Antoine Lechenault Follow
French Full-stack web developer based in Montréal. Web passionate.
Oct 18 · 6 min read

Useful tips to help you create good habits as a web developer

Photo by rawpixel on Unsplash

This article is aimed at junior developers, but may interest anyone as a bank of useful tips on how to grow some good habits.

I'm constantly trying to challenge myself and get out of my comfort zone. Here is the summary of the best daily tips I came up with.

I try to perfect my working habits as I'm growing, so feel free to suggest some tips not specified in this article to help me too!

Alright, first tip.

Source: Giphy.com

1. Do technology watch. A LOT.

“Technology watch” is the activity of keeping abreast of innovations in a given sector.

There’s not a lot more to say, but in my opinion, it’s the easiest way to learn just by reading. Even if you don’t understand the whole subject or learn how to do the magic thing the article talks about, you know that it exists. And that’s what really matters.

A lot of services and tools are now available online to help you do this. You should use them because they’ll be your first source of information and new ideas on a daily, weekly or monthly basis.

Tools like Netvibes can help you save a lot of time by monitoring all your technology watch in one place. Plus, I would recommend tracking new contents and read all your feeds for example on Monday and Thursday, not every day.

Some of the website/forum I read every week :

- Collectives by Codrops
- Hackernoon

- SmashingMagazine
- FreeCodeCamp
- CodyHouse
- Twitter
- Web Fundamentals by Google

But when you're using content monitoring tools, try to stay focused on a subject related to the skills you want to work on.

It's easy to get lost and jump from one subject to another. Ok, the last article on the new Crypto ICO is SUPER INTERESTING, but do you work in cryptocurrency? Stay focused.

2. Look at awesome websites. And unpack them!

You can easily find awesome websites, resources, or code experimentations here:

- Awwwards
- CSS Design Awards
- FWA
- Codepen

But the point isn't to look at them and say: "I'll never be able to do this!"

Ask yourself: how you would do the magic, and try to understand what they used to build it. New technology, new JavaScript library, unknown CSS property? We live in a world where every day new things come up and we are not able to learn as fast as the technologies grow themselves.

But I would suggest that you not only try to understand the process but also try to reproduce it, and explain it. Generally, if you can teach to someone, you have mastered the subject.

Over the past year, I spent a lot of time changing my process. Unpacking the websites and understanding how skilled people built them. All these steps helped me to combine different techniques and find smart ways to construct my websites.

That directly leads me to the next point.

3. Learn from those with more experience

Source: Giphy.com

Usually when you start as a junior in a company, you'll be directed by a Lead Dev who will have a look at your code, schedule your tasks, and prevent you from dropping the database.

Don't be shy to ask to learn more from them. Ask them to show you how to do the cool things they did on the last website they launched, or the nice component they made to make the life easier for everyone on the dev team.

But most important: don't be afraid to ask for things you think are dumb.

Everyone starts with different education, so it is natural to ask questions that may sometimes seem "silly", but those answers will help you for years to come.

Another way to learn from the best is to code review.

If your team doesn't do it on regular basis, be the first to introduce it to your dev team. Don't hesitate to ask other members of your team to review your code and try to improve it.

Listen to them. Start a debate. Explain why you did it that way and not the way they would have done it. Exchanging, discussing, and learning from each other's strengths is probably the healthiest way to improve.

4. NEVER, EVER, forget to comment your code

Source: Giphy.com

Ok, I know, you've read that sentence a million times since you started coding. On the internet, by your superiors, by your teachers, even maybe by your dog.

The point of commenting your own code isn't to make it only easily understandable by another developer or yourself. It's also a way to think about your function before writing it. To define a purpose. And only one.

Stop yourself from creating a multiSuperTaskFunction().

