97 Things Every Programmer Should Know

– COLLECTIVE WISDOM FROM THE EXPERTS

Chapter 21-30

Chapter 21: Do Lots of Deliberate Practice

Deliberate practice isn't just about completing tasks; it's about improving skills through focused repetition. Mastery takes around 10,000 hours of deliberate practice, challenging yourself beyond current abilities and learning from mistakes. It's about choice and dedication, not just innate talent. Deliberate practice involves challenging yourself with tasks just beyond your current ability, analyzing your performance, and correcting mistakes. It's about learning and evolving, even if it's not always enjoyable.

Chapter 22: Don’t Be Afraid to Break Things

DSLs are like deciphering secret codes across different domains, offering a deeper understanding and connection with experts. As a developer, mastering DSLs isn't just about technical prowess; it's about unlocking the language of specialized communities and broadening my skill set. It's like deciphering the hidden language of a specialized community, where mastery brings not just proficiency, but a sense of belonging to that unique realm of expertise.

Chapter 23: Don’t Be Cute with Your Test Data

Using playful or inappropriate test data might seem harmless, but it can lead to embarrassing consequences if it becomes visible, potentially harming your team or company's reputation. Whether it's quirky user names or humorous error messages, these seemingly innocent choices can backfire spectacularly, potentially harming your team or company's reputation. In today's interconnected world, even the smallest slip-up can quickly spread online, causing significant fallout before it can be rectified.

Chapter 24: Don’t Ignore that, Error!

Ignoring errors in code is like ignoring physical pain, it might seem minor, but it can lead to serious consequences. Just as I regretted not addressing my leg pain immediately, programmers often regret ignoring errors in their code. Checking for and handling errors is crucial for maintaining solid and secure code, despite common excuses like cluttering code flow or looming deadlines.

Chapter 25: Don’t Just Learn the Language, Understand Its Culture

Learning a programming language is exploring a new culture. It's not just about memorizing words (syntax); it's about understanding how things are done (culture). By immersing yourself in the culture of each language, you learn new ways to solve problems and become a better programmer overall. Enabling us to tackle problems in novel and elegant ways across different languages.

Chapter 26: Don’t Nail Your Program into the Upright Position

In handling errors in C++ is to avoid trying too hard to keep the program going when things go wrong. If I tried a fancy approach, but it just made things worse - programs crashed without telling us why. Instead, it's better to use simple methods to report errors so we can fix them properly. It was a tough lesson to learn, but it's important to remember to avoid making the same mistake again.

Chapter 27 : Don’t Rely on “Magic Happens Here”

Simplifying tasks by relying on what seems like "magic" can sometimes lead to trouble when things go wrong. Understanding at least some of the complexities involved and appreciating those who do can help prevent issues. And it's crucial to ensure you could fix things if the "magic" unexpectedly stops.

Chapter 28: Don’t Repeat Yourself

When you're writing code, try not to repeat yourself. It's called the "Don't Repeat Yourself" (DRY) rule. Repeating the same code over and over makes things messy and can cause errors. Instead, look for ways to automate repetitive tasks. And if you notice you're writing similar code patterns, find ways to make it simpler and clearer. Sometimes, you might need to repeat code for specific reasons, but in general, following the DRY rule keeps your code neat and easy to understand.

Chapter 29: Don’t Touch That Code

After putting code in the version control system, it goes to the development server for testing. Developers stop making changes there. The staging manager takes charge, moving the code to the testing server for QA. Nobody should mess with the development server except the release manager, who deals with both development and live servers. Developers should never mess with live servers; any problems are fixed by support staff or by using updates from version control. This rule helps prevent big problems.

Chapter 30 : Encapsulate Behavior, Not Just State

Encapsulation in coding is like organizing your stuff into neat folders. It keeps things tidy and makes it easier to handle complex tasks. For example, a door can be open, closed, opening, or closing. Similarly, in coding, objects should behave differently based on their status. 3 classes: Customer, Order, and Item. A Customer has credit limit info, and an Order knows about its associated Customer. When adding an item to an order, the order checks the customer's credit limit. If it's too high, the purchase is canceled. Some programmers might try to put all these rules into one big mess of code, but it's better to keep things organized and separate, just like we do with our stuff. So, remember, neat and tidy code is easier to manage and understand!