97 Things every programmer should know

**Chapter 4 Automate Your Coding Standard**

Before-

When I write a code, it didn’t matter to me about, commenting, the naming (as long as it connects to what my functions work on) spacing, and more, as long as the code runs what I wanted it to be.

After-

Reading the standard way of coding, makes me think of my past ways of coding, if I did these things before my coding should be all in proper way by now, if I learn how to code clean it would help me understand my errors more easily and also it helps me fix or debug easily because of the comments in my code, I should practice the proper standard way of coding.

**Chapter 5 Beauty Is in Simplicity**

Before-

I do read about this, but my code is still messy, but for me its readable and understandable, I don’t know about others who read about my it.

After-

The chapter mentioned that I should seek some source code from well known, acknowledge expert.  
and see some forum in codepen.io, who has so much expert code and how well they did it, its simple and understandable so the viewer of his/her code will get to know easily.

**Chapter 6 Before You Refactor**

Before-

I do have some experience in making changes in internal code, by adding some features or changing some little detail about specific function because it got slow or unused.

After-

Reading this chapter, give me a lot of idea, and how crucial refactor can be, it can be in good way, or worse because of a wrong decision making, make time for thinking before doing something, is the key I learn about this chapter, don’t make haste, take it slowly a step at a time will get you to your desire progress.

**Chapter 7 Beware the Share**

Before-

I never do some code reusing in my times, or I have but I cant remember it, for me code reusing is good thing because it can make less time calling out same functions but have other output, or making multiple task and it can be good if one function can do it all at same time.

After-

As I understand in the context, don’t jump into conclusion that you used to be, not all things you do in the school, are same with the industry/workplace, as the speaker said, “These mistakes are insidious in that, at their core, they sound like a good idea. When applied in the right context, these techniques are valuable. In the wrong context, they increase cost rather than value.” So check it carefully and make a thousands of testing before doing so.

**Chapter 8 The Boy Scout Rule**

Before-

I always seek help of my friend for the things that I cant understand or get to know what was wrong in my code, and I want to seek how if a team can do much better because they will really help each other to solve one problem how big or small it is.

After-

The rule of a boy scout, for me this is the best practice to be a good coder, helping each other and follow the rule to improve for everyone not for just yourself.