# Developer’s Best Practices Tutorial

# DEVELOPER'S BEST PRACTICES TUTORIAL

*Simply Easy Learning by tutorialspoint.com*

tutorialspoint.com

## ABOUT THE TUTORIAL

# Developer's Best Practices Tutorial

This small tutorial is based on my past 16+ years of experience in software development industry. I have gone through different stages in my career starting from trainee software developer till senior management.

I do not want to keep my learnings with myself so I had written a small tutorial few years ago and after getting lot of motivation from my dear readers, I thought of revising it and adding few more learnings which may benefit many other software engineers and developers working in this lovely industry.

I'm not going to dictate any of the points, but all the practices listed here contributed a lot in my software developer career, so if you think they make some sense for you then try to adopt few. If you have any +/- comments, kindly feel free to write me back. Contact Us

## Audience

If you are working for software industry as a software engineer or a software developer then I'm sure you are going to enjoy this tutorial. Try to relate the facts mentioned in the tutorial, with your day-2-day life and find so many hidden facts which are very obvious but we never gave our serious attention to them.

## Prerequisites

Before writing all the practices mentioned in this small tutorial, I have made an assumption that you are working as a software professional and you understand basic software terminologies and atmosphere around a software professional.

## Copyright & Disclaimer Notice

# Table of Content

CHAPTER

1

# What is Practice?

When I'm saying "Practice", what does it mean? 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 thousands of examples which you think about practice. I can list few for your understanding

Shooting, Driving, Writing

Any of the above listed skills comes from practice. When, initially, you start driving, you need to remember each step and you think twice before taking any action, but once you "have good practice" of driving then you do not need to remember any step. It becomes your habit and routine for example, your feet goes automatically at brake if you see a red light but definitely it comes from practicing a lot and needs a lot of dedication and commitment.

One of the most important attributes of practice is that it **forces you not to divert** from what you used to do.

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 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.

Through this small article, I'm going to tell you few major best software developer's practices which you may find useful. So let's start....

CHAPTER

2

# Code Reading and Reading

## Best Practice 1- Keep 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 road side 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.

If you want to become a novelist, can you just start writing novels? I would say 100% no!!, you definitely need to read hundreds of novels before you start writing **GOOD** novels. If you want to become a movie script writer, can you start writing good movie script until you have gone through various good movies scripts? again my answer would be no!!

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 developer. 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.

One thing to experiment, just go in past and check the code you had written few years ago, you will definitely laugh....because you are always improving by doing practice.

CHAPTER

3

# Documentation is the Key

## Best Practice 2 - Complete your documents before next step

I had passed out my master 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 stake holders?

Once you have positive answers for all the above questions, you are safe and ready to proceed for the coding. Many organizations would have strict rules to be followed, but others would not have. Best practice is to complete all the required documentation and take appropriate approvals before proceeding for the software coding.

## 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.

CHAPTER

4

# Follow the Standards

## Best Practice 3 - Follow the defined standards, don't create it

Most of the standard software organizations maintain their coding standards. These standards would have been setup 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 standard of 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. I would definitely say:

**Source code is your BABY!**

So keep it clean, consistent, and beautiful. When I say beautiful, it really means beautiful. If your code looks beautiful then it would be easy for others to read and understand it. If you will keep changing coding rules everyday then after few days you, yourself would not be able to read and understand the code written by you.

CHAPTER

5

# Write to be Reviewed

## Best Practice 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:

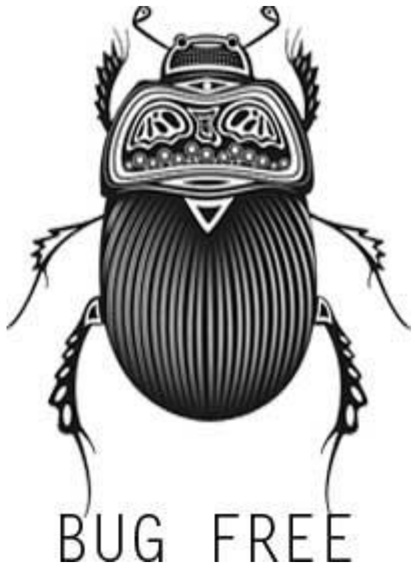
- Bad coding
- Not following standard
- Not keeping performance in mind
- History, Indentation, Comments are not appropriate.
- Readability is poor
- Open files are not closed
- Allocated memory has not been released
- Too many global variables.
- Too much hard coding.
- Poor error handling.
- No modularity.
- 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 90% 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.

