the front-end mastery framework

A guide to help you start learning at a much deeper level

kevinpowell.co

CONTENTS

INTRODUCTION

When I decided to try and create a framework to master front-end development I ran into a bit of a problem. **These days, front-end development can mean a lot of different things and can branch into many different directions**.

Front-end development can mean HTML, CSS and vanilla JS. It can dive in another direction with JS libraries and frameworks such as Vue, React, Angular and more, and the divide between those and the more "traditional" HTML & CSS side of things seems to be growing.

It also keeps getting more complex with pre-processors, post-processors, and different build tools.

There are so many different directions that any one person could go in!

So how can we make a single framework that will lead people toward being a front-end master when every person could be taking a different path and heading toward a different end goal?

Well, I do think that this framework does just that, but instead of giving you a direct roadmap on what you should learn next, it's a framework to help you learn better and faster whatever it is that you're trying to learn.

BECOMING A TRUE FRONT-END MASTER

Constantly learning is a bit of prerequisite for this industry.

So really, being a master at front-end development is often about being a master at learning new things.

This framework is put together for exactly that one reason. To help you be able to learn new skills as efficiently as possible and to help those who are stuck in tutorial hell break free.

WHY I CREATED THIS FRAMEWORK IN THE FIRST PLACE

I get a lot of people who ask me how they can learn CSS just as well as I know it. They ask me how I learned it, how long it took me, and what steps they can do. I get emails, DMs and tweets about this on a regular basis.

In trying to help them out, it got me thinking. Slowly but surely, in answering people, my answer got more and more developed and it made me realize a few things about how I approach learning new subjects and what I've done to get as comfortable with CSS as I am now.

I figured if I was regularly getting asked about this there were a lot more people that would like to know as well, so I decided creating something I could share with everyone.

It also has the benefit of being a lot more thorough than the answers in emails and DMs that I was giving.

WHY I KNOW THAT THIS FRAMEWORK WORKS

I teach both online and in the classroom. When I'm teaching a subject, I need to really understand it to teach it properly. In fact, teaching has made me a much better dev because it forced me to learn things at a deeper level and really understand how they work.

So I broke down the steps that I use when I'm learning a new subject, and then to test it out, I started using this technique to teach my students in the classroom.

I also started skateboarding at age 34, and while it might seem silly, it opened my eyes to a few things that really helped me cement everything here (so yes, this framework will make you a better dev, but also a better skateboarder, or really anything else. It just works).

My students didn't realize it, but with the way I break down my content in the classroom, I intentionally put them through the steps in this process and I see the results that it gets them compared to when I use other approaches.

This framework works for people of all ages, from all backgrounds. **It will work for you, as long as you're willing to put the work in**.

It will work if you're following written tutorials, video tutorials on YouTube, or even following a really well built, in-depth course that you are struggling to really get a grasp of.

IT WORKS, BUT THAT DOESN'T MEAN IT'S A SHORTCUT

I'll be 100% honest with you right now, **if you're looking for a way to shortcut the learning process, you're in the wrong place**.

This framework will help you learn more efficiently. That means less banging your head against the wall when things aren't working and less wasting time going in circles and being stuck in tutorial hell.

That doesn't mean it doesn't take effort to do it properly though. You're here to be a master front-end developer. Mastery takes time and patience, even if you're learning as efficiently as possible.

If you want this to work for you, **you have to be willing to put the work in and to use it properly**. If you do that, it will work for you.

Also, at first it might seem a little forced, especially on some simpler topics (or at least topics which sound simple).

The more you use it, the more it will become second nature, plus you'll have a more foundational knowledge.

It's amazing how much easier the more complex stuff is when you have a better understanding of the fundamentals!

OVERVIEW OF THE FRAMEWORK

The most essential part of learning something properly—that is, really understanding it and not just having a basic knowledge of it—is to **pick one thing to learn at a time and to learn it properly**.

The basics steps of the framework are:

1. Pick a single "thing" that you want to learn
2. Research it (tutorials, videos, MDN, etc)
3. Practice along with a tutorial
4. Practice the same thing on your own
5. Implement it into a different situation
6. Keep on using it

That sounds simple, right?

Too simple maybe? You might even feel like you're already following those steps, more or less.

If you're reading this, my guess is you struggle from time to time, so if you are following these steps and it's still not working, keep reading. **We're going to break down everything in here into a lot more detail.**

If you follow these steps, you will learn things in a way that will stick with you.

