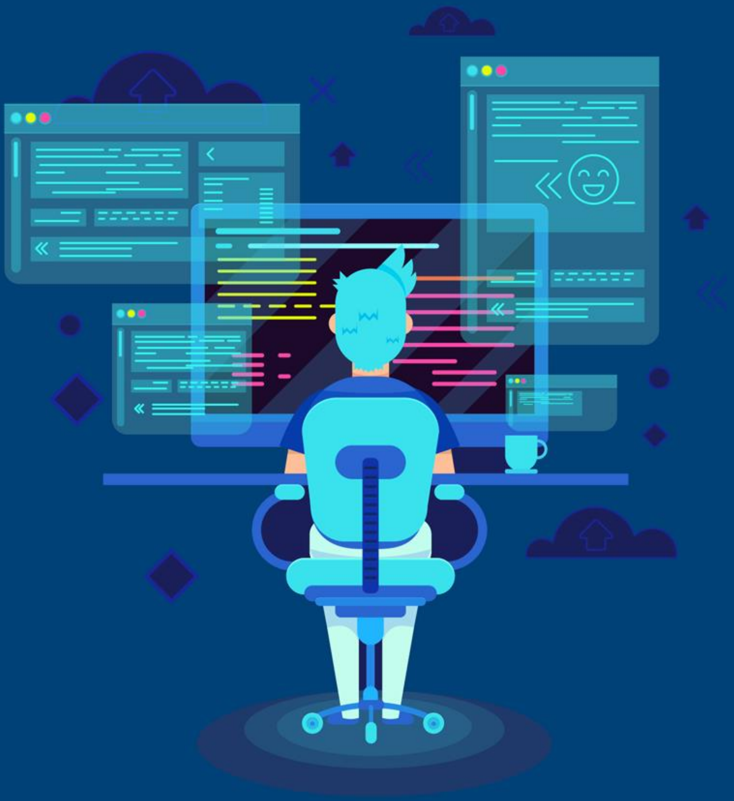
ALEX
WEB DEVELOP

7 ACTIONABLE STEPS TO BECOME A BETTER PHP DEVELOPER



Hi, I’m Alex.

This cheat sheet contains 7 actionable strategies to improve your PHP skills and to help you stand out from other developers.

I have been using these strategies myself for many years, and now I’m sharing them with you so that you can improve your skills just like I did.

So, here they are:

7 ACTIONABLE STEPS TO BECOME A BETTER PHP DEVELOPER

#1 - DON'T RELY TOO MUCH ON FRAMEWORKS

PHP frameworks like Laravel, CodeIgniter, Symfony and CakePHP are becoming more and more popular. Many developers like frameworks (especially beginners), because:

1. they help you **organizing your code** properly;
2. they help you dealing with PHP **advanced functionalities**;
3. they help you getting the work done faster and **avoiding bad coding practices**.

Frameworks have many advantages and it's perfectly fine to use them.

But they can be dangerous, because the more you rely on them, the more difficult it will be for you to code using **plain PHP**.

And in case you're wondering: you *will* need to use plain PHP sooner or later, because the more you will master your skills, the more you will find frameworks to be limiting.

Don't get me wrong, I'm not saying you must avoid frameworks completely.

But you if you want to become a professional PHP developer, you must be able to write any kind of PHP project (even complex ones) just with **plain PHP**.

#2 – STAY UP TO DATE AND KEEP LEARNING

The world of web development is evolving fast.

New technologies are constantly being released, **new tools and libraries** appear almost every day and web browsers keep changing too.

Fortunately for you, there’s no need to learn *everything* new.

However, staying up to date is important to have an idea of what tools are available (should you need them) and to make sure you don’t miss important opportunities.

The best way to stay up to date is to follow high quality resources (websites, blogs, podcasts, YouTube channels…) that offer insights and news related to web development.

But staying up to date is just half the work.

The other half is to keep learning what you need.

In my opinion, to know how (and when) to learn new things is the #1 skill of successful developers.

To sum up:

1. Stay up to date and keep your interest in the evolving world of web development alive.
2. Be committed to keep learning new things that can be useful for your work.

