**How to improve your code readability? Useful clean code ideas to know**

* In many ways, working in software industry is like working in industries such as construction or interior design. It requires knowledge and precision. When you work on a giant project with many features, sometimes it feels like analyzing DNA strings on inventing a vaccine for the worst threats to human health. The slightest **mistake can result in numerous errors** and ultimately cause a project to be extended or put on hold. That’s why working with code may be associated with individualism, and cause the programmer to be perceived as a lonely scientist who spends a lot of time solving intricate programming problems alone. Meanwhile, working on code is largely a heavily collaborative effort between many engineers. The completion is possible only if each one of them understands every part of the project. **So the easier to read and understand the code, the better**.

## Programmers spend a large portion of their time **reading code that has been written by someone else**. To make the right interpretation of such a code, they need to make sure they understand what the author wanted to convey. Usually, it’s much harder if what you made is not readable. That's why at Accesto we **stick to CLEAN CODE**

## Writing clean code - 7 useful ideas for readable code

### 1. Think about other people in your team

Yes, that's true. While coding, you shall not only think about the code, but **should also think about the others**. When we work alone on a project, we may feel that the most important thing is to make the code understandable only to us. After all, who else might sit down and need to use what we wrote. Remember, however, that at any time there may be a situation in which someone else will need to look at our code and make changes there. Even if you're a one-man Army programmer at first, that can always change, and it's worth accepting that in some time someone else may gain access to your code. Therefore, it is a good idea to **write code with the assumption that someone else WILL use it**. Even it you think than noone will ever do it. Create it in such a way that all the rules related to Clean Code are kept. To make sure that our code is readable by other people, we can also show what we created to other people and ask them to check its readability. Remember that we are not infallible, and other people can point out what we have missed.

### 2.Make reusable code and avoid copy-pasting

Is writing the same code several times over the course of one project a good idea? No, the best way is to write the function once and then use it several times in the project. This simply saves time and also makes other people see some repetitive pattern in it, which of course makes it easier for them to understand what the author wanted to create. Although it may be tempting to just copy & and paste the code, this only increases the <a href="<https://accesto.com/blog/technical-debt-the-silent-villain-of-web-development/> "target="\_blank">technical debt and converst your code info maintenance nightmare.

### 3. Leave your code a bit better than you found it

Or in other words - be a good Boy Scout. As Boy Scouts try to leave the campground cleaner than they found it. And you should think the same about the code. Always try to transform bad code you see, into something better. But watch out! As our senior dev Michal mentioned in his great arricle on [Boy Scout rule in PHP](https://accesto.com/blog/Boy-scout-rule-in-6-examples-the-basic-principle-of-web-development/), you should focus on the campground, not the entire forest! Over-refacotring is a common anti-pattern may make things worse.

### 4. Keep your modules, classes, and fuctions small

Have you ever heard that small is beautiful? It's especiall true when it comes to...code 😉 Size may be something that differentiates good and bad code. Some people tend to create large pieces of code that that are generic, robust, flexible and... unreadable. Meanwhile, it is way easier to test, analyze and maintain smaller pieces of code. In short, to write good code you should keep it digestive. Therefore, it is a good idea to break down parts of the project into smaller components, classes, and modules, which increases readability and contributes to the fact that it is simply easier for other people to work with our code. Reading hundreds of lines of a complex code logic can be a true nightmare. That's why a good software craftsman write functions that are small and easy to read at once. Usually many small functions are better than a few but large ones - you can use it as a rule of thumb.

### 5. Maintain a consistent code style

Here, a good analogy is writing a book. World bestselling authors are masters at creating and using their style, which not only allows them to stand out from other authors, but also allows them to draw the reader in and keep their attention. If an author changed his style several times in his book, he might introduce dissonance in the reader. The same is true with writing code. To write clean code it is very important to format code in a consistent way. Not having a specific style for formatting code can cause someone who uses what you've created to have to guess what the changes are for. This, in turn, can lengthen the process. Consistency in style allows others to avoid confusion.

### What you can do is implement style automation for your code. EditorConfig can help you with this. It is a solution that allows you to configure and use the same code style and formatting style to achieve the right level of readability6. Trust your feelings about the code

How you feel about the code also matters. If you recognize that the code may be highly complex, it's worthwhile for you to consult other team members to gain a fresh perspective. Then you can revisit the entire process and think about where improvements are worth making. Usually if you smell that something is wrong with your code, you are right, and don't ignore that smell.

### 7. Use a second pair of eyes

Last but not least - **code review**. The only way to check if you actually produce readable code is to let someone else read it. It is hard to overstate the the **power of peer code review**. And you shall practice it whenever possible. If you use comments or provide any other explanation to developer reviewing your code, it already indicates poor readability. A second pair of eyes reading your source code is the best check of code's readability. Period.

## 100% clean code?

Being a software developer poses numerous problems and requires you to constantly think about the fact that there is always room for improvement. An essential part of expanding your coding skills is understanding that you often won't write 100% perfect code right away. There is nothing wrong with that. Making mistakes, correcting them, and then making improvements to eliminate the same mistakes in the future are a natural part of developing knowledge and experience. When you know something might not work, making improvements will allow you to enjoy better collaboration, reduced development time, and less debugging pain. Focusing on creating clean and understandable code is one of the most important factors that will help you achieve these results.

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"id": "5d4b0a8e",

"metadata": {},

"outputs": [],

"source": [

"train\_dir=r'C:\\Users\\Sree Ram\\Desktop\\ibm\\Dataset Plant Disease\\fruit-dataset\\fruit-dataset\\train'\n",

"test\_dir=r'C:\\Users\\Sree Ram\\Desktop\\ibm\\Dataset Plant Disease\\fruit-dataset\\fruit-dataset\\test'"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "e859089d",

"metadata": {},

"outputs": [],

"source": [

"import tensorflow as tf \n",

"from tensorflow import keras \n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator"

]

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"id": "5d4b0a8e",

"metadata": {},

"outputs": [],

"source": [

"train\_dir=r'C:\\Users\\Sree Ram\\Desktop\\ibm\\Dataset Plant Disease\\fruit-dataset\\fruit-dataset\\train'\n",

"test\_dir=r'C:\\Users\\Sree Ram\\Desktop\\ibm\\Dataset Plant Disease\\fruit-dataset\\fruit-dataset\\test'"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"id": "e859089d",

"metadata": {},

"outputs": [],

"source": [

"import tensorflow as tf \n",

"from tensorflow import keras \n",

"from tensorflow.keras.preprocessing.image import ImageDataGenerator"

]

}