EPISODE 20

[INTRODUCTION]

[0:00:00.1] NA: Welcome back to the La Vie en Code podcast. Dedicated to helping self-taught web development students, learn, grow and become exceptional web developers. I'm your host, Nicole Archambault, and I'm a fellow self-taught front-end web developer and educational technology entrepreneur. If you're joining us for the first time on the podcast, Welcome! I hope you'll find some great value in this episode and don't forget to stick around at the end of the episode for a special offer just for La Vie en Code Podcast listeners. I have some really cool stuff in the works that I'm super excited to share with all of you.

So, today we're going to be talking about one of the biggest mistakes that I see new web developers out there making and I, too, made these mistakes. So, I'm by no means, exempt whatsoever. But that mistake looks something like this: New web developers out there learning to code and probably working through some kind of guided eLearning platform like Treehouse, freeCodeCamp, Coursera, Udemy courses and you're cruising along in the beginning doing really basic stuff, getting, you know, great feedback that you're doing well, because let's face it, there is usually, maybe only, a couple of ways tops to approach the more basic stuff that you're learning early on, you know, probably some HTML, CSS and you get up to like JavaScript if you're in the front end. Or if you've been purely back end from the start, you know, maybe that's when you get to like object-oriented programming or something.

I don't know? That's where I personally started stressing a bit, I think, and you start stressing because you have to like, stop occasionally and think back to what you knew. Even if it clicks, your application is clunky at best and it can be really frustrating to be at that point. And everyone, I mean, everyone gets to that point when they start stressing because it's starting to get hard it's really beginning to not feel so comfortable and at that point you're officially not cruising anymore; you hit traffic. So, what do you do? You're not feeling confident, or competent, you know, where you are. How does that make you feel on a deeper level? Possibly pretty doubtful about your place in the industry or even worried about your ability to get a job, maybe?

And if you're not careful, our brains start to associate those feelings with coding itself. And, before long you could drop off entirely because you don't feel that same pleasurable rush that you initially did. Hey, I get it 100% and like I said, I've absolutely been there. Sometimes, people begin focusing in at this point on their skills again. You know, they think they're just lacking something, some maybe they need more practice. You may have just restarted an entire lesson or a course on, you know, trying to revisit what you missed to fortify it in your mind. But let's just assume for this case that you actually stick with the course wherever you are and you decide to push through it. We all know that what you need at that point is practice, right?

There's a ton of adages out there like "practice makes perfect", you know, which is silly because nothing is perfect, ever. Throw that right out of your head right now because that'll probably keep you from doing code at all, more likely than it will, you know, help you in your coding. So, practice helps us improve and that's the best that we can hope for and we also know that consistency is key, right? You need to show up everyday and learn and practice in order to grow. But, you may not even realize as I didn't for a long time that the way you were practicing your coding has a lot more bearing on your overall success than nearly showing up everyday to practice. You may even be practicing incorrectly. At this point, my setup, a bunch of folks are probably, like furrowing their brows and thinking, "But Nicole, how can you practice wrong?" Or, like, "Why is there another person here to tell me that I'm not doing enough or something the right way?" We get enough of that, too.

Well, I'm going to give you some information about the psychology of learning that should get you onboard though, and that's a nice part about a lot of the topics that I discuss on the La Vie en Code podcast. There are folks out there actually doing the research on this stuff. So, today we're going to be talking about the right way to learn, a skill known as, deliberate practice. Mind you, I called it a “skill” for a reason; skills can be learned and practiced with deliberate practice. I'm also going to show you some of the benefits of practicing the right way and demonstrate the harm the practicing incorrectly can actually do. And finally, I'll give you some tips on how you can act today on introducing deliberate practice to your life.

If you are the type that likes those checklist that, you know, you can print out and post on your cubicle or wall, as some folks had said that they are, I gave in because I love you and I have a

rad download for you guys at the end of the episode as well; an at a glance checklist for making sure that your practice is as deliberate as you can make it.

Okay, well we've got a lot to cover, so let's go.

[EPISODE]

[0:06:06.1] NA: So, what the heck is deliberate practice? I mean, you're sitting down, practicing is deliberate, right? You made a deliberate choice to sit down and practice. Well, that's a part of it. But deliberate practice has some set characteristics by which we can identify it. Deliberate practice, generally speaking, is like regular practice but with a specific goal of improvement. When you do something with a very specific goal, it makes sense that your approach would change. If you want to prepare for a marathon, you'll train and practice differently than you would if you were training for, like, a hundred meter dash. Similarly, if you're practicing with a goal of learning JavaScript in order to be a front-end developer, you'll want to focus in on JavaScript and not a whole bunch of other things that you don't need.

