What every beginner absolutely needs to know about the journey ahead

Quincy Larson was just a "guy in a suit in an office" and decided he wanted to learn how to code. So he asked around. He started by picking up a bit of Ruby then found himself skimming through other languages like Scala, Clojure and Go. He learned Emacs then Vim and even the Dvorak keyboard layout. He picked up Linux, dabbled in Lisp and coded in Python while living on the command line for more than half a year.

Like a leaf in a tornado, the advice Quincy received jerked him first one way and then another and then another until he'd finally taken "every online course program imaginable". By the end of it all, despite having ultimately landed a software development job, Quincy:

> ... was convinced that the seemingly normal programmers I ran into were actually sociopaths who had experienced, then repressed, the trauma of learning to code.

Ouch. Does that sound familiar?

Phase I: The Hand-Holding Honeymoon

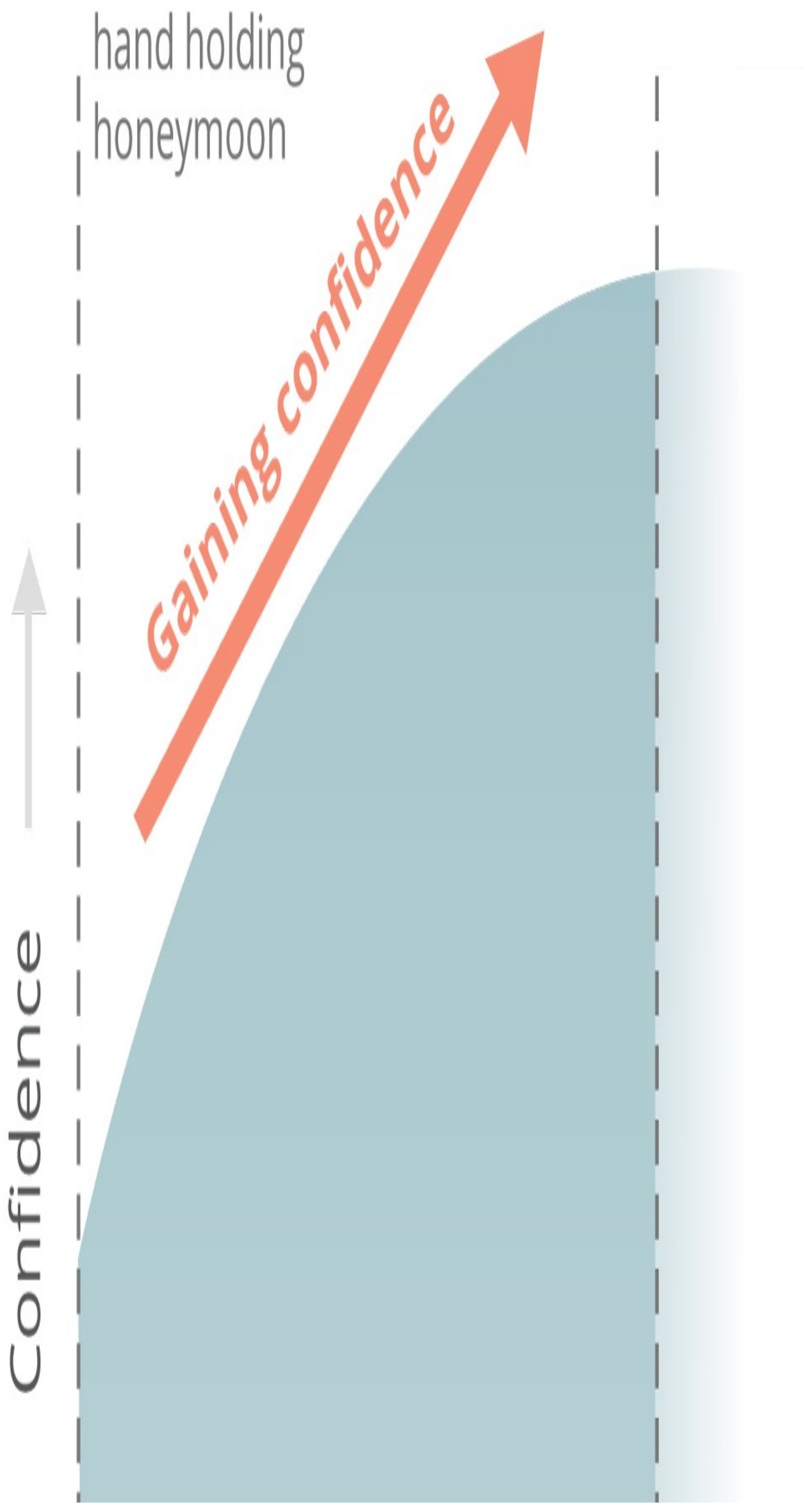
It's really hard to blame anyone for coming into the programming industry with outrageous expectations.

