More comments in code or just simple, readable, maintainable code suffices? [closed]

Asked 15 years, 10 months ago Modified 15 years, 10 months ago

Viewed 1k times

12

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 12 years ago.

Sometimes its really difficult to decide on when exactly you have written enough comments for someone to understand your intentions.

I think one needs to just focus more on writing readable, easy to understand code than on including a large number of lines of comments explaining every detail of whats happening.

What are your views about this?

comments

Share

Improve this question

Follow

asked Feb 13, 2009 at 13:34

Lonzo

2,818 ● 4 ● 25 ● 27

13 Answers

Sorted by: Highest score (default)

31

Comments aren't there to explain what you're doing. They're there to explain why you're doing it.

Share Improve this answer

Follow

answered Feb 13, 2009 at 13:37

Paul Tomblin

183k ● 59 ● 323 ● 410

Not entirely true (e.g. when you're writing tricky code, like bit twiddling hacks) but +1 regardless. – strager Feb 13, 2009 at 13:39

1 Anyone remember everybody's first assembler program, where you'd have a comment on every line that basically echoed the instruction? "TAX ; Transfer the Accumulator to X" – Paul Tomblin Feb 13, 2009 at 13:41

Lol, yup because assembler was hard and we had to comment everyting ;-). – Toon Krijthe Feb 13, 2009 at 13:42

Btw, TAX sounds like a 6502 instruction. – Toon Krijthe Feb 13, 2009 at 13:43

@Gamecat - you got it exactly. Started on the Pet 4032, got a KIM-1 when I wanted to do some deep 6502 machine code (hand assemble, load the bytes through the keypad).
– Paul Tomblin Feb 13, 2009 at 13:49

12

The argument is based on a false dilemma: Either your code is a horrible abomination and you write tons of comments to explain every statement and expression, or your code is beautiful poetry that can be understood by your grandmother with no documentation at all.

In reality, you should strive for the latter (well, maybe not your grandmother but other developers), but realize that there are times when a couple of comments will clear up an ambiguity or make the next ten lines of code so much more plain. People who advocate *no comments at all* are extremists.

Of course, gratuitous comments should be avoided. No amount of comments will help bad code be more understandable. They probably just make it worse. But unless you're only coding trivial systems, there will be times when comments will clarify the design decisions being made.

This can be helpful when catching bugs. Literate code can look perfectly legitimate while being completely wrong. Without the comments, others (or you six months later) have to guess about your intent: Did you mean to do that, or was it an accident? Is this the bug, or is it somewhere else? Maybe I should refer to the design

documentation... Comments are inline documentation, visible right where you need it.

Properly deciding when the need for comments actually exists is the key.

Share Improve this answer

Follow

answered Feb 13, 2009 at 13:51

Adam Bellaire
110k ● 19 ● 152 ● 165

> Agreed. Whenever I see someone saying that only bad code contains comments, I think that this is obviously someone who has never written code for a large, complex, production system. – Graeme Perrow Feb 13, 2009 at 22:07

5

Try to make the code self-explaining. One of the most important things is to use meaningful names for classes, functions, variables etc.

Comment the sections that aren't self-explaining. Trivial commenting (e.g. i++; // Add 1 to i) makes the code harder to read.

By the way - the closer to pseudocode you can work, the more self-explaining your code can become. This is a privilege of high-level languages; it's hard to make self-explaining assembly code.

Share Improve this answer

Follow

edited Feb 13, 2009 at 13:48

answered Feb 13, 2009 at 13:41

Joonas Pulakka

36.6k ● 29 ● 108 ● 171

2

Not all code is self-documenting.

I'm in the process of troubleshooting a performance issue now. The developer thought he discovered the source of the bottleneck; a block of code that was going to sleep for some reason. There were no comments around this code, no context as to **why** it was there. We removed the block and re-tested. Now, the app is failing under load where it wasn't before.

My guess is someone had previously run into a performance issue and put this code in to mitigate the problem. Whether or not that was the right solution is one thing, but a few comments about **why** this code is there would now be saving us a world of pain and a whole lot of time...

Share Improve this answer

Follow

answered Feb 13, 2009 at 13:56

Patrick Cuff

29.7k ● 12 ● 69 ● 94

Why you need comments. The name of the method should be clear enough that you don't need comments.

Ex:

```
//  This method is used to retrieve information
about contact
public getContact()
{
}
```

In this case getContact doesn't need the comments

Share Improve this answer

Follow

edited Feb 13, 2009 at 14:41

Jonathan Leffler

752k ● 145 ● 946 ● 1.3k

answered Feb 13, 2009 at 13:54

bcorriveau

55 ● 6

Aim for code that needs no comments, but don't beat yourself up too much if you miss.

Share Improve this answer

Follow

answered Feb 13, 2009 at 14:55

Mike Woodhouse

52.3k ● 12 ● 89 ● 127

0

I think commenting enough so that you could understand it if you had to review your code later in life should be sufficient.

I think there would a lot of time wasted if you commented for everyone; and going this route could make your code even harder to understand.

I agree that writing readable code is probably the most important part, but don't leave out comments. Take the extra time.

Share Improve this answer

Follow

answered Feb 13, 2009 at 13:39

JoshFinnie

4,911 ● 7 ● 30 ● 29

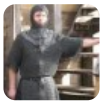
0

Readable code should be the number 1 priority. Comments are, as Paul Tomblin already wrote, to focus on the why part.