HOW TO USE THE FRAMEWORK

The first three steps of this framework can often be "skipped".

By that I mean you'll often find that you're at step 3 before even thinking about it. It's important that you understand the first two steps anyway because while "picking a topic" might seem pretty straight forward, it's important to really nail down.

It sounds simple, but people don't do enough.

So what I'd encourage you to do is to read through the entire framework carefully right now. Go through all the steps, don't just skim or skip parts of it, but read through everything.

Get a good idea of how it works and make sure you understand the idea behind each step.

Once you’ve read it once through (it’s not that long of a document, take the time to do it properly!), you’ll have a great understanding of it all.

After that, I’d say just to refer to the flow chart that I included with it if ever you aren’t too sure.

STEP ONE: PICK A SINGLE TOPIC TO LEARN

The mistake that people make here is they pick "JavaScript".

JavaScript is not a single topic. Yes, it's one thing, but it can be broken down into pieces. If you can keep breaking something down, you aren't digging deep enough.

These topics need to be micro topics. Super, tiny, small, nothing topics.

It might seem crazy and even overly simple at times, but you are reading this because you want to be a master. If you want to master anything, you must put in the effort, even if it seems mundane at times.

HOW TO PICK A TRUE, SINGLE TOPIC

As I mentioned earlier, this step is one you'll often already have completed. I mean, you generally aren't worried about learning something until you have a reason to use it!

There are a number of different ways that you might find a topic.

It could be that you are reading an article and it mentions something that you've never heard about and you decide to look it up.

It might be because you are trying to figure out how to do something in your code and it just ain't working.

What's really important at this stage is that you **pick a very singular topic**.

Let's say that you've decided that you want to learn JavaScript.

JavaScript isn't just "one thing". It's made up of a ton of smaller, little things you'll need to learn, such as:

- what does it do
- what do we use it for
- the syntax
- what's a variable
    - `var` vs `let` vs `const`
- functions
    - `function()` vs arrow functions

The list goes on, and on, and on.

GOOGLE IT

If you want to know how to do something, so have hundreds of thousands, if not millions of other people before you. Google it.

So, if JavaScript is what you want to learn, find a free course on it! If it's something more specific, like CSS grid or flexbox, find a tutorial on it!

BE CAREFUL AT THIS POINT

There are a number of different places you might be researching and following along with. There are free sources like blog posts and YouTube videos, and then there are courses (both paid and free).

BLOG POSTS AND YOUTUBE VIDEOS

The problem with this type of content is it's generally an overview on a topic. Either it will be a very generic overview that lets you know all about it or it'll be a hyper specific tutorial on how to use it to accomplish one specific thing.

Both of these are great, but they can lead to you having to do a little extra work.

If I look up CSS Grid, I could get all sorts of stuff. CSS Grid has a lot of different parts to it, and all of them will be new to me.

So the trick is, **when a few new things come up, hit pause** (or stop reading), open up a new tab and put this framework into practice.

COURSES

If you're following a course you purchased, we can do the same steps, but they will hopefully be doing a lot of them for you.

Courses tend to be better than free content as they'll break content down more and really focus in on each piece, so they're a bit of a shortcut, but whatever you do, **never plow all the way through a course**.

As you progress through a course, they do the first steps for you:

1. Pick a single "thing" that you want to learn
2. Research it (tutorials, videos, MDN, etc)
3. Give you a tutorial to practice along with

What most won't to do next is give you a situation to practice it on your own other than following along with what they are doing.

So even if you're following an awesome course, slow down. When you finish a lesson or a section from the course, we always want to keep going. The forward progress feels amazing.

Stop. Follow the next steps in this framework first.

Too often people keep moving forward in a course and get stuck later on, not because the new material is too complicated, but because they didn't really understand the earlier content (I've been there too!).

Most people then, instead of going backward, bang their head against the wall, get frustrated, and way too often simply give up on the course.

LET'S FIND SOMETHING TO LEARN!

For this, we're going to role play a little.

You're in the middle of writing some CSS for what looks like a simple site, but you've run into a problem. You're adding some `margin-top` to an element, but things aren't working as you'd expect them to!

Instead of adding space above that element, it's adding space above the parent of that element. What the heck is going on here?!

[Insert image here]

Well, we’ve found our topic! We always thought we understood how `margin` works, but clearly there is something at play here that we didn’t know about.

STEP TWO: RESEARCH IT

In a situation like that above, we'll often want to solve that one specific problem that we're having. When we don't really know how to describe something like that though, it can be hard to search for the right thing.