## Accept criticism but don't criticize

Poorly written source code teaches you to write good source code provided you take it positively and learn a lesson from it.



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.

CHAPTER

6

# Testing is the Religion

## Best Practice 5 - Testing to be followed like a religion

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

If you don't want to follow it? You will be in trouble!

## Celebrate every bug you find

Yes you should not feel unhappy if you or another tester finds a bug in your software source code. Following are the enough reasons to celebrate this important discovery:

- Bugs are your enemies, so you have killed one.
- Now your software is having one bug less.
- Mistakes are good as long as they are not repeating.
- What you learn today prepares you for tomorrow.

Same time do not criticize any developer in case any bug arises in his/her code because so far at least I do not know any programmer who can write bug free source code in the world, second this is one of the reasons we have a separate phase in SDLC (Software Development Life Cycle) which we call post production support (or support & maintenance).

CHAPTER

7

# Keep the Assets Safe

## Best Practice 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.

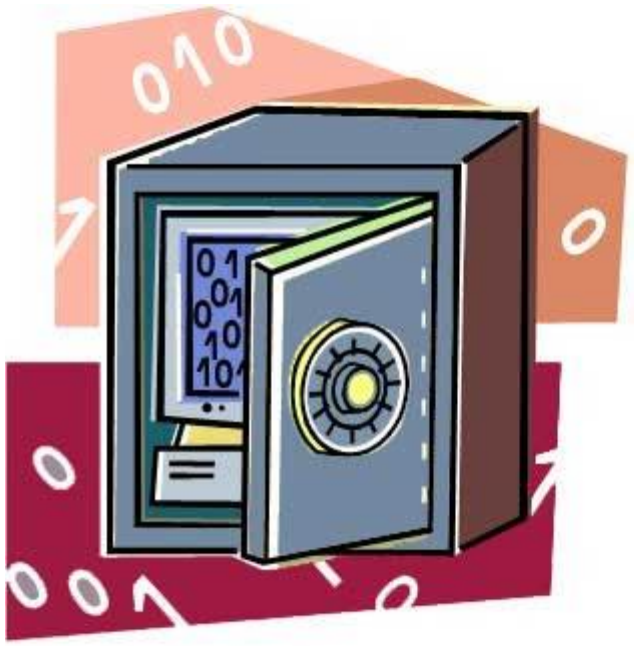
## Multiple copies create confusion

This is true that having backup is one of the most important best practices but it should be maintained in well managed way as you can use tags like name, date and time of the backup, version etc. If you have multiple copies of the same source code or document, then it will create confusion and it would be difficult to identify latest code or document.

It is strongly recommended to use proper source code version control system. There are many source code version control software applications available for free (like SCCS, CVS, Subversion etc.) which you can use to store different versions of the software. But while using a source code control system, follow the rules below:

- Always take source code from the version control system.
- Always assign a new version to every change.
- Always put source code back into control system.

## 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.
- Keep changing it on frequent basis and it’s good if you have some logic to drive your passwords, otherwise during your long vacation, you will forget them.

CHAPTER

8

# Handy Tools and Techniques

## Best Practice 7 - Keep your Tools & Techniques Handy

I remember an instance when I wanted to find out **debug** keyword in all the C++ files available in various directories and sub-directories. It took me 30 minutes to find the command, but finally I kept a note of the command and whenever I'm in need, I use it without wasting a second.

```
$find. -name \*.cpp -exec grep -q "debug" '{}' \; -print
```

So I made it one of the best practices to keep such commands and tools handy so that they can be used anytime without doing any R&D and to save valuable time. Better to maintain a text file having all such frequently used commands and create its link at desktop.

## Few Essential Tools

It depends on what type of programming you are doing but following are few of the essential tools which should be readily available with a software developer:

- A good text editor to write and edit the program.

- A nice debugger to debug the program.
- A memory detector in case you are using dynamic memory allocation.
- Putty to connect to a remote machine.
- WinSCP or FileZilla to ftp files on a remote machine.
- IDE (Integrated Development Environment) for rapid development.

## Always keep adding new tools & techniques in your box

Make sure you keep applying latest patches of your tools and utilities and same time I will suggest to clean unwanted software from your computer as they unnecessarily make your computer slow and you never know if any one of them is having a security hole which can expose your computer to the outside world.