On the one hand, you've heard rumors of how difficult programming is since you were young, like old wives tales meant to scare children into studying social sciences instead.

On the other, the "Learn to Code" movement has done a fantastic job of breaking down barriers and showing people that code is actually quite harmless. Tools like Codecademy and Treehouse and Code School reach out with the gentlest of touches to assure you that you too (nay, anyone!) can not just learn to code but become a full-fledged developer as well.

Suddenly the problem isn't fear, it's an overabundance of hopes and high expectations.

And, for the most part, these introductory tools do a great job of guiding you like a child in a crosswalk past the big scary variables and conditional statements and through the early phases of programming syntax. As you conquer one after another of their gamified challenges, your confidence rises. Maybe you can do this after all! How hard can it be? *You're basically a developer already!*


hand holding
honeymoon
Gaining confidence
Confidence

The Hand-Holding Honeymoon

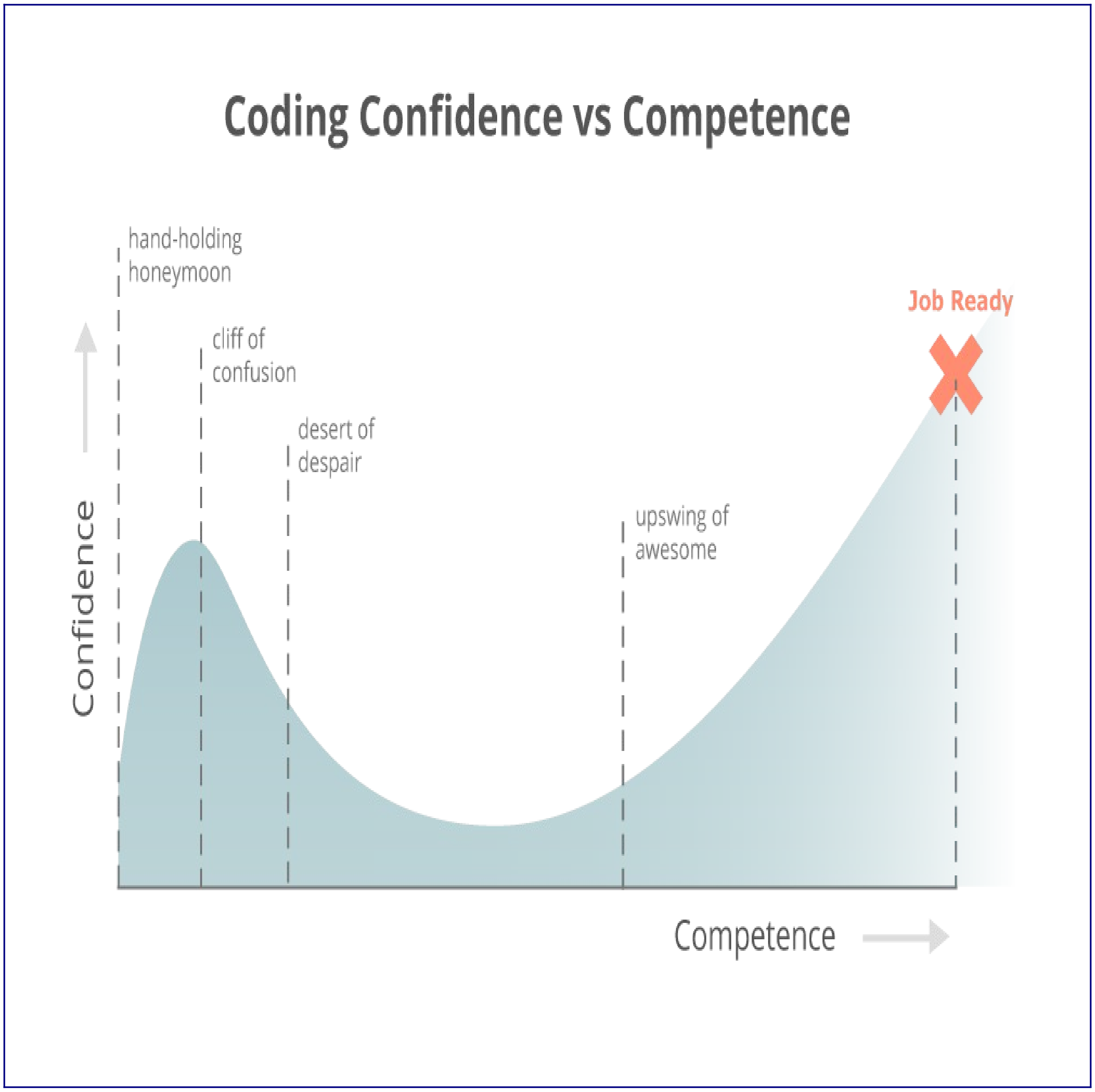
Here's the problem -- you're in what I like to call the "Hand Holding Honeymoon" phase. Though you may feel like the end is around the corner, you're only a fraction of the way there. This is just the beginning...