USE GOOD SOURCES

The one suggestion I'll give to learning is to rely on good resources.

Places like StackExchange can be useful, but it can often have outdated or conflicting information as well.

Also, sometimes the best answer isn't the one that was marked as correct. I still use it, but it's not my first choice.

My suggestion is to always start with “your topic MDN”.

The MDN started as the Mozilla Developer Network, but is now just known as the MDN and is my primary resource for researching topics.

I always start there, and then if I’m still stuck, I’ll look for blogs or video tutorials that give examples of things in action.

Usually this helps reinforce what I just read on the MDN, and if there is conflicting info, I’ll trust the MDN and go find another blog post or video or two.

You aren’t a visual learner. This myth has been put to rest. Some people *prefer* to learn with video, or *prefer* reading, but that doesn’t mean it’s more effective for them, they just like it more. The best thing? Read about it, then watch a video, then talk to someone. The more different sources and types of lessons the better.

DON'T TRY TO SOLVE THAT ONE PROBLEM YOU ARE HAVING

Often we are looking up something because we run into a specific problem we are having. What happens in these situations is we look for a very specific answer.

You might find exactly what you need, but often it's not an answer, but only a solution.

By that I mean, often you'll find a chunk of code that solves your problem, but you have no idea why it works (another issue with StackExchange).

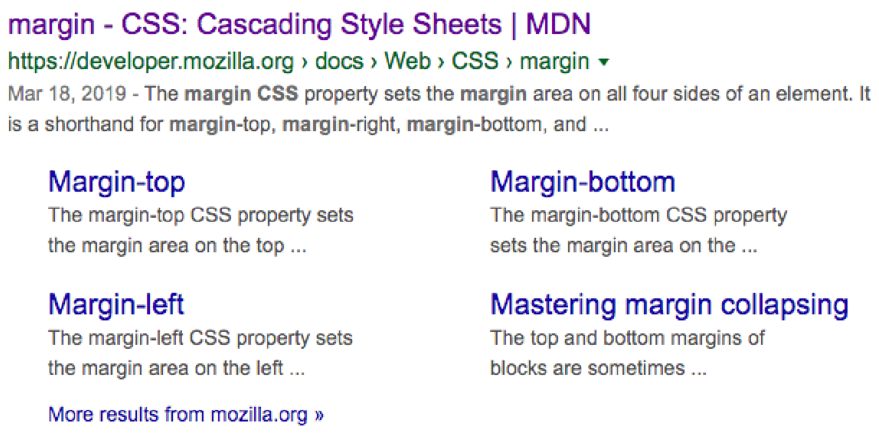
So sure, you solved the problem, but you didn't really get an answer to why it wasn't working in the first place.

This is very important: We don't want to solve that one thing that is going wrong. It's not going wrong because CSS/JS/framework is stupid, but because even though `margin` seems incredibly simple, clearly we don't know everything there is to know about it.

I realize that in the real world, "working" is good enough. We don't always have time to dig deeper. That's reality, and that's fine if you really are in a rush to hit a deadline. Right now, we're talking about learning things though, so that isn't an excuse.

If you are in a rush and found a solution that works, at the very least take a note to investigate more later on (*not* a mental note. Put a reminder on your phone, or actually take a note down on paper) and try to figure it out when you do have time.

Going back to that `margin` issue, let's skip trying to search through StackExchange for 6 year old answers and use my above advice and do a simple google search for "CSS margin MDN"



margin - CSS: Cascading Style Sheets | MDN
https://developer.mozilla.org › docs › Web › CSS › margin ▾
Mar 18, 2019 - The **margin CSS** property sets the **margin** area on all four sides of an element. It is a shorthand for **margin**-top, **margin**-right, **margin**-bottom, and ...

Margin-top
The margin-top CSS property sets the margin area on the top ...

Margin-bottom
The margin-bottom CSS property sets the margin area on the ...