CHAPTER

9

# Eager to Learn

## Best Practice 8 -Leave the ego behind, Be eager to learn

We always learn from books and now-a-days 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. Every day 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.

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.

CHAPTER

10

# Stress Management

As you grow at your position, your responsibilities increase in multiples of your salary increment which definitely bring lots of stress in your personal and professional life. As such there is no formula to get rid of your stress and you will find fat books and training programs to teach you how to manage stress but I believe an open communication is the biggest weapon which can help you up to some extent to relieve yourself from big stress.

## Let's identify root cause of the stress

You are a software professional so you should know how to debug a problem. Similar way stress is a problem for you and you have to debug it. You should find out why it’s coming to you and what its root causes are. Let's take few examples which may be the cause of stress in your day-2-day life:

- Work load is too much and you are not able to handle it properly.
- You had been assigned to a module which is not ready though deadline has arrived.
- So far you really do not know what exactly you have to do and how exactly you have to do?
- You had developed a code which got deleted by mistake or not working at final moment.
- You are leader of the team, but team is not doing so great and ultimately delivery is getting delayed.
- Though you have enough time to deliver, but meanwhile you planned for a travel as well which may cause delay in your delivery.

## Communication, Communication.....& Just Communication

For you none of them should be a problem if you take them in professional way. Let's pickup any of the above mentioned point for example, first point where you feel overloaded and not able to finish your task within office hours.

Simply setup a small meeting with your manager and put the facts in front of him, mentioning your current assignments, bottlenecks and reasons why you feel you are overloaded. You can request him to share one more resource with you or to give you more time. I'm sure your manager will listen on this and will help you if he needs a good delivery from you. You need to plan how you are going to convince your manager about it and make him realize that what you are saying is correct.

Similar way any of the mentioned issues can be resolved with proper communication with your manager and if it's not working with manager then many organizations give you chance to talk to your higher management and take your issues to them. So in case your problem does not get resolved, you take it to higher level, but you need to be careful because it can be a little sensitive as none of the managers would like you to by pass him and talk to his boss directly. But yes it could be your last try if nothing is working.

One more important issue is poor prioritization of the work. If you can discuss priority of the work with your manager then you can handle and schedule all the tasks one after another. You can give some extra time after office hours or during weekend to relieve yourself.

## Personal vs Professional

Try to identify whether you are not able to work properly because you have some personal issues and they are impacting your professional life. In such case your family is the best one who can help you in resolving your personal issues. You can share your personal issues with your close friends or family, spouse etc. and get them fixed as soon as possible. If it is going very serious then it’s better to talk to your manager and explain him the situation and try to get few days off and then fix your personal issues and come back to catch up with your work.

## Stress could be momentarily

Aha, it’s part of everybody's life and you should not get stressed due to little overload, little delayed delivery or some minor issues happening around you. Let's make them part of your day-2-day life and keep moving on. So let's do a little more extra overtime to finish your delivery, take little help from your friends, and be ready to listen few comments from your manager.

Make sure you are not repeating problems, and problems are also not repeating with you and if this is the case then it’s time to take action and find out its solution.

## Few more quick remedies

Try to use any of the following if they relieve you from stress:

- Some exercise
- Little or more yoga, meditation
- Morning walk
- Evening movie

- Pass some time with your friends, family, spouse, kids
- Avoid sitting for long time and have a coffee break at work, read magazine, newspapers, internet browsing, using stress removal toys

Bottom line is that you should not keep quiet and keep creating a volcano, which will erupt someday later and produce lots of damages. Be communicative, be transparent and be honest. Keep in mind, if you are under stress then your productivity will reduce unexpectedly, so try to keep yourself healthy, happy and active…..

CHAPTER

11

# Managing Managers

As a software developer i.e. programmer, one of the most challenging issues you face is related to managing your manager and his/her expectations. You may come across various complicated and confusing situations which are unexpected and difficult to resolve and ultimately you become a victim of unnecessary stress we discussed in last chapter. Following examples may be few of them:

- Your manager does not give you due respect and value.
- One of your peers does not deliver still he is always in news and getting appreciation notes.
- There has been some misunderstanding between you and your manager.
- A cold war is running between your and your manager.
- From last few years your manager did not think about your promotion or salary revision.
- You think your manager is not capable enough and it is difficult to convince him/her.
- It does not matter what you deliver, still you have to get negative feedback.
- Your manager does not like you because of XYZ reasons.

