Good code should be able to scale with you.

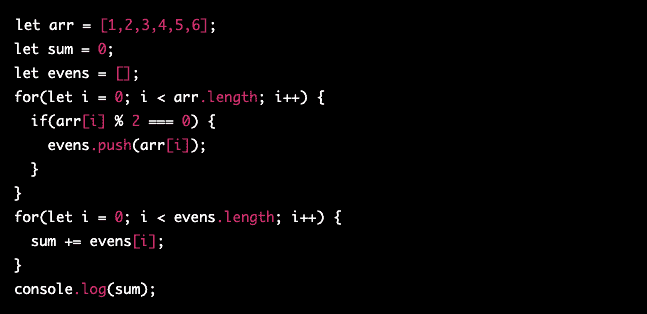
If you’re building software to handle a specific problem, but as other details such as your data set change, your code slows significantly, then you have bad code. But if your software is able to solve the problem at a similar speed, no matter how much your data changes, then you have written some good code.

Scalable code is code that has been written in such a way that it doesn’t get easily overwhelmed. It’s efficient and solves problems or performs tasks as quickly and as frequently as they pop up.

When building a website with long-term goals, for example,  a web development firm should get together with its front- and back-end developers, UX team, client, and other stakeholders, to outline a long-term roadmap for the site. This can help coders future-proof their code and create beautiful, clean code that can stand up to whatever is thrown at it.

The above code defines a function sumOfEvens that takes an array of numbers and returns the sum of all even numbers in the array. The function is well-structured with a clear name and variable names that indicate their purpose. It uses a for loop to iterate through the array and add even numbers to a sum variable. The function is scalable as it can handle an array of any length, and the code is commented to explain what each line is doing.

**Bad code is unnecessarily complex**



Let’s say you’re designing a webpage. You might have different elements of your page that you want to look a certain way. Maybe you want every other paragraph of text to be a certain color. Let’s say blue.

You build out your page, write some CSS code that makes every other paragraph of text blue, and you hit the publish button. A week later, your boss wants to change that blue text to green for some reason.

When you go back into your code, are you changing one line of code or multiple lines of code? If you have to change one line of code, then your code is probably pretty good. If you have to change the code on each and every individual paragraph with blue text, then your code is probably unnecessarily complicated.

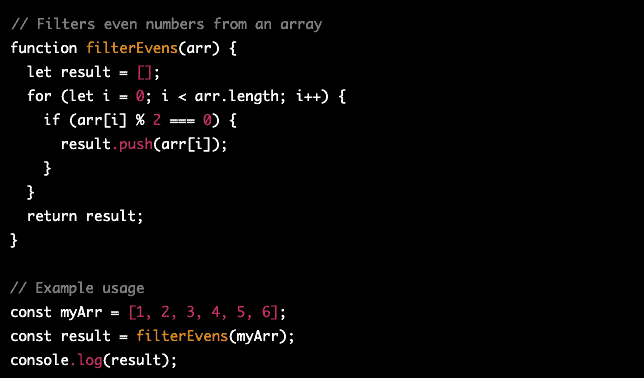
This example seems unlikely at first, because the program is so simple. But over time, as the codebase builds up, sometimes with multiple coders working together or in succession, it can get cluttered and unruly.

The result, often called “spaghetti code,” lacks structure, cohesion, and becomes difficult and costly to update or maintain.

In a real-world example of bad code, a web development firm assuming control of a client’s website from a previous firm might encounter spaghetti code and have to make a decision. Depending on the client’s business goals, it might be better to rebuild the site from scratch at increased cost to the firm or the client in order to meet their goals, rather than try to “fix” the spaghetti code.

The above code achieves the same result as the previous “scalable” example, but the code is unnecessarily complex and hard to follow. It uses multiple variables and loops to calculate the sum of even numbers in an array. The code is also not well-structured and lacks comments, making it difficult to understand. This kind of code is often referred to as “spaghetti” code because it is tangled and hard to disentangle. It is difficult to maintain and modify, and it is not scalable to handle larger arrays.

**Bad code doesn’t have logical functions**



A function is a self-contained module of code that exists to accomplish a specific task.

What does that mean?

Let’s say you built your own word processing software. You’d probably have a spell check feature somewhere in your code. That way you could write in your word processor and get automatically flagged every time you misspelled something. That spell-checking feature is called a function.

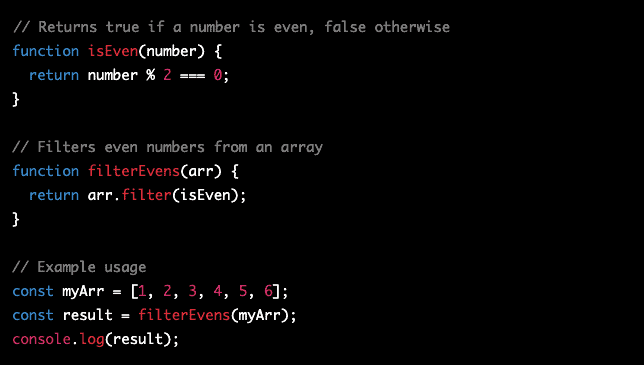
Now if your function was called “spell\_checker” that would make sense. That tells other programmers exactly what that function is and does anytime it pops up in your code.