Deliberate practice also requires focus and attention, as well as some sustained effort and energy. Focus, attention, energy; those are all really limited resources that we possess as human beings. But, I can't really think of a better way to use your energy than on improving yourself, right? So, let's talk about what deliberate practice looks like so you can see the differences. First, it focuses on the quality of the practice not the quantity. If you look at most resources and writings out there about practice, they encourage consistency. You know, showing up everyday, ready to learn. Showing up is indeed half the battle and you can't get better without practicing the information that you encounter, which is why we stress when the training wheels are taken off.

But, just showing up and doing some stuff to get better at coding isn't going to cut it, because it isn't sustainable. Now, I've been hearing the 10,000 Hours quote that was popularized by Malcolm Gladwell for years now and honestly, I kind of think it's a load of crap. Sorry, I don't hold back on my opinions here. What happens when you practice for 10,000 hours? What if you focus on the wrong things? Or you find that you still can't recall related or pre-requisite information in between your learning sessions? 10,000 hours of unfocused work is meaningless,

which is why I also have strong feelings about established industry folks counting their years of programming experience. I mean, often they're looking things, like basic things up and referencing them as often as newbies are or you know, maybe a little bit more because of course, they've had more time exposed to it. Because they never really took the time to master their work.

Mastery is really, really, rare, folks. It's not as common as we're led to think it is, since a lot of the noise in the industry comes from the folks who claim that they know what they're doing. So, we just kind of have to trust that when we first come in. Just remember, that low quality practice sessions are poorly structured and they lack clear goals or desired outcomes. High quality practice sessions on the other hand, have a greater chance of the student emerging with understanding that sticks. Second, deliberate practice is highly structured. So, in other words, you can't just kind of wander into a practice session with a very loose goal to get better. Get better at what? How much better are you going to get? Once you get to that point, what's your next small step to reaching the bigger goal? Without structure, you're not likely to stick with it, either, and that's a big part. If you're not going to stick with something, then there's no way that you're going to make it work.

Third, you need to be getting immediate feedback on your progress. Fortunately, for web development in particular, errors serve is an excellent learning tool. Also, a lot of eLearning platforms like Treehouse and freeCodeCamp that integrate practice on their sites with test-driven development environments provide instant feedback to satisfy this criteria.
But even with the feedback, like errors in test-driven development, there are other considerations. Is that everything involved in meeting your goal of being a really good developer in this particular area? What about things like, technique, style, and approach to problem-solving? We still mostly need human beings for that, since it's humans and not machines, that ultimately use our code. We also need that feedback to be super informative with specific information regarding how and why you were right or wrong, and guidance to learn more as needed.

So, human and computer feedback, as you can see, is a really critical part of deliberate practice as we're learning the code. Fourth, the best practice involves doing the same task or similar or related tasks repeatedly. Practice is not watching videos or reading tutorials; you should know

this. Even simply repeating the same actions over and over, won't necessarily produce improvement. So, think of it this way: if you show up to deliberately practice a specific action, like writing a function from scratch because you're not comfortable with doing it on your own yet, you would set up multiple challenges to write a function from scratch in order to deliberately practice it.

If you know that your weak area is writing functions, but you decide to write things that aren't functions, you're not practicing deliberately. Because deliberate practice requires exposing yourself to the same or similar challenges in order to bring you comfort and familiarity with solving them. There are some tricks to fix this that I'll share with you later. A lot of new web developers think that all expert developers program from what's commonly called a "flow state". They think that the developer just naturally knows how to overcome obstacles.

In reality, they practice that task frequently enough that they recognize all the obstacles on their path to their goal. Deliberate practice can absolutely lead to more relaxed natural programming and problem solving ability. But you cannot achieve expertise without deliberate practice. It's just not possible. You'll sustain in mediocrity at best, and that's okay also for a lot of folks. I want to recognize that. But I want to help people become the best developers that they can be if they want that. So, if I have anything to say about it, you're adding deliberate practice to your toolbox today.

