97 Things Every Programmer Should Know

Chapter 61-70

Chapter 61 : "Own (and Refactor) the Build"

This chapter talks about how important it is for developers to take charge of the process that puts their code together into a working program. By doing this, they can make sure that the program is built correctly every time, making it easier to fix problems and add new features. Refactoring means cleaning up and improving the build process over time, so it stays efficient and reliable.

Chapter 62: "Pair Program and Feel the Flow"

Pair programming means two programmers working together on the same task at the same time. This chapter explains how this can be really helpful. It helps catch mistakes early because one person can spot what the other might miss. It also means problems get solved faster because two heads are better than one. Plus, it makes work more enjoyable because you have someone to bounce ideas off of and share the workload with.

Chapter 63: "Prefer Domain-Specific Types to Primitive Types"

When you're writing code, it's better to use words or labels that are specific to the problem you're solving, rather than just using basic numbers or letters. For example, if you're writing a program about cars, it's better to use terms like "car" or "speed" instead of just using generic terms like "int" or "float". This makes the code easier to understand for other people who might read it later, and it can help prevent mistakes because the code is more closely related to the real-world problem you're trying to solve.

Chapter 64: "Prevent Errors"

This chapter emphasizes the importance of being careful and thorough when writing code to avoid mistakes. It suggests techniques like double-checking your work, testing everything thoroughly, and designing user interfaces that guide users away from making errors. By paying attention to detail and thinking about ways to prevent mistakes before they happen, developers can save time and frustration down the road.

Chapter 65: "The Professional Programmer"

Being a professional programmer means more than just writing code. It means taking your work seriously, always trying to learn and improve, and treating others with respect. Professionals write clean, well-organized code, communicate effectively with their team members, and take responsibility for their work. By adopting a professional mindset, programmers can build trust with their colleagues and produce high-quality software that meets the needs of users.

Chapter 66: "Put Everything Under Version Control"

This chapter emphasizes the importance of using version control systems (like Git or Subversion) to manage changes to software code and other project files. By placing everything under version control, developers can track changes, collaborate effectively with team members, and revert to previous versions if needed. It promotes good practices such as committing changes frequently, writing meaningful commit messages, and branching for feature development or experimentation.

Chapter 67: "Put the Mouse Down and Step Away from the Keyboard"

This chapter encourages developers to take breaks and step away from their computers to refresh their minds and avoid burnout. It highlights the importance of balancing intense coding sessions with periods of rest, physical activity, and social interaction. By taking breaks, developers can maintain their focus, creativity, and overall well-being, leading to better productivity and enjoyment of their work.

Chapter 68 : "Read Code"

Reading and understanding code written by others is a valuable skill for developers. This chapter advocates for actively engaging with codebases outside of one's own projects to learn new techniques, gain insights into different programming styles, and broaden one's

understanding of software development practices. By reading code, developers can identify patterns, learn from others' mistakes, and improve their own coding skills.

Chapter 69: "Read the Humanities"

This chapter emphasizes the importance of broadening one's intellectual horizons beyond technical subjects by reading literature, philosophy, history, and other humanities disciplines. It argues that exposure to diverse perspectives and ideas from the humanities can enhance creativity, critical thinking, and empathy, which are valuable qualities for software developers. By drawing inspiration from the humanities, developers can bring new insights and perspectives to their work, leading to more innovative and meaningful solutions.

Chapter 70 : "Reinvent the Wheel Often" by Jason P. Sage:

Contrary to the common adage "don't reinvent the wheel," this chapter encourages developers to reinvent solutions to common problems as a way to deepen their understanding and mastery of programming concepts. It suggests that reimplementing existing solutions from scratch can be a valuable learning experience, allowing developers to explore different approaches, experiment with new techniques, and gain insights into the underlying principles of software development.