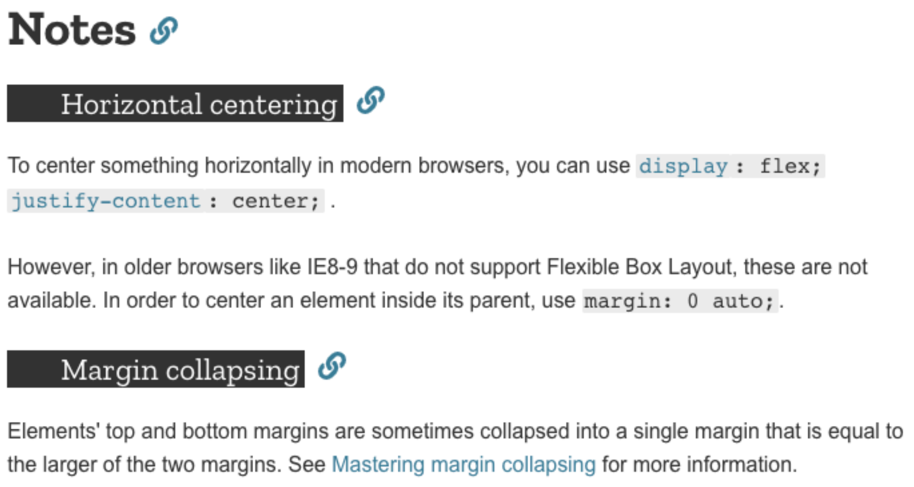
Margin-left
The margin-left CSS property sets the margin area on the left ...

Mastering margin collapsing
The top and bottom margins of blocks are sometimes ...

More results from mozilla.org »

And everything there looks fine but what the heck is "Mastering margin collapsing" about?

Or, even if you missed it there and clicked on the main result, you probably started skimming through it because everything was really straight forward and then suddenly you see this:


Notes

Horizontal centering

To center something horizontally in modern browsers, you can use `display: flex; justify-content: center;`.

However, in older browsers like IE8-9 that do not support Flexible Box Layout, these are not available. In order to center an element inside its parent, use `margin: 0 auto;`.

Margin collapsing

Elements' top and bottom margins are sometimes collapsed into a single margin that is equal to the larger of the two margins. See Mastering margin collapsing for more information.

What in the world is "Margin collapsing"?

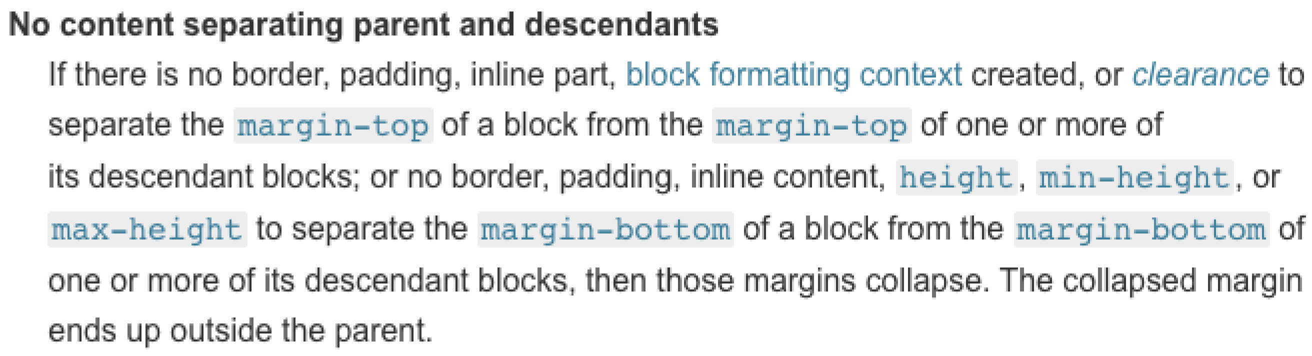
Now this is an important decision you have to make. Sometimes you'll see something like this that you know doesn't have to do with your problem, but is still something you've never heard of before.

You have a choice in that case. Pause what you are looking for and follow this new path with plans to return later on, or take a note to look into that later and continue on what you are currently looking for.

Remember, we need to have a single focus right now!

In this case, there is nothing else on the margin page that you haven't seen, so you click into their "Mastering margin collapsing" page and dive in to see if that might be what's going on here.

And in there, you find this interesting (albeit verbose) paragraph:



No content separating parent and descendants

If there is no border, padding, inline part, block formatting context created, or *clearance* to separate the `margin-top` of a block from the `margin-top` of one or more of its descendant blocks; or no border, padding, inline content, `height`, `min-height`, or `max-height` to separate the `margin-bottom` of a block from the `margin-bottom` of one or more of its descendant blocks, then those margins collapse. The collapsed margin ends up outside the parent.

That very last sentence is key. So even though I'm putting a `margin-top` on the child, it's ending up on the parent instead. Strange? Sure is, but we just learned something!

Now, this page doesn't give a real solution to the problem, but at least we know why it's happening (there are some implied solutions that you could figure out, but it doesn't explicitly say "do this to fix it").