#3 – OVERCOME YOUR IMPOSTOR SYNDROME

Do you ever feel like you’re not *good enough* as a developer? That you’re not *qualified enough* for your work?

It’s called impostor syndrome: the feeling that you’re not good enough for what you are trying do to.

You feel like a ***fraud***, like you’re just ***pretending*** to know web development.

Believe me, you’re not alone. A lot of web developers experience this (it happens to me, too).

There is so much to know (languages, libraries, tools, frameworks) that it’s nearly impossible for a single person to learn everything.

Even so, you may think that you are *expected* to know everything. But is it true?

Let me tell you something.

In my more than 12 years as a PHP developer, I never met a single developer who knew everything. Not even once.

Nobody knows everything.

Knowing it all is not just nearly **impossible**, it’s **useless** as well. There is really no need to learn everything at once.

If you are willing to keep learning and improving yourself as needed, you will do just fine.

Don’t let your impostor syndrome get in your way.

#4 – FOCUS ON PHP AND BACKEND DEVELOPMENT

Back in the early days of the web, there were no such things as *backend* (or *server-side*) and *frontend* web development.

There wasn't any kind of web development really, just static web design.

Today, **there is a clear distinction between backend and frontend development**. They are two different worlds that require very different skills.

As web development evolves and gets more complex, it becomes more and more important for you to acquire specific skills.

In other words, you need to choose what you want to specialize in.

Nothing stops you from learning the basics of both backend and frontend development.

However, it's very difficult to become a TOP backend developer *and* a TOP frontend developer at the same time.

Instead of learning too many different things, decide what you want to focus on and aim to become an **experienced, highly specialized developer.**

#5 – LEARN HOW TO OPTIMIZE YOUR CODE

For many years I didn’t care about code optimization as much as I should have, because I thought my projects were too small for code optimization to matter.

Guess what happened?

My projects got bigger, and the lack of proper optimization eventually forced me to review and rewrite much of my code.

Learning how to write **optimized code** is a must if you want to create complex, high quality PHP projects.

And you know what? Very few developers know how to do it.

If you start caring about code optimization from the start, you will soon get an edge over other developers.

This is what you should focus on:

- Use meaningful, coherent naming for variables, functions and classes
- Comment everything: everyone should understand your code just by reading the comments
- Use the most appropriate programming paradigm for each situation; if in doubt, OOP is usually the best choice
- Avoid code duplication at all costs
- Make your code easy to debug and analyse
- Apply the Separation of Concern and Modularity principles: each element of your program (class, function…) should address only one specific problem (and do it well)
- Keep your code readable: avoid too many nested loops, variable declarations inside conditional statements (IF…ELSE) and other bad practices that could make your code less understandable

#6 – CODE WITH SCALABILITY IN MIND

Scalability is the capability of an application to keep working properly as the workload grows.

Think of Wikipedia:

it's a PHP-based web application capable of handling more than *5 million* pages and more than *200 million* daily page views very well. It's very scalable.

How can you make your web applications scalable?

First, you need to identify the potential scalability bottlenecks, like:

- The number of users using your application
- The amount of data your application needs to handle

Then, you need to analyse the link between these factors and the **scalability metrics** such as:

1. how much memory your application requires
2. how much CPU power your application uses
3. how much storage space your application requires
4. how fast your application responds

Now, you are probably thinking: *"that's fine, but my projects aren't nearly as big as Wikipedia, why should I care?"*

You should care because you can learn about scalability even with small projects. A foreach loop, a class implementation, a simple database query… everything can be made more scalable and efficient.

It's a great way to improve your coding skills very fast.

#7 – BE CREATIVE WITH PHP

PHP was created to build dynamic web pages, but you can do much more with it.

For example, you can write PHP scripts to:

- Send backend data to frontend applications
  (AJAX receivers, HTML5 apps, web games…)

- Provide support data to real-time applications
  (Node.JS, chat systems…)

- Implement REST web services

You can also write **stand-alone PHP scripts** (or PHP *daemons*) to:

- Perform database maintenance tasks
- Analyse and process data in background
- Implement an automatic email alert system
- And much more…

There are no limits to what you can do with server-side PHP programming, except your imagination.

Dynamic web pages are just the tip of the iceberg: go deeper, think out of the box and try using PHP in different ways.