We're going to move on to how deliberate practice can help you as a self-taught web developer. But to recap deliberate practice is different than what's commonly referred to as “practice” because it focuses on the quality of the practice over the quantity. In order for practice to be deliberate, it needs to be highly-structured, provide detailed feedback and guidance for improvement immediately, and involve completing the same or similar task repeatedly. That's it! Now you know that there's basically a super practice available to you to take your learning to the next level.

[BREAK]

[0:14:59.1] NA: Okay, so now we know what deliberate practice looks like, and what distinguishes it from standard practice, but how can it help you become a better programmer?

How can it help you get a job? How can it help you build up a sexy portfolio that will catch clients' attention? How can it help you make more money at whatever you end up doing? How can it help you get to these much bigger goals that we actually identify with, and perk up at the mention of? The answer is that, because deliberate practice can be applied to literally anything, not just coding, you know. You can focus it on just the specific skills that you need to be able to get your dream job.

So, let me say that again, framed a little bit differently. When you know what you need to be practicing, and you know how to practice it deliberately the best way that you can, you then have complete control of your education. Anything short of having a goal and working deliberately towards it everyday, is really going to be a waste of your time if you have a concrete goal like what I mentioned.

I mean, if your goal is to stay and chill learning web development and kind of like, hit some stuff along the way, by all means, take as leisurely an approach as you like. If it's just a casual thing you're not looking for a job, have fun. I always say, I never honk at anyone driving the speed limit on the highway. I'll just go around them if they're in my way. I have great appreciation for the scenic group and stopping to smell the roses, and it's what makes life amazing.

But, if you apply that leisurely mentality to hefty goals that mean a lot to you, without identifying what path exactly you need to take to get there, you're probably going to find that the equation just doesn't add up to achieving your goals at all. That was where I ended up and big question, is deliberate practice like your one key to web development success? I'd say it's not "the one", but it's certainly up there. And, here's the thing about practice: it needs to be practiced like anything else. That's pretty meta, huh?

Another thing, practice isn't always fun. But it's necessary to becoming good at the job that people are going to pay you good money to do, whatever it might be. That's what we all want. You want them to pay you a lot not because of your title, but because of how damn good you are. So, you can't let them down and you can't let yourself down. No pressure and plus, games and gamified learning, can actually help with the fun part, as I was talking about in EdTech September. But, you still need to do the work to show up and maximize those tools.

I talked about this a bit in Episode 18's Tips for Maximizing your Web Development and E-learning Experience, also. E-learning platforms provide some of the tools, but you have to bring the heavy machinery. Your learning skills, your motivation, your focus and a personalized strategy to help achieve your goals. Learning deliberate practice by doing it every single day will change your life. I don't know how I always feel like it's super tacky when people say that, like "This is going to change your life." But, I've seen in more cases with people that once they actually figure it out, how to get their practice going properly, that everything just kind of took off.

So, you'll start realizing that things just get easier, because your brain has full context for how to overcome the challenges that you're inevitably going to face. Because so many problems and code also can be solved with common approaches, you may even find yourself building on solutions that you already feel comfortable with, to create new and more efficient ones. Seriously folks, I wouldn't dedicate a whole episode to deliberate practice if it weren't something that you really, really need.

When you're learning web development or anything, you know, "properly", you come to realize, that it's really nothing more than a cycle, a daily cycle of reminding yourself why you want it, what you want, of course, too, refocusing in on your goals, and then putting in the deliberate practice, not just regular practice or work, while remembering that it's a marathon not a sprint so you don't burnout. That's the careful dance that is learning web development. You're not going to be an expert overnight. Also, it's just not possible so you need to adjust your expectations there.

But you could be looking back in 30 days, 3 months, 6 months, a year from now really, really glad that you made changes to your approach to learning code back, you know, after listening to Nicole's amazing podcast. It's really up to you, though. I'm just here to provide guidance to help you facilitate that transformation.

[BREAK]

[0:20:49.1] NA: So, if you're sold on deliberate practice at this point and the power it unquestionably holds over your ability to succeed, then I've done my job. I'm so glad. The next

step is going to be figuring out how to add Deliberate Practice in to your everyday life in a way that you can actually commit to and see changes from, starting today. I really go into a lot more depth in my course, which I'm going to talk about shortly after. But here's what I can tell you here on the podcast through my little bit of personal experience, this is what has worked with me for integrating deliberate practice into my life.