Charting The Path Ahead

Before we dive into Phase II, let's look at the bigger picture.

In this post, I'll walk you through the four phases of the typical journey into coding and what you'll need to do to survive each of them. You'll also see how two key factors -- the density of resources and scope of required knowledge -- define this journey.

The trek towards job-readiness can be plotted in terms of how your confidence level changes as your capability increases:



The Learn-to-Code Journey -- [Click to Enlarge]()

This is a relevant relationship because your confidence is highly correlated with your happiness and because the point where your confidence and capabilities match is the best proxy I have for the sweet spot when you're officially "job ready".

We'll look into the unique challenges of the remaining 3 phases in a moment, but this is what each of them essentially involves:

1. **The Hand Holding Honeymoon is the joy-filled romp through highly polished resources teaching you things that seem tricky but are totally do-able with their intensive support. You will primarily learn basic syntax but feel great about your accomplishments.**
2. **The Cliff of Confusion is the painful realization that it's a lot harder when the hand-holding ends and it feels like you can't actually do anything on your own yet. Your primary challenges are constant debugging and not quite knowing how to ask the right questions as you fight your way towards any kind of momentum.**
3. **The Desert of Despair is the long and lonely journey through a pathless landscape where every new direction seems correct but you're frequently going in circles and you're starving for the resources to get you through it. Beware the "Mirages of Mania", like sirens of the desert, which will lead you astray.**
4. **The Upswing of Awesome is when you've finally found a path through the desert and pulled together an understanding of how to build applications. But your code is still siloed and brittle like a house of cards. You gain confidence because your sites appear to run, you've mastered a few useful patterns, and your friends think your interfaces are cool but you're terrified to look under the hood and you ultimately don't know how to get to "production ready" code. How do you bridge the gap to a real job?**

I've interviewed hundreds of aspiring developers over the past several years and heard echoes of the same story again and again. My goal for this post is that you approach the learner's journey with both eyes open and enough of a plan that you can avoid the common pitfalls of those who have come before you.

Let's get back into Phase II...

