Coding Standards

Forey, Truman

Bellevue University, CSD 430

CSD 430 Module 11 Paper

7/26/2025

REPLACE TITLE

On a surface level, coding can appear to be a simple practice. People with less experience in the subject may see a fragment of code and assume that’s the only or the best way that that task can be completed. However, as projects develop in size and complexity and developers become more skilled, it quickly becomes obvious that there’s a hundred slightly different ways that any singular task in programming can be done. As such, it’s important for any team to develop a coding standard that everyone involved adheres to. While these standards don’t have to be different from team to team, they often share common factors that can be applied to almost all circumstances.

But first, it’s important to ask, ‘why do we need a unified coding standard, exactly?’ While it’s intuitive that having one will be helpful in some way, what are the exact benefits of doing so? One of the primary benefits is that it makes collaboration on a large project much easier. When everyone involved follows the same guidelines for how code should be written and formatted, it makes it easier for others to read and edit that code when they need to make adjustments. This also makes code easier to maintain, since any code created will be in a format that you know will work. In addition, this makes catching simple errors much easier since any deviations from the norm will be immediately obvious and can be corrected. (GeeksforGeeks) Finally, following a unified coding standard makes reusing code much simpler; if code is created for one part of a program, it’s likely to be easily adaptable to a different part.

So what are some standards that every developer should follow? For one, the names of all variables, functions, and objects should be clear, meaningful, and helpful. Instead of a variable being called something ambiguous and non-descriptive like “x,” it should instead be called something that describes what purpose it fulfills, such as “x\_coordinate” if you’re using the variable as a literal X value in geometry or “function\_first\_value” if you’re using x as a generic mathematics variable. In addition, it’s also common practice to change the formatting of the text based on what is being described. General variables are often written in ‘snake\_case,’ functions in ‘camelCase,’ and objects in ‘PascalCase.’ (BrowserStack) Doing this makes it simple to understand the nature of the code without any extra comments. That being said, in-code comments should be used frequently and heavily. No matter how good external documentation may be, it’s unlikely to be as immediately helpful as having direct records of what does what right in the code.

One of the more important directives that a coding standard should establish is how to create efficient, modular code. Generally, this is done by focusing on creating small functions – each function should be designed to accomplish a singular and able to accomplish the same task even if they’re used in different parts of the project. Functions that need to be large in some way should be broken down into as many smaller functions as possible, as this helps with the debugging process and the smaller functions created in this process may become individually helpful at a later point. And finally, global variables should always be avoided, as much as possible. Global variables reduce modularity by being accessible and changeable across the entire project and they always occupy a space in memory, even when they’re not being used – the exact opposite of reusable, efficient code.

But more than any single piece of advice, the most critical part of developing a coding standard is consistency. While there will always be an argument over what is the absolute best way to code, the reality is that many methods are either equally efficient or have negligible upsides and downsides to each other. As such, it’s less important to find some perfect standard and instead to define a standard that is functional and helpful to developers, and stick to it. (Patel, 2024) For one example, it’s open to debate as to whether open brackets ( { ) should be on a new line or at the end of a line that calls for them. In contrast, it’s practically universal that closed brackets ( } ) should always be on a new line. Both of the former options are valid choices to implement for your organization’s coding standards, while the latter is a practice something that every developer should follow.

# References

*Coding Standards and Best Practices to Follow*. (2024). BrowserStack. Retrieved at <https://www.browserstack.com/guide/coding-standards-best-practices>

*Coding Standards and Guidelines*. (2024). GeeksforGeeks. Retrieved at <https://www.geeksforgeeks.org/software-engineering/coding-standards-and-guidelines/>

Patel, R. (2024). *Understanding Coding Standards: Why They Matter*. Medium. Retrieved at <https://medium.com/@ravipatel.it/understanding-coding-standards-why-they-matter-cc75adcba245>