But if it was called “cherry\_soda” that wouldn’t tell the programmers anything about the function.

Simply put, that’s bad code.

The above code does not use logical functions to simplify the code. Instead, it uses a for loop to iterate through the array and check if each number is even. This approach makes the code more difficult to read and maintain, especially for larger and more complex problems. Without logical functions, the code is more prone to bugs and is less scalable.

Alternatively, the code below achieves the same result as above, but with logical functions.



This code defines two functions, isEven and filterEvens. isEven is a logical function that returns true if a number is even and false otherwise. filterEvens uses isEven to filter even numbers from an array. The code is well-structured, and the functions are named descriptively to indicate their purpose. The code is also easy to understand as the logical functions make the code more readable and maintainable.

**Where Can I Find Good Code?**

So where can you even find good code? A good place to start is a Git repository. Git, meaning, free, open-source software that tracks changes like in a Google or Word document through a coding system called “version control.” (Read: [Git vs. Github: What’s the Difference?](https://devmountain.com/blog/git-vs-github-whats-the-difference/)) There are a number of git repositories you can start in, as well as blogs, chat forums, and social media and other resources, all of which may be helpful for first-time users of Git repositories.

Here are a couple of our favorite places to find good code.

**GitHub**

GitHub is probably the most popular software development and version control hosting company in the world. Millions of developers use GitHub to work together on a variety of different coding projects day after day. A lot of these projects are public-facing, meaning you can actually see the code. There are even open-sourced projects where you could actually have the opportunity to contribute to the code.

If you’re looking to find example after example of good code, GitHub should be your first choice.

**BitBucket**

Bitbucket is another Git-based repository like GitHub. While it’s not as big as GitHub, BitBucket is still used across the world by different commercial enterprises. This also means that it has a lot of public-facing commercial code to look at and make sense of.

**StackOverflow**

StackOverflow is one of the most popular question-and-answer websites for both professional and amateur programmers. If you’re wondering “Why doesn’t my code work” or “Why does my compiler keep giving me ‘Error X,’” chances are really good StackOverflow will be one of the first search results that’ll pop up if you look for an answer on Google.

**StackExchange**

StackExchange is a StackOverflow property and it’s another great resource for looking for examples of good code as teams of users who are working together to make their current code even better. If you’re going to get into coding in any serious fashion, you should probably just have StackOverflow and StackExchange accounts just in case.

You can also look to sites like Quora, Reddit, and chat services such as Discord to find numerous coding communities. And if you’re a web developer, you could always just look at the source code of your favorite websites. There may be some restrictions to what you can and can’t see, but you’ll be able to see a lot of the front-end elements with a simple right-click.

And don’t ignore Google. As you learn coding languages, your vocabulary will grow and those keywords can be very effective for unearthing blogs and forums where others have preceded you on your journey.

Quotes around keywords will match exact phrases and the minus sign will exclude results if, say, you want to read about the Django web framework but don’t want to see results about the movie Django, Unchained (Django -movie).

Google is a programmer’s best friend.

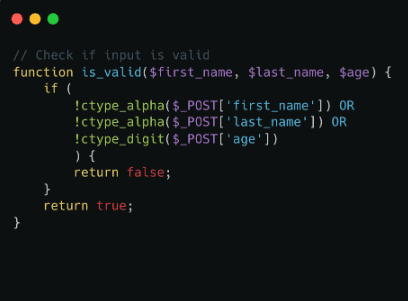
**How Can I Contribute To Good Code?**

So let’s say you’ve been reading a bunch of code, and you’ve been leveling up your own coding skills. You may now feel like it’s time to start contributing to one of the open-sourced projects you’ve been looking at.

But where do you start?

Below are a couple of tips to help you get started when you feel like you’re ready to contribute.

**Offer to write comments**



There’s a lot of code that could stand to be made better. One of the biggest impacts you can have on a coding project is adding comments to the text so that it’s easier for contributors to parse through the code and know what does what. It’s also a way to get access to a project’s code when no one really knows who you are.

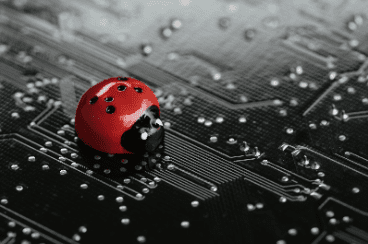
Adding comments as a volunteer will help you level up your ability to understand code, while adding real value to the project, without risking breaking the project. Good for everyone.

**Write tests**



Code is only as good as it has been tested. Writing tests to find out where a project could stand to be improved is a great way to provide real value on an open-source project while also learning more about good code itself.

**Fix bugs**



If you’re feeling really brave, and you’ve established a portfolio enough to get access to the project, you can work on fixing the bugs in the program’s code. Everyone loves someone who can find and fix problems so they don’t have to worry about them.