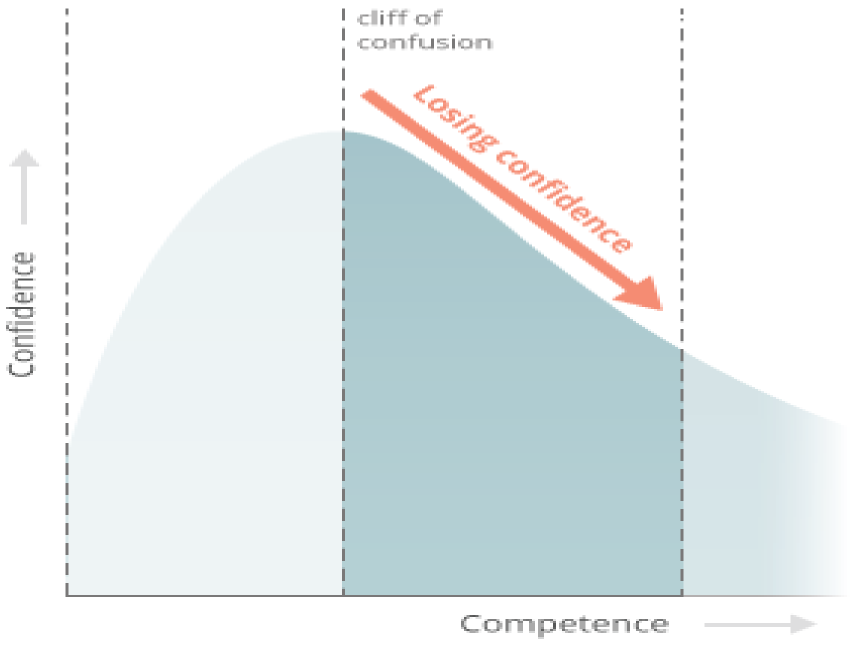
Phase II: The Cliff of Confusion

So, you're in Phase I -- the "Hand-Holding Honeymoon" -- checking off badges and completing coding challenges while your confidence and capabilities grow. This isn't so bad... what's all the fuss about? You've arrived at the "Peak of Irrational Exuberance"...

Be careful! You're about to overstep a precipice that's broken many strong aspiring learners and relegated them to the "coding is too hard" camp. The precise moment this leap occurs is the first time you sit down at your keyboard, open up your text editor, and try to build a project from scratch without any of the fancy in-browser editors, scaffolded code or helpful hints.

Crap.

You might stretch this out a bit by following tutorials, but no one has ever reached the skies without leaving the ground, and, at some point, you're going to have to create magic from a blank text file. You've just entered the second phase of learning, where confidence comes crashing down to earth -- the "Cliff of Confusion":



The Cliff of Confusion

So you build. You fight and scratch your way to a barely-functional solution but there's something missing. You're at a war with bugs that makes Starship Trooperslook benign. It feels like each victory was gained only by a stroke of lucky Googling and your confidence that you can ever figure this stuff out plummets.

Buuuuuuuuuuuuuuuuuuuuug!!!

This is a particularly frustrating phase to see as an educator and to all participants in our industry. Programming may not be perfect for everyone, but we *want* you to make progress because sometimes the unlikeliest of stories become the grandest successes.

When the hand-holding ends and students are pushed off the cliff and told to fly, too many potentially awesome people are spiraling onto the rocks of frustration without learning how to flap their wings.

The scary part is that you haven't even gotten to the meaty stuff yet. This second phase, the Cliff of Confusion, is still very early. Once you've finally squashed enough bugs to end the eighth plague of Egypt and actually finished a couple of projects -- thus marking the end of Phase II -- you're still just getting started.

For those who are truly ready to make a career out of this, surviving the Cliff of Confusion is often the point where you decide to go all-in with your new life. But too many are left behind. And, unfortunately, you're just about to enter the "Desert of Despair".