Just think what is going on between you and your manager, I am sure you will be able to add your issue in the above list. That's the first and most important task to identify why there is an issue. It could be X... Y... or Z....

## Managers are always correct....

Yes, if you are disagreeing with me then it's obvious why you are in trouble. Try to recall when you were a kid, and your parents always stopped you from doing X...Y...or ...Z activities and they used to emphasize on certain things which you never liked in your childhood. But now will definitely say, Alas! it would have been so good for us if we would have done the way parents instructed. Now if you are inline with me then it means you found out half of the solution of your problems.

So, crux of the discussion is the given attention what your manager is asking for and do the way he suggested. Your ultimate goal should be to make your manager happy and few of the points can help you in achieving this:

- Try to give fast deliveries, it does not matter if you put your effort during weekend.

- Reduce your complains about things around you.
- Reduce your demands in terms of salary revisions or promotions.
- Do not lose a chance to present your work to your manager, does not matter it’s small or big but your manager should be aware of what you do.
- Be neutral as much as possible, do not criticize any other peer in front of the manager.
- Take things positive done or presented by your manager, as I said they are always right.
- You will have to observe why your manager likes any particular resource and try to inline with that resource.
- Never try to think your manager is inferior to you, that may be the case but it’s not allowed to think like that, otherwise by doing so you create problem for yourself.

## Managers always need great resources

Great, so you have adopted all the points mentioned above, now you will say I will give fast and clean deliveries by putting my honest efforts during weekends and holidays, still I should not demand for salary hikes or promotion, why????

My answer is yes, you do it and things will come automatically, just have patience. You will hardly need to demand for anything once you make your manager realize that you are one of the brightest resources and you are most important for the project. Once you achieve this, your manager would never like to lose you, and now it’s your time to enjoy your work and working environment.

If still you find things are not moving as per expectation, then you have to initiate a healthy discussion with your manager and ask for the reasons why you are not getting hikes and promotions. It could be some other HR related issues or project budget etc. You can ask for improvement areas if needed and set expectations accordingly, but again your cycle will start from the above mentioned activities.

If you have some misunderstandings with your manager then call for a meeting with the manager and accept the mistakes you have done if any and clarify the things which went wrong and give an assurance to take care of such incidents in future.

Many things depend on situation and you need to be smart enough to understand the situation and act accordingly. All the very best.

CHAPTER

12

# 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 will 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.

Whatever it is, you must be aware of where do you want to reach. Once you are sure about this, you should start working in the same direction starting from your project preference till your trainings and certifications. Your current organization may not be giving you appropriate opportunity to reach your desired destination then you can wait for right time and make a move to other good organization but it should not be very frequent. I have seen guys doing monkey jump every six months from one organization to another one just because of little hike and it’s being done without a proper thinking and proper planning but these fellows do not know what they are losing in long run.

You can discuss about your career path with your manager/line manager and most of the organizations have standard career path defined for their employees, so you can check if it suits your interests and work accordingly.

## When to make a move?

This is very interesting question that when I should move to another organization, but I cannot answer it in simple words. You know your career path and if your current organization is enough to put you at your final destination then why do you want to leave it. Leaving an organization just because of few bucks is never a good reason, even leaving organizations too frequently is not a good idea though you are getting great position and big hikes, this is simply because you are losing your credibility and none of the good companies will rely on you because you are always behind money and position so who knows when you will leave them.

If you have some internal HR, or Management issues within your organization then try to resolve them because you never know your next organization may have even bigger issues than your current organization. You can discuss your issues with your manager, director or with HR and resolve them gracefully.

If you see no further growth and good career options in the current organization and same time your learning curve got a saturation then it’s time to make a shift to another organization. There may be a situation when you are not getting a fat salary and having great position in your current organization but you are learning a lot, which will add a lot of value in your resume and your career, then better to stick with the current organization until your learning is over.

CHAPTER

13

# Summary

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 of your developer's life.

Tomorrow your kids are going to use it...

I'm sure, tomorrow same tutorial will be used by your kids if luckily they are software developers i.e. programmers or engineers, so let us improve it all together. If you like this tutorial then share it with others, and write me back about the improvement: Feedback