code written by someone else. If you never did it, try it once and check the coder's expression. Your target should be to stop the bugs at first place and create a BUG-FREE code. Think like a tester, so that you should have a challenge for the testers.

Testing is the Important

Rule 5 - Testing to be followed like a Important thing

Testing is mandatory after every small or big change no matter how tight schedule you have or you just changed a small comment inside the code, you have testing due for the changed code.

There is nothing like trust while developing software, no matter how expert or how senior you are in writing source code, you would have to perform testing for each and every change you did in the code.

- Tight schedule, no compromise.
- Changed just a comment, still you have to test it.
- Changed just a variable name, testing has to be done.
- If you feel lazy...it's too dangerous.

Keep the Assets Safely

Rule 6 - Keep your Code and Documents Safe

A smart developer keeps habit of taking daily backup of the produced artifacts, otherwise machine crash can crash you as well. You should keep your artifacts at your local machine as well as another secure machine, so that in case of machine crash, you can continue with the saved copy of the source code or documents.

If you have the habit of taking daily backup then in worst scenario you may lose at most one-day effort, but if you take weekly or monthly backup, then there is a risk of losing whole-week or whole-month effort, and you will face biggest disappointment you ever had.

Password sharing is strictly prohibited

• Love, affection, friendship and relationship are on top of everything, but never embrace anybody asking for password.

• If you are sticking to first point, then why you would share your password with anyone if you are not asking from anybody.

• Are you sharing your toothbrush with other? No- then why need share password with friends.

Eager to Learn

Rule 8 -Leave the ego behind, Be eager to learn. Keep Learning

We always learn from books and nowadays from internet. But IT is such a field, where we learn a lot from our colleagues. They are our best references, but there are software developers, who either feel shy in asking their doubts or are not thankful to others, so ultimately when they ask next time, they get zero answer.

IT is vast and nobody can have complete knowledge on any subject. Everyday, we come across different problems. So Ask...Don't feel shy if you don't know X.

I'm not suggesting you to bother someone unreasonably and asking for spoon feeding to learn anything. NO, be polite, thankful, directly come to the point, understand and support others.

New technologies are coming everyday

If you want to sustain in the market, then you would have to keep yourself updated with latest IT tools, and technologies. Following are the few sources:

• Technical Forums over the internet.

• Technical magazines on various IT subjects.

• Technical Bulletin Boards

• Conferences, Trainings and Workshops
• Latest versions of old tools and packages, languages, etc.

Career Planning

Today's professional life is very dynamic and to move along with it we need a proper career planning. When you start your career as a software developer, you really do not know how exactly you will perform in the industry, though you have confidence that whatever you do, will be done in the best way.

So take some time to investigate yourself, what are your major strengths and weaknesses and based on at least 3-4 years of experience you can come up with different options:

• Do you want to continue as software developer forever, which could be a very good option and there are many people, who love coding forever.
• If you are very good in designing software components and your past designs have been appreciated a lot, then you can think to go in technical side and become software architect.
• If you are very good in managing things, have good command over people and have great convincing abilities, then you can think of going towards management role, which will start with leading a small team.
• If you are very good in managing things and at the same time you have great architectural sense, then you can think of becoming techno-manager, where you will keep contributing in designing components and will manage team and projects.

To summarize, it is easy to do just coding but to become a good programmer i.e., software developer needs some hard work and dedication in doing lot of practice. There could be a list of thousands of best practices, which can be listed down by veteran software developers but let us eat the quantity, which we can digest easily.

Just keep your list small but follow them strictly throughout your developer's life.

Share it with others…

If you have any feedback/comments, kindly feel free to write me back: rkitonlinetrainiings@gmail.com