The Two Key Factors at Play

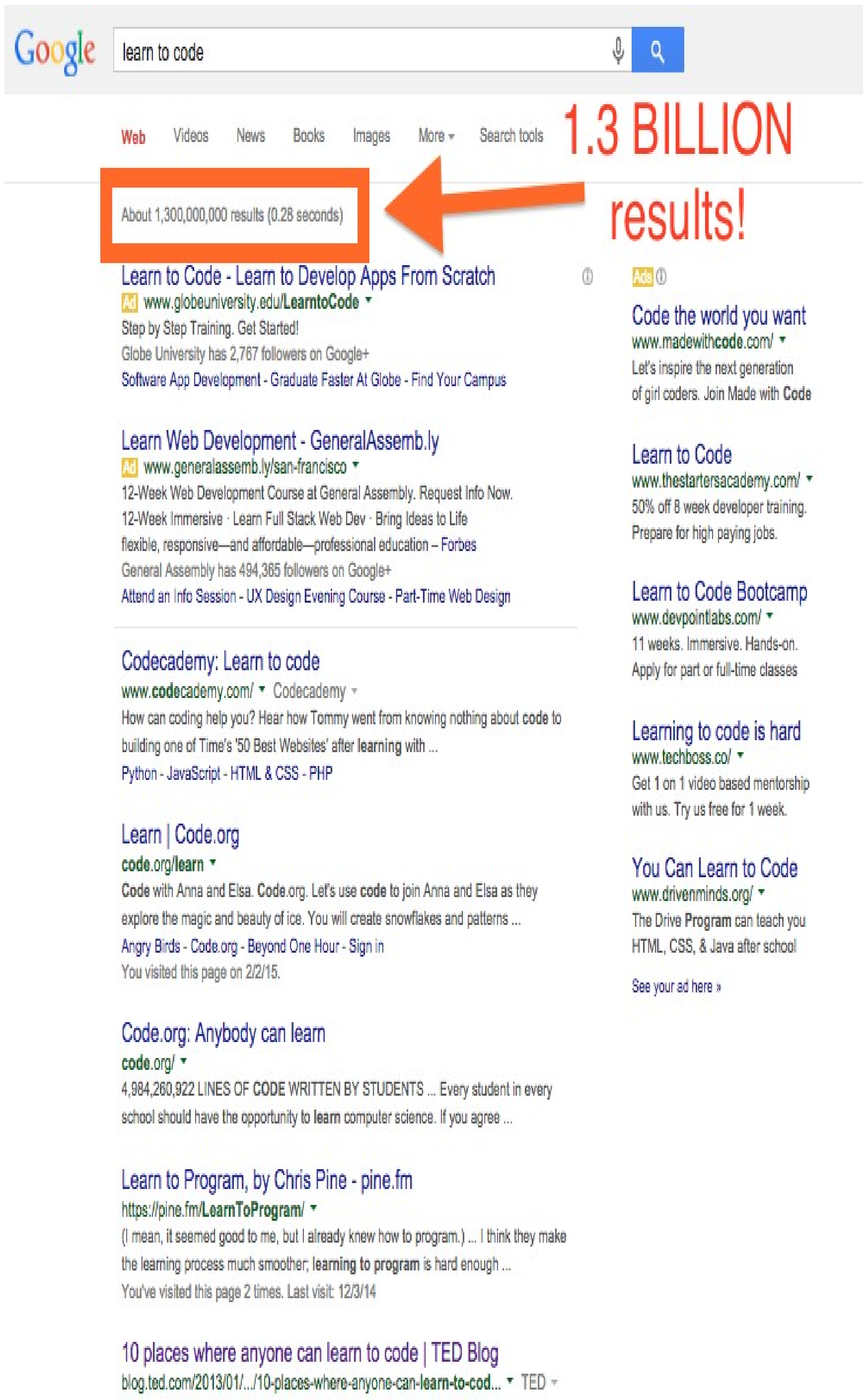
So what really marks the difference between one phase and the next? Why was Phase II (the Cliff of Confusion) so awful compared to Phase I (the Hand-Holding Honeymoon)? Understanding this will help you realize that it's not your fault at all if your journey looks like what we've just described.

Basically, there are two key forces at work in every phase -- Resource Density and Scope of Knowledge. Let's see what these are before exploring how they define Phase III.

Factor 1: Resource Density

As I said above, when you first start out, it feels like there are a million resources out there trying to hold your hand and pull you into coding. That's because there are!