But it does bring up something interesting. What the heck is block formatting context and how do we create one?

We could click that link and discover an entirely new rabbit hole. But we must decide, are we putting collapsing margins on the side, or are we diving into a new rabbit hole for awhile, because we only want to focus on one thing at a time!

It's really easy to fall down rabbit holes. Sometimes you have to dig deeper to really figure things out, but a lot of the time, just taking a note (remember, it has to be a real note of some kind) is the best idea.

STEP THREE: FOLLOW ALONG WITH AN EXAMPLE OR TUTORIAL THAT YOU FOUND

The MDN gives examples. Some are better than others. In this case, I think it might be a good idea to find a tutorial that does a better job of showing me how to actually fix the issue though.

And since we're role-playing, we come across a nice video that explains how it works, reinforcing and clarifying what we saw on the MDN. Even better, it has links to 2 codepens so that we can follow along with the video!

And this is essential. I realize not all videos or blog posts have code that you can use to follow along, but as much as possible, follow along with the tutorial (or even the MDN examples).

Recreate what they are doing and play with the code to make sure that you understand what is happening. Tools like Codepen and JS Fiddle are perfect for this sort of thing.

This is incredibly important to do. People often get stuck in tutorial hell because they read/watch a tutorial, then they feel like they understand, and when they try to implement it in their own project it doesn't work. Then they return to watching another tutorial, and another, and another.

Practice along, even if it's a really simple example file like in the above video.

The other advantage of actually practicing here, even if it's SUPER simple, is that it also builds up a little muscle memory. It's one thing to see it and remember it. **Writing it and making it work is key**.

In this case, the video also shows us not only how they work, but also how we can solve the problem too, which is awesome.

So, now we know that instead of using `margin-top` on the child, or trying to fuss around with using some of those things the MDN mentioned, we could simply used `padding-top` on the parent and we'll be good to go.

STEP FOUR: PRACTICE THE SAME THING ON YOUR OWN

This is more useful in more complex situations, such as creating a JavaScript function or trying to create a layout with CSS Grid, as these involve a lot of sub-components (we'll talk more about learning these bigger components later), but I do think it's worth doing even for simple tasks as it doesn't take long.

This is where you **take what you learned in the tutorial and try to recreate it from scratch without looking back at the tutorial/video/lesson that we followed**.

And this isn't where you do it in your own code. Y**ou need to replicate exactly what you just followed** along with, but without referencing the video.

By doing this, we are helping to make sure we understand what we actually set out to learn!

This step is vital because often we can't do all of it, especially as things get more complicated.

If you can recreate everything and you're feeling comfortable, that's amazing. You move onto the next step.

If you cannot recreate everything on your own, you go back to the previous step.

So far back that you're re-watching the tutorial (skipping to the relevant bits that you are stuck on) and following along. **At this stage you are not fixing your current file!** You are following along once again.

Once you finish working out the kinks and you think you figured it out, you **start over from scratch again**.

This is what everyone skips.

They watch a tutorial, follow bits and pieces of it, then keep going back to it to fix their file until what they created is working.

They didn’t do it on their own, they kept going back for help.

You need to redo whatever it is from the very beginning on your own before you can move onto the next step. If you want to cheat and skip ahead because you’re sure you know it, you aren’t cheating anyone but yourself.

Do whatever was done all on your own, from start to finish without watching ANYTHING or reading ANYTHING.

When you can do that, it means that you’re starting to understand how it works and it’s time to move on.

STEP FIVE: IMPLEMENT IT IN A DIFFERENT SITUATION

No one does this.

No one.

But you want to be a front-end master, and you want to learn things properly. To do that, you're going to do this step.

This, combined with the previous one where you completed the entire tutorial on your own without having to look back at it is going to unlock things for you.

You're going to take what you just did, and you're going to try to use it again in a different situation.

In the case of our collapsing margins, I'm going to throw together a very nice and simple layout. I'm going to make some margins collapse to make sure I get what's happening when they do. I'm going to play with different values to make sure I really get it. Then I'm going to see if I can recreate the problem I originally had, and then solve it.

I could even just add onto the previous code I wrote where I was following along with the tutorial. I don't need a new file, it doesn't have to be a big site, I just want to go through the steps again but in a slightly different situation.

Or let's say that you just learned how JavaScript functions work and you made an alert pop up when you click a button.

Make a new button that makes an alert with different text show up, or maybe instead of a button, can you do the same thing but if I click on text, or anywhere on the page?

