**97 things every programmer should know**

**▼ Title Of Book and Chapter**

**Chapter 1 Act of with Prudence**

**▼ What are Three Things I learned today**

**1.** **If you find yourself having to choose between "doing it right" and "doing it quick"**

Before – Doing it quick is a good choice.

After - In this line I can totally agree that it is more appealing is "to do it quick". But if I were to choose between two I choose "do it right", why? Is it because why would I do it quick. Then after I need to come back because, I need to fix something. So choose a right decision even if it takes time for you to do it, but you do it right and you can proceed to what you need to do next, without worrying the outcome of your first project. And you will not encounter any problems.

**2. Deferred work is known as technical debt and it is not your friend**

Before – I don’t know this technical debt before

After - Yes, this is truly not your friend it is because you just benefited it in the short term, but you need to pay it with interest. But if you commit something to the company that you’re working, then you need to do what you must need to do in order for you to fulfill it and not reached this kind of debt.

**3. Pay off technical debt as soon as possible**

Before – I don’t have any idea of this before.

After - If you reached this stage you need to pay in order for you not to make things complicated.

**▼ Title Of Book and Chapter**

**Chapter 2 Apply Functional Programming Principles**

**▼ What are Three Things I learned today**

**1. Mastery of the functional programming paradigm**

Before – I don’t know this Mastery of functional programming paradigm

After - If you understand and know the functions of these then it is easy for you to code and understand.

**2. Referential transparency is a very desirable property**

Before – I don’t have any idea about this.

After - So referential transparency is a fancy term that is attached to a computable expression.

**3. Master the functional programming paradigm**

Before – I also don’t know about this

After - So in this stage you are able to judiciously apply the lessons learned to domains.

**▼ Title Of Book and Chapter**

**Chapter 3 Ask "What Would the User Do?" (You Are not the User)**

**▼ What are Three Things I learned today**

**1. Users do a core of things similarly**

Before – Users do the things similarly specially to those users who are not reading instructions.

After - So the users try to complete tasks in the same order and they make the same mistakes in the same places. As a programmer or a UI designer you need to design around that core behavior. So the programmer questions themselves "What if the user wants to...?". So the developer need to do is to make a design that is suitable for their target age or target users. For example, we the Synergy created a mobile-based simulation game. So we need to create a UI that is suitable for the users who are we targeting, in order for them to easily access the game app and it is easy for them.

**2. You'll see users getting stuck**

Before - Yes a lot of this happened if you make an application without an instruction

After - So if I am a user and getting stuck I need to look for an instruction in order for me to continue. And if I am a developer I need to put some intructions or tools that can help them to avoid getting stuck.

**3. Users tend to muddle through**

Before – User feel puzzled if they don’t familiarised what’s in your application.

After - So for me as a game developer we need to ask some user to play our game application so that we need to know what's the user want or if they have suggestion to put something in the game.