Search for "Learn to Code" and you'll be hit with a wall of helpful and useful tools, texts, videos and tutorials. And, frankly, they're great! Never before have there been so many ways to start learning to code.


learn to code

Web Videos News Books Images More Search tools

1.3 BILLION results!

About 1,300,000,000 results (0.28 seconds)

Learn to Code - Learn to Develop Apps From Scratch
Ad www.globeuniversity.edu/LearntoCode
Step by Step Training. Get Started!
Globe University has 2,767 followers on Google+
Software App Development - Graduate Faster At Globe - Find Your Campus

Learn Web Development - GeneralAssemb.ly
Ad www.generalassemb.ly/san-francisco
12-Week Web Development Course at General Assembly. Request Info Now.
12-Week Immersive · Learn Full Stack Web Dev · Bring Ideas to Life
flexible, responsive—and affordable—professional education – Forbes
General Assembly has 494,365 followers on Google+
Attend an Info Session - UX Design Evening Course - Part-Time Web Design

Codecademy: Learn to code
www.codecademy.com/ Codecademy
How can coding help you? Hear how Tommy went from knowing nothing about code to building one of Time's '50 Best Websites' after learning with ...
Python - JavaScript - HTML & CSS - PHP

Learn | Code.org
code.org/learn
Code with Anna and Elsa. Code.org. Let's use code to join Anna and Elsa as they explore the magic and beauty of ice. You will create snowflakes and patterns ...
Angry Birds - Code.org - Beyond One Hour - Sign in
You visited this page on 2/2/15.

Code.org: Anybody can learn
code.org/
4,984,260,922 LINES OF CODE WRITTEN BY STUDENTS ... Every student in every school should have the opportunity to learn computer science. If you agree ...

Learn to Program, by Chris Pine - pine.fm
https://pine.fm/LearnToProgram/
(I mean, it seemed good to me, but I already knew how to program.) ... I think they make the learning process much smoother; learning to program is hard enough ...
You've visited this page 2 times. Last visit: 12/3/14

10 places where anyone can learn to code | TED Blog
blog.ted.com/2013/01/.../10-places-where-anyone-can-learn-to-cod... TED

Ads

Code the world you want
www.madewithcode.com/
Let's inspire the next generation of girl coders. Join Made with Code

Learn to Code
www.thestartersacademy.com/
50% off 8 week developer training. Prepare for high paying jobs.

Learn to Code Bootcamp
www.devpointlabs.com/
11 weeks. Immersive. Hands-on. Apply for part or full-time classes

Learning to code is hard
www.techboss.co/
Get 1 on 1 video based mentorship with us. Try us free for 1 week.