Sometimes you change the situation a little more and you find something unexpected that actually forces you all the way back to the research phase! But most of the time you'll nail it, or maybe have to go back to the tutorial, or maybe find another tutorial to help you figure out what's wrong, or for a little bit of clarification.

If that happens, if you go back a step, you are starting from that step and working your entire way back through to this point.

You might feel like working through all those steps is a waste of time, it was just a little small thing that you needed clarification on. But this is where you separate yourself from everyone else.

You go through all those steps again, and by the time you do, you know how that one thing works so well you could explain it to someone in your sleep.

STEP SIX: KEEP ON USING IT

This is the easy part! You've done the hard work and you actually, really understand what is happening.

You've followed along with a tutorial.

You redid the steps of the tutorial on your own without help.

You took what you learned and applied it to a new situation.

You did all that, and now you know it so well you could explain it all to someone in your sleep.

Now you can use it with confidence!

Truth be told, you'll probably still run into issues once you've done all this, but even though you may not know exactly how to fix it, it'll take a fraction of the time to figure out.

You can use this new knowledge of yours now, and you’ll help yourself start coding with confidence.

BONUS FINAL STEP: TEACH IT

No, you don’t need to find a group of people to teach things to. You don’t even need to find one person.

But teaching things is one of the best ways to reinforce what you’ve learned. You have to talk about it in ways you weren’t thinking about it before. You have to break the steps down into more parts.

I’ve become a much better designers and developer since I started teaching.

I always encourage strong students in my classroom to help the weaker ones. Not because I don’t want to do it myself, but because I know they’ll learn so much by doing that.

You don’t need to do it publicly either.

Start a blog where you post what you learned about each day, week, or month.

Write one short tutorial about it whenever you have a chance.

The blog can be an WordPress.com site that you don't share with anyone. No setup required, no one cares about the theme. Just start doing it.

Or even better, make it public, even if you don't share it. You never know, someone might stumble across it one day and your post will really help them out.

Or you could do like I did and start a YouTube channel, but that's a much bigger commitment. I love doing it and would encourage anyone to do the same if they can invest the time into it!

You could even find a friend that's interested and just teach it to them. Or honestly, if you're really worried about it, even a Google doc where you write things down to save for yourself and no one else.

It doesn't matter how you decide to do it, but I 100% guarantee it'll help you be a better developer.

WHEN THINGS GET MORE COMPLEX

We looked at `margin`, and discovered we didn't know about collapsing margin. That's cool, but fairly simple.

What happens if you want to learn, say, CSS Grid?

I'd start by researching it and seeing what it is. Then I'd watch a tutorial on it which would open up a TON of questions.

- `grid-template-columns` and rows
- `grid-areas`
- the `fr` unit
- `minmax()`
- implicit and explicit grid
- line numbers and line names

There is a lot to grid, so where do we start? Well hopefully that initial research gave you a few ideas, but let's say for fun that the thing that interested you the most was `grid-template-areas`.

If you start researching those, you'll quickly see that they refer to `grid-template-columns` and `grid-template-rows`, so then we can pivot and see how those work first.

Once you fully understand how the row and column system works (after going through all the above steps) then we move back into `grid-areas`.

Heck, you could say that before using `grid-template-columns`, you first need to know:

- HTML
- How to link CSS to an HTML document
- The basic CSS syntax
- CSS units
- etc.

When you know those things really well though, they're implied parts of the puzzle.

We don't have to return to how to create a CSS document each time. So if you already know something, you don't need to keep breaking it down or going further back.

The same applies for, say, learning how a function works in JavaScript. If you jump straight to that, and you start hearing all about variables, and loops, and other things, you need to go back and figure those out.

But if you already know what a variable is, you don't need to rewind all the way back to there, you go to loops and figure that out, then keep moving forward.

CONCLUSION

This framework might seem like a lot of work to implement, but once you get into the swing of things, it’s not that bad, and in the long run it saves you time.

It saves you time because, while you do have to invest more at the very beginning, in the long run you will understand things so much better it’ll blow your mind.

It also becomes second nature. Instead of just reading something, it becomes second nature to follow along with the tutorial, and then to practice it in other situations as well.

You start to become more curious and to try and break things down and see how they work in different contexts.

You also start realizing that there is often a lot more going on that you didn’t even know about before!

And once you’re doing that, you’ve won.

You’re doing the work, you’re going to start understanding things at a deeper level and you are going to be a master front-end developer!