Share Improve this answer

Follow

answered Feb 13, 2009 at 13:40



Toon Krijthe

53.4k ● 38 ● 149 ● 202

0

I try to avoid commenting as much as possible. Code should be self explanatory. Name variables and methods properly. Break large code blocks in methods which have

a good name. Write methods that do one thing, the thing you named them for.

If you need to write a comment. Make it short. I often have the feeling that if you need to elaborate long on why this code block does this and that you already have a problem with the design.

Share Improve this answer

Follow

answered Feb 13, 2009 at 13:46

raupach

3,102 ● 23 ● 30

In my experience, "code should be self explanatory" is too idealistic and just doesn't scale to large, complex, multi-developer systems. In the real world, sometimes the best-named variables and functions still can't tell you WHY the code does what it does. – Graeme Perrow Feb 13, 2009 at 22:13

I totally agree. But there are quite a number of developers who don't even try to write self explanatory code. – raupach Feb 14, 2009 at 17:56

0

Only comment when it adds something.

Something like this is useless and definitely **decreases** readability:

```
/// <summary>Handles the "event" event</summary>
/// <param name="sender">Event sender</param>
/// <param name="e">Event arguments</param>
protected void Event_Handler (object sender,
EventArgs e)
```

```
{
}
```

Share Improve this answer

Follow

edited Feb 13, 2009 at 14:44

Jonathan Leffler

752k ● 145 ● 946 ● 1.3k

answered Feb 13, 2009 at 14:01

User

30.9k ● 22 ● 81 ● 108

0

Basically, putting aside a good but possibly brief comment at the beginning of a class/method/function declaration, and - if necessary - an introductory comment at the beginning of the file, a comment would be useful when a not-so-common or not-so-clearly-transparent operation is coded.

So, for example, you should avoid commenting what's obvious (i++; on a previous example), but what you know is less obvious and/or more tricky should deserve some clear, unconfusing, brilliant, complete line of comment, which naturally comes along with a Nobel prize for the clearest code in history ;).

And don't underestimate the fact that a comment should be also funny; programmers read much more gladly if you can intellectually tease them.

So, as a general principle tend to not be overwhelming with comments, but when you have to write one, be sure about it to be the clearest comment you could write down.

And personally I'm not a big fan of self-documenting code (a.k.a. code w/o a single damn slashstar): after months you've written it (it's just days for my personal scale) it's very likely you couldn't tell the true reason for choosing such design to represent that piece of your intelligence, so how could others?

Comments are not just that green stuff among code lines; they are the part of code which your brain is better willing to compile. Qualifying as braincode (laughing) I couldn't affirm comments are not part of the program you're writing. They're just the part of it which is not directed to the CPU.

Share Improve this answer Follow

answered Feb 13, 2009 at 16:08

{fz.name} Federico Zancan
4,884 ● 4 ● 48 ● 62

0

Normally, I'm a fan of documentation comments that clearly spell out the intent of the code you're writing. Spiffy tools like NDoc and Sandcastle provide a nice, consistent way in which to write that documentation.

However, I've noticed a few things over the years.

- Most documentation comments don't really tell me anything I can't really glean from the code. That assumes, of course, that I can make heads or tails out of the source code to begin with.

- Comments are supposed to be used to document *intent*, not behavior. Unfortunately, in the vast majority of cases, this isn't how they're used. Tools like NDoc and Sandcastle only propagate the incorrect use of comments by providing a plethora of tags that encourage you to provide comments that tell the reader things that he should be able to discern from the code itself.
- Over time, the comments tend to fall out of synch with the code. This tends to be true regardless of whether or not we're using documentation software, which purports to make documentation easier because it puts the documentation closer to the code it describes. Even though the documentation is *right there next to the method, property, event, class, or other type,* developers still have a hard time remembering to update it if and when the intrinsic behavior changes. Consequently, the documentation loses its value.

It's worth noting that these problems are, by and large, due to the misuse of comments. If comments are used solely as a means of conveying intent, these issues go the way of the dodo, since the intent of any given type or its members is unlikely to change over time. (If it does, a better plan is to write a new member and deprecate the old one with a reference to the new one.)

Comments can have immense value if they are used properly. But that means knowing what they are best used for, and constraining their use to that scope. If you

fail to do that, what you end up with is a plethora of comments that are incorrect, misleading, and a source of busywork (at increased cost) since you now have to either remove them or somehow get them corrected.

It's worth it to have a strategy for using comments in a meaningful way that prevents them from becoming a time, energy, and money sink.

Share Improve this answer

Follow

answered Feb 13, 2009 at 21:25

Mike Hofer

17k ● 11 ● 78 ● 113

0

Studies have stated that optimal readability happens when you have about 1 line of comments for 10 lines of code. Of course, that's not to say that you need to keep your ration at 1/10 and panic if you go over. But it's a good way to give you an idea of how much you should be commenting.

Also remember that comments are a code smell. That is to say that they may be indicative of bad code but aren't necessarily so. The reason for this is that code that is more difficult to understand is commented more.

Share Improve this answer

Follow

answered Feb 13, 2009 at 21:53

Jason Baker

198k ● 138 ● 382 ● 520