97 things every programmer should know

Chapter 1**. Act with Prudence**

**Three Things I learned today**

1. **Before** - I usually create and write codes without minding if it’s clean and is up to the standard as long as it works without thinking about the consequences in order for me to finish it as soon as possible to meet up with the deadlines.

**After** - I learned about the technical debt and what it can do to the overall projects if I let it accumulate as I continue with writing codes in a quick way and not in the right way. This can cause multiple problems in the long run and will eventually create a problematic issue within the project that would take time to solve.

1. **Before** - When I write codes I don’t always put on comments on the codes that I wrote and when I did its only from time to time and that is to track the errors that I found and which codes I need to come back later to fix and try to clean them up but I never do because of the deadlines, other projects or it gets forgotten.

**After** - I realized that I would need to document those lines of codes and write it in a task card or log it in an issue tracking system so I don’t forget to fix those lines of codes and I can come back to it as soon as possible so it won’t accumulate more.

1. **Before** - I never thought about the consequences of writing a code quickly instead of cleanly as soon as it gets the job done and I can submit my works, I always think that it’s better to submit a loosely written code that’s working and on time rather than nothing at all.

**After** - I learned that it’s not the right mindset for a programmer and that over time it becomes a bad habit, and that it can get progressively worse over time as you progress and can bring problematic consequences in the long run.

97 things every programmer should know

Chapter 2**. Apply Functional Programming Principles**

**Three Things I learned today**

1. **Before** - When writing the codes I don’t tend to follow the set standards due to having the habit of writing them as quickly as possible without implementing quality codes.

**After** - I know now that I should learn and practice the mastery of functional programming paradigm to greatly improve the quality of my code and lessen the issues and bugs that I would need to fix.

1. **Before** - The function calls I usually writes in my codes only serves one purpose and with the same variable returning the same value and can’t be reused, this results in me creating multiple function calls for different functions that typically has the same output.

**After** - I found and learned about referential property and that function calls can have its parameters value replaced which enables it to produces a different result and output without changing the programs behavior.

1. **Before** - I usually write multiple function which causes multiple errors and defects that needs to be debug later on and id have trouble locating those error and which functions those value occurs.

**After** - I learned about the net result design which has better responsibility of allocating those multiple functions rather than referencing mutable member variables which results in fewer defects and easier to debug.

97 things every programmer should know

Chapter 2**. Ask “What Would the User Do?” (You are not the User)**

**Three Things I learned today**

1. **Before** - When I design a front end website I usually don’t consider what the user experience will be like and create designs based on to my own liking and what’s more easier for me.

**After** - I learned that it’s actually a biased to think in such a way, and that users don’t think in a way that programmers so when designing websites pages I should put into consideration how the users responds.

1. **Before** - When I do consider the front end designs on how the users would interact I just imagine myself as the user not thinking without realizing that the way I imagine myself as a user is not actually in line with how the users would react.

**After** - Users would focus on the problem they can’t solve which is different from how programmer does it, they don’t look for any other options and would get stuck on the same problem so it’s better to ask the users about their own experience.

1. **Before** - In order to consider the users behavior and gain their focus on things would be usually just changed the color of certain objects on the design in order to gain eyes and focus their attention where I want them to.

**After** - Users don’t typically follows what you expect or what you want them to do so it’s much better to provide them a much obvious ways of doing things on how operate your website and gives them tips on how it works.