Lucas Becker

[CS-210-T4249 Programming Languages 21EW4](https://learn.snhu.edu/d2l/home/687811)

Professor Eric Grigori

4/25/2021

8-2 Assignment: Making Standards Actionable

My list:

1. Contribute to society
2. Do no harm
3. Respect privacy
4. Emphasize security
5. Prevent harm
6. Always be learning
7. Build for the future
8. Follow workplace style guidelines
9. Be honest and realistic
10. Maintain objectivity

I put the highest ideals at the top of the list and put more specific items towards the bottom. Contribute to society is number 1 because I feel it encapsulates the main idea we should all be striving towards as software developers. Our code should always be built for a positive and beneficial purpose. It should always make life better or easier for people in some way.

Do no harm is another high ideal. We should always ensure that our projects aren’t causing damage in the course of making their contribution, and of course we should not take work that will be used for the purpose of causing harm.

Respecting privacy is another ideal which complements the first two. Ensuring that all private data is handled with care and only used for legitimate purposes is a big part doing no harm. It’s a decent litmus test for whether your project is contributing to society or not. Farming personal data to sell to the highest bidder is likely not going to be considered ethical by many.

Emphasizing security builds off the idea of respecting privacy. We may not be using the data to breach privacy, but someone out there probably wants to. It’s part of our duty as developers to ensure that our code can’t be exploited and the data we handle can’t be breached.

Prevent harm is a similar idea to doing no harm, but it’s different in one important aspect. Simply not causing harm yourself isn’t enough to be ethical. We must ensure that we blow the whistle on harmful and illegal code in order to actively prevent harm.

Always be learning is something I’d put higher on the list if I could, but it didn’t feel right to place it above some of the higher ideals on the list. Continuing to learn new skills and improve our knowledge of existing ones is critical to our own job security and future viability as developers. Beyond that, it’s also important to be familiar with the latest technology and best practices so you can effectively enact the top 5.

Build for the future is simply the idea that code should be robust, reusable, and extensible. This will help to ensure your code can have as long of a life as possible. Future programmers should be able to extend your code, add to it, inherit from it, etc, without breaking everything.

Following workplace style guidelines is just to ensure internal consistency with how things are done. It’s got to be difficult to deal with a large and complex code base, and the only thing that could make it more difficult is inconsistent naming conventions and ideas about what exactly constitutes “best practices.” Even if your opinion, you should stick to what your co-workers use.

Being honest and realistic doesn’t apply solely to software development, but it’s still very important to being a successful dev. Nothing is worse than underestimating the length of time something will take in order to please people, only to end up with severe crunch time and maybe a late delivery of the finished product. Instead, be straightforward about your skill level and what’s realistically possible in the established time frame.

Maintaining objectivity is something I try to aim for in my day-to-day life, but I feel that it’s also very applicable to software development. Sometimes it’s tempting to try out some new trick you learned or try to implement some complex pattern that you’ve read improves performance, but in most contexts readability and maintainability are usually the priority. Stick to what’s going to work best for your project and your co-workers.