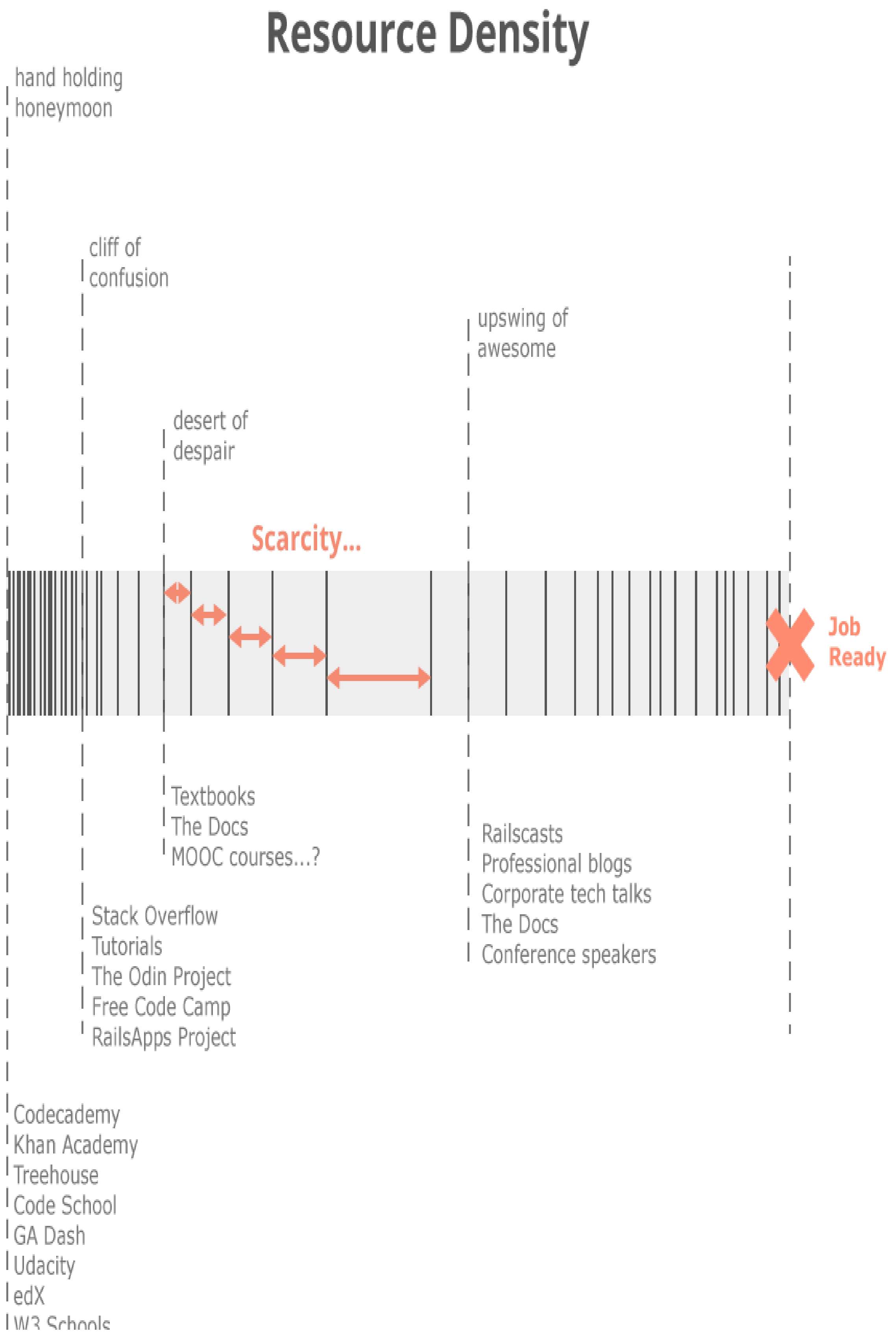
You Can Learn to Code
www.drivenminds.org/
The Drive Program can teach you HTML, CSS, & Java after school

See your ad here »

Unfortunately, in later phases the density of resources drops off fast. Anyone who's made the jump from beginner to intermediate can attest that there is a BIG difference between the amount of resources available when you first start out versus when you're first looking for help building things on your own without too much hand-holding.

This problem exacerbates as the amount of knowledge increases rapidly entering Phase III, and is one reason why we call that phase the "Desert of Despair". Once you get past this and start to become comfortable with what exactly you need to search for, the resources return and you're able to work with more technical tools like industry blogs and screencasts. Part of this is just understanding which questions to ask.