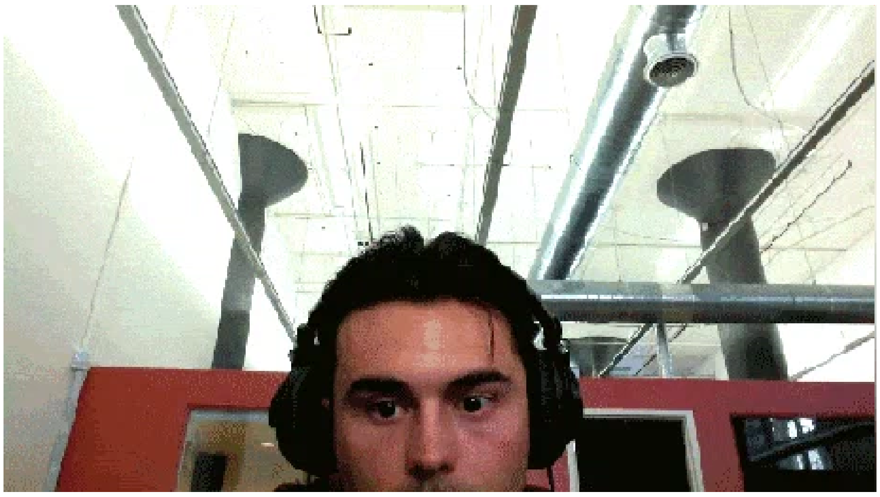
```
// I don't know Rick
mySuperFunctionAddRemoveCreateEarth( $string,
$purpose1, $bool) {
  $var1 = true;
  while($var1) {
   addNewHumans++;
  } else {
   heregoesnothing();

  }
}
```

Let's say, for example, you start to write a function to manage a planet ecosystem. With the habit of commenting your code, you'll automatically sequence your code in multiples functions. And every function has only one task to execute.

```
/*
* Handler to create the planet system
*/
Class Planet {

   public $name;

/*
   * Set the name of the instancied planet
   * @var STRING $new_name Contain only a string to
name the planet
   */
   function set_name(string $new_name) {
         $this->name = $new_name;
   }

   /*
   * Function to access the name of the planet
   * return STRING $this->name Contain only a string
with the name
   * of the planet
   */
   function get_name() {
        return $this->name;
   }

}
```

This principle of separating all your functions in small ones will help you identify your bugs more quickly. This'll also allow you to isolate elements and reuse them later (modular components). Finally, it will be much more readable and you won't need to hide when you show your code.

5. Improve your own code: Refactor.



Source: Giphy.com

If you rely on everyone except you, you won't be able to progress AND gain self-confidence. That's the reason why it's important to go back to your dirty code and clean it.

I think that's even more important than developing a new feature. So, the next time your project manager asks you what you were doing during those 5 hours of non-billable work, you can answer back: I'm refactoring my code.

> *"Code refactoring is a controlled technique for improving the design of an existing code base. Its essence is applying a series of small behavior-preserving transformations, each of which "too small to be worth doing". However, the cumulative effect of each of these transformations is quite significant. By doing them in small steps you reduce the risk of introducing errors."*
>
> *Martin Fowler, Improving the Design of Existing Code*

Long story short, try to review your own code and improve it without impacting the functionality. Faster, cleaner, more readable, modular.

The way I do it is to isolate a code part I'm not proud of or had to code quickly due to a short deadline, and come back to it one or two weeks later.

There is always a good reason to explain why you wrote that piece of bad code but you can't really argue why you didn't take the time to improve it.

An unstable portion of code will never be anything else than the sword of Damocles hanging over your head.

It can be challenging to recover your code or get involved in a complex function. But it will never be as hard as having to do it under the pressure of a bug triggered live.

Take your time, try to understand your mistakes. It will save you a lot of time, stress and health in the long term.

6. Make mistakes.

Ok, but…?!

Since I started coding, I found out that the most valuable way to improve myself was to fail again and again. Trust me, some mistakes are almost impossible to avoid, and that's totally normal.

Did you write the same function five times in the same project? I did, a lot of times. But this is not happening anymore, thanks to my experience and the processes I've created.

Don't be afraid to make mistakes. You'll keep making them time and time again, but less and less often. You never stop having to learn new things and adapting to new technology changes, but this is the beauty of our job. Learning new things every single day.

Thanks for taking the time to check out my first article on Medium!

If you liked this article, drop a 👏, follow me on Medium, and recommend this article to your friends.

Feel free to connect with me on LinkedIn! ✌️