I've always struggled too, as a disclaimer, with focus issues. So unless I really put literally every drop of my energy in to sitting my ass down and getting the deliberate practice even started. It probably won't happen. I can think of a gazillion things I'd rather be doing in most cases than starting Deliberate Practice. And guys, that doesn't change. It just gets a little bit easier to sit down and focus each time because your brain gets positive feedback that you're getting results and brings light good feels.

So, I had great success personally using the Pomodoro Technique with an app called Be Focused Pro, which I'll link in the show notes and using that app to set super short 15-minute intervals with one to two minute breaks. Just knowing that the breaks were there had like a profound psychological effect on me because it didn't feel like I was just working non-stop. But at the same time I didn't usually end up taking them. I often worked right through them and that's where you know that some real, you know, some magic is at play. Something's going on when your just kind of not even thinking about going against her bad behaviors and you're just kind of fixing it.

I also turned on the chronometer, which is that tick, tick sound. At first, just the first I don't use it anymore or if I'm having difficulties sometimes I use it still to keep myself aware of the fact that I was being timed. Which tends to help me focus on the task at hand. I am an anxious perfectionist and I find that perfectionists tend not to have a very good sense of time that it takes to complete a task. We think that something will take forever when it might just take a few moments. We think that it will just take a second, you know, we constantly late to stuff too, totally guilty. But that perfectionism really is what you need to be overcoming, if that's your case. You need to be aware of time as your first step in the right direction and that chronometer sound really did help with that.

So, I picked one particular thing that I wanted to practice. It might be closures, it might be API calls, it might be algorithms on freeCodeCamp. Make a list of stuff that makes you uncomfortable. If you're comfortable with it obviously you don't need to practice it as much. But what you want to be focusing on is not practicing the stuff that you are already comfortable with to make yourself feel better because it won't make you feel better. In the long run, you're only going to feel better by improving which means you have to do the things that are tougher.

If you're doing freeCodeCamp working your way through there, what I love about the algorithms is that you can do those algorithm challenges as often as you want. Then you put in the deliberate practice. You design some tasks. So all write out a few on a piece paper something or just, you know, set up some task. Design some tasks or pick tasks that already have been created on like a website. Do some Googling.

You can even just do challenges you received on your eLearning platform but alter them slightly because your brain will respond better to having things slightly changed. There are some tools out there that I'll link in the show notes that can help you, like flashcards for coding, there's spaced repetition software, which I've spoken about before. You know, don't stop with just one iteration as solving a problem though. Go back and do it again, and again, and again, and again. Pay attention to how you're feeling as you go. The stress really should start to kind of melt away as you remember your approach last time with each pass.

Oh, if you get stuck practicing, that's completely okay like you're stuck while making your practice it could easily derail you as getting stuck anywhere again. But, in my opinion, that actually means that you're practicing content a little above your head, which is a good thing. So, pause, figure out how to overcome it and then find a way to integrate that new challenge in to your challenge orientation. The key here is that you want repeated exposure to challenges so they don't seem scary and you build a deeper context for how to work through them. And, that approach has worked for me for so many things not just programming.

For instance, I applied this to working out early this year, twenty pounds ago. Rather to mindlessly to going in to a workout to work out, which means really nothing as I have been doing, I took the time to plan out what particular things that I want to get better at and I based them on things that I wanted to do but couldn't do yet. You know, because I was pretty out of

shape. So maybe like a certain number of push ups, planking for a certain amount of minutes, cardio in a certain heart rate range for a certain amount of time.

The goals were all super specific so I'd know when I met them. If you don't know when you're meeting your goals it will feel like you're never meeting them. So, that's why you need to build a context for that and I wasn't getting feedback on other really important things too, like form and posture, which I'd really like to change in the future. I would like to have somebody giving me feedback in one person.

Unfortunately though, that's one of the downsides of not having a personal trainer and deciding to go at it on your own. Much like teaching yourself to code! See where I went with that? But, we can bridge that gap of self teaching by having our friends and other people who are better at that particular task help us. I could record my workouts, actually film them and send them to a friend to give me critique, so I would be less likely to hurt, you know, to hurt myself and end up out of working out entirely if I was executing my workouts with poor form.

Anyway that's just my experience and examples. I honestly don't know much about the specific steps that others take with deliberate practice besides kind of what I adapted for myself. So, I'm really interested to know how others approach their deliberate practice sessions. Please leave me a comment on the episode page, which is up at lavieencode.net/20 or you can hit me up on Twitter @lavie_encode.