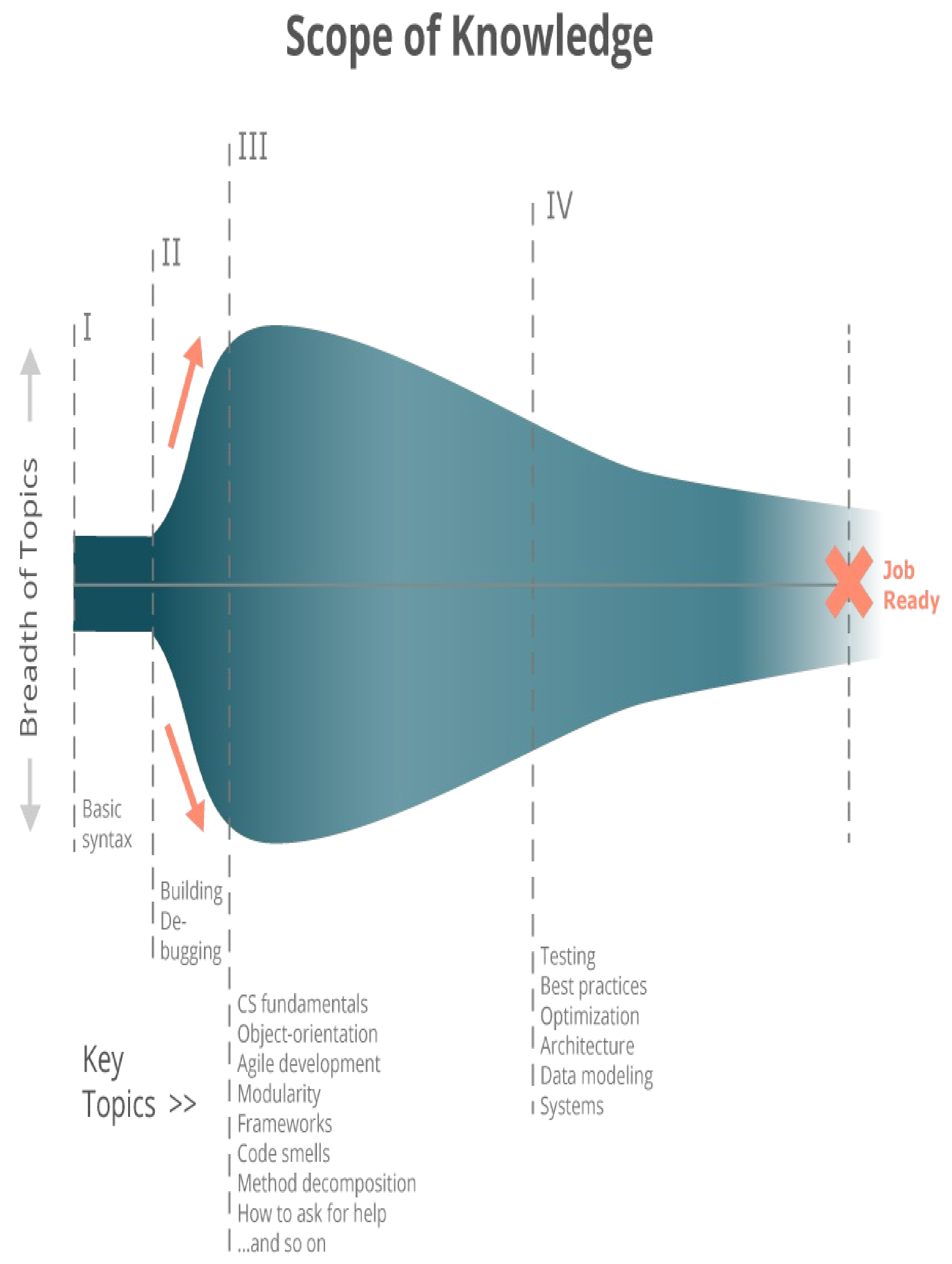
Here's what the Resource Density looks like in each phase (greater line density indicates more resources):


Resource Density
hand holding
honeymoon
cliff of
confusion
desert of
despair
upswing of
awesome
Scarcity...
Job
Ready
Textbooks
The Docs
MOOC courses...?
Railscasts
Professional blogs
Corporate tech talks
The Docs
Conference speakers
Stack Overflow
Tutorials
The Odin Project
Free Code Camp
RailsApps Project
Codecademy
Khan Academy
Treehouse
Code School
GA Dash
Udacity
edX
W3 Schools

Density of Resources in Each Phase -- Click to Enlarge

Factor 2: Scope of Knowledge

Now let's talk about a related issue -- the Scope of Knowledge. This represents the total breadth of new topics you need to learn in each phase. Here's what it looks like:


Scope of Knowledge
Breadth of Topics
I
II
III
IV
Job Ready
Basic syntax
Building De-bugging
CS fundamentals
Object-orientation
Agile development
Modularity
Frