Summary Report

Art of Readable Code

Chapter 05

Knowing what to comment

- ***The purpose of commenting is to help the reader know as much as the writer did.***

What not to comment

- ***Don't comment on facts that can be derived quickly from the code itself.***
- But for most programmers, reading the commented code is much faster than understanding the code without it.
- Don't Comment Just for the Sake of Commenting
- Don't Comment Bad Names—Fix the Names Instead
    - A good name is better than a good comment because it will be seen everywhere the function is used.
- In general, you don't want "crutch comments"—comments that are trying to make up for the unreadability of the code. Coders often state this rule as good code > bad code + good comments.

Recording Your Thoughts

- **Include "Director Commentary"**
    - You should include comments to record valuable insights about the code.
- **Comment the Flaws in Your Code**
    - Code is constantly evolving and is bound to have flaws along the way. Don't be embarrassed to document those flaws.

Marker	**Typical meaning**
TODO:	Stuff I haven't gotten around to yet
FIXME:	Known-broken code here
HACK:	Admittedly inelegant solution to a problem
XXX:	Danger, major problem

    - The important thing is that you should always feel free to comment on your thoughts about how the code should change in the future. Comments like these give readers valuable insight into the quality and state of the code and might even give them some direction on how to improve it.
- **Comment on Your Constants**
    - most constants could be improved by adding a comment. It's just a matter of jotting down what you were thinking about when you decided on that constant's value.

Put Yourself in the Reader's Shoes

- A general technique we use in this book is to imagine what your code looks like to an outsider—someone who isn't as intimately familiar with your project as you are. This technique is especially useful to help you recognize what needs commenting.
- **Anticipating Likely Questions**
    - To address the users, or other developers question.
- **Advertising Likely Pitfalls**

Summary

What not to comment:

• Facts that can be quickly derived from the code itself.

• "Crutch comments" that make up for bad code (such as a bad function name)—fix the code instead.

Thoughts you should be recording include:

• Insights about why code is one way and not another ("director commentary").

• Flaws in your code, by using markers like TODO: or XXX:.

• The "story" for how a constant got its value.

Put yourself in the reader's shoes:

• Anticipate which parts of your code will make readers say "Huh?" and comment those.

• Document any surprising behavior an average reader wouldn't expect.

• Use "big picture" comments at the file/class level to explain how all the pieces fit together.

• Summarize blocks of code with comments so that the reader doesn't get lost in the details.