[END OF EPISODE]

[0:28:53.1] NA: Whoa! That was a packed episode. I know I always forget to do this myself, but if you found value in any of the information that I provided to you today or ever, please don't forget to leave a five-star review and a comment for the La Vie En Code Podcast in whatever your favorite podcast-listening app is. Also if you haven't already be sure to subscribe on iTunes, GooglePlay, Stitcher or SoundCloud. New episodes of La Vie En Code podcasts are available every Tuesday at 9 a.m. eastern.

As I promised at the beginning you can also download a special cheat sheet for ensuring you're practicing deliberately. It's at the episode page at lavineencode.net/20 and you can view the

show notes on the transcript for the episode there as well. So, go download that, take action today and let me know how it goes.

My other La Vie En Code promise to you — scouts honor — is, you know, as my listeners and readers I will always try to keep things a sync shortcut-like and down to just what you need while ensuring that you still understand the why behind all of it. For some folks though, they just need a little bit more support than I can offer through the podcast and the blog. It can also be a lot to try to pull all of this information that you need to know from everywhere when a lot of the time you don't even know how to articulate the problem that you are facing. I know that's probably familiar, and that's totally fine. I both heard and saw that pain and frustration in the newbie devs that I've talked to.

So, if you feel that way, know that you're not alone, first off. You're not any less also of the developer if you need a little extra help. Self-teaching means a lot of different things to different people and this is why I was so excited to officially and finally announce that we're closing in on the 30 Days to Web Development pre sale cart open date. 30 Days to Web Development is my very first online course and it's a prep course for career changers in to web development.

So you guys, this is the course that I really needed as a new web developer before I even started learning with Treehouse like I should have this before I started using Treehouse. I say that too with a lot of mixed feelings because I know I could've saved myself a lot of time, money, energy had I known what I know now. But, I can't go back in time. I can move forward with the solutions I've found, which I have and I have the opportunity now to use my knowledge and to help other people achieve their goals in the process.

We're going to cover so much stuff and you're going to be a transformed learner and an aspiring web developer by the end of just four weeks. You're going to learn how to avoid wasting money and time with resources that don't fit your needs and goals overcome the anxieties holding you back from committing to a career change. Identify your ideal job, and develop a super detailed career transition path. Communicate with perspective employers to catch their interest and your unique assets, and then finally enter the web development learning arena with greater confidence and more direction. Because I want to make a 100% sure this course covers everything career changers need to make a successful transition in to web development, I'm

actually going to be fleshing out the course with a special VIP group of beta or pre-sale customers.

I have a very thorough outline that I've vetted, but it doesn't make any sense for me to spend all this time recording videos until I know exactly what's the most helpful to the students. And guys, all these considerations should really show you how much this course means to me and how much it centers you. I don't succeed with the course at all unless you're able to totally own your web development education and start working with a company, start freelancing, or start creating your own products, you know, entrepreneurially for your own customers.

In exchange for helping me create the course tailored specifically to their needs, those special VIP folks are going to be getting 50% off the lowest price this course will ever be. As it gets more fleshed out, I can assure you that the price is going to go up. But it's also my goal to make this course as accessible as possible. So, once it’s officially launched to the public there will be payment plans available and even some giveaways for course access to 30 Days of Web Development.

So, if you're super stoked now and ready to find out more about 30 Days of Web Development, always sending out VIP pre-sale information only to folks on the pre sale notifications list. Head on over to 30daystowebdevelopment.com and sign up for those updates so you can be sure you don't miss your opportunity to get the course at the lowest price it will ever be offered at. As always, you can get a hold of me for questions, feedback, conversation, whatever; I'm available on Twitter @lavie_encode. That's @lavie_encode and the La Vie en Code Facebook page lives at laviencode.net/facebook. That will take you to the Facebook page, it's just much easier to type in the Facebook URL.

Please allow me to join you next Tuesday at 9 a.m. eastern, perhaps for your morning or your evening commute. We will be discussing hackathons what they are, what they aren't and how you can go compete in and win one. My authority here is that my 1st ever hackathon team was like total badass and took first place.

So, if my confused beginner self can hack together an award winning solution with people I had never met before that day, I'm pretty certain that I can help you build the confidence to get

hacking in no time. Definitely be sure to catch that episode, and until next time my dearest friends, peace, love, and code.

[END]