## **Clean Code**

### **Writing Clean Code**

Developers spend SO MUCH more time reading code than writing it. This is true even with your own code. As a favor to yourself and whoever will need to use, maintain, or further develop your code, please learn to write readable code.

### **Learning Outcomes**

By the end of this lesson, you should be able to do the following:

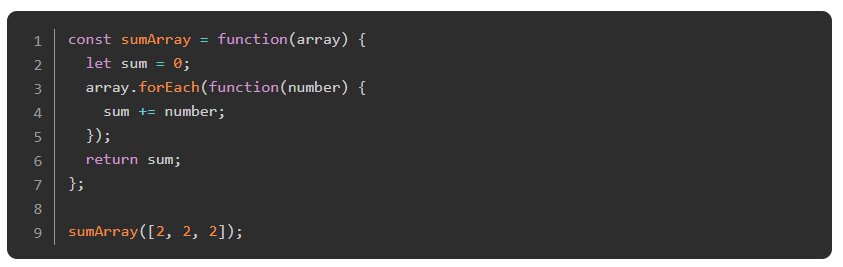
* Know how to distinguish hard to read code between easy to read code.
* Use programming principles to make your code cleaner.
* Write good comments.

Consider the following 2 snippets of JavaScript:

ugly, hard to read code:



clean and easy to read code:



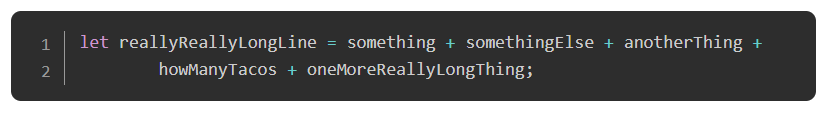
Believe it or not, both of those functions do the exact same thing (in the exact same way!), and both of them are perfectly valid code, but obviously the second one is much easier to follow. Imagine you’re working on a project with someone else and they’ve written the first function… How long is it going to take you to figure out what’s going on there so you can do your work? Imagine you’re working on a project all by yourself and YOU wrote the first function a week or two ago… chances are good that you aren’t going to remember exactly what you were up to there and it’s *still* going to take you a good while to figure it all out again.

The second one, however, is much easier to follow. Even if you don’t know exactly what everything in the code is doing, things are named clearly enough that you could guess, and the indentation is consistent enough that it’s easy to parse the different parts of the function.

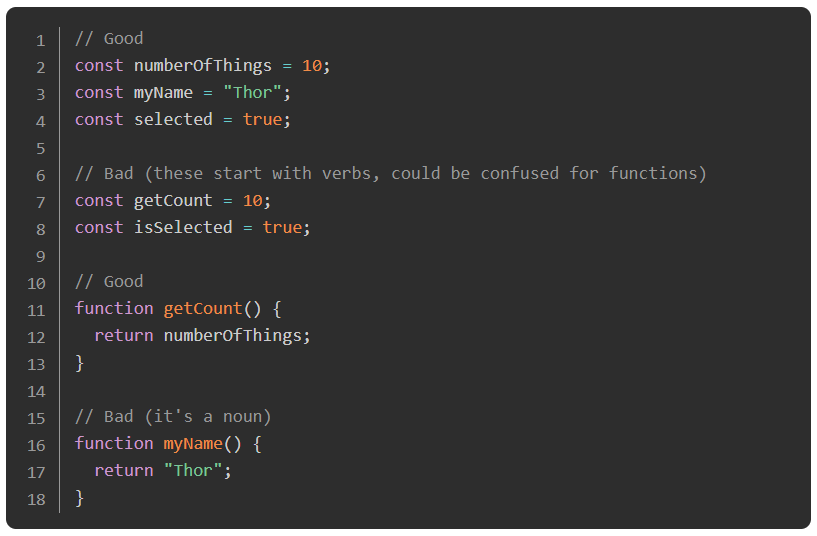
There are many different opinions on what constitutes great JavaScript code. The most important thing is just that you’re consistent. The war between coders that use tabs and coders that use spaces to indent their code is so ingrained that [it’s essentially a joke by now](https://www.youtube.com/watch?v=SsoOG6ZeyUI), but it doesn’t *really* matter as long as you’re consistent.

### **Rules of Thumb**

1. Indentation: It doesn’t *really* matter what style of indentation you use. Various JS style-guides recommend different options, and one is not really superior to the other. What *is* important, however, is consistency. In our examples we will use 2 spaces for indentation.
2. Semicolons: Semicolons are *mostly* optional in JavaScript because the JS compiler will automatically insert them if they are omitted. This functionality CAN break in certain situations leading to bugs in your code so it is better to get used to adding semicolons. Just do it!
3. Line length: Again, different style guides will recommend different options for this one, but just about ALL of them suggest limiting the length of each line of code. This rule is not quite as strict as some of the others, but as a general rule, your code will be easier to read if you manually break lines that are longer than about 80 characters. Many code editors have a line in the display to show when you have crossed this threshold. When manually breaking lines, you should indent the second line 2 levels, and should try to break immediately *after* an operator or comma:



1. Naming Things: Names for functions and variables should be descriptive. Always use camelCase. To keep things consistent and easy to read, variables should always begin with a noun or an adjective (that is, a noun phrase) and functions with a verb. It is ok to use single characters as variable names in the context of a loop or a callback function, but not elsewhere.



### **Assignment**

Read through these articles that discuss a few elements of writing good clean code.

1. [This list of clean-code tips](https://onextrapixel.com/10-principles-for-keeping-your-programming-code-clean/).
2. [This article](https://blog.codinghorror.com/coding-without-comments/), [and this one too](https://blog.codinghorror.com/code-tells-you-how-comments-tell-you-why/) about the role of comments in your code.

### **Additional Resources**

This section contains helpful links to other content. It isn’t required, so consider it supplemental.

* [A nice op-ed](https://www.martinfowler.com/bliki/CodeAsDocumentation.html)
* THE complete guide to [self-documenting code](http://wiki.c2.com/?SelfDocumentingCode)
* [Airbnb style guide](https://github.com/airbnb/javascript)
* [Chaining methods to write sentences](https://web.archive.org/web/20190211152543/https://javascriptissexy.com/beautiful-javascript-easily-create-chainable-cascading-methods-for-expressiveness/)

### **Knowledge check**

This section contains questions for you to check your understanding of this lesson. If you’re having trouble answering the questions below on your own, review the material above to find the answer.

* [Why is it important to write clean code?](https://www.theodinproject.com/paths/foundations/courses/foundations/lessons/clean-code#writing-clean-code)
* [Name 5 clean code principles previously mentioned](https://onextrapixel.com/10-principles-for-keeping-your-programming-code-clean/)
* [What is the difference between good comments and bad comments?](https://onextrapixel.com/10-principles-for-keeping-your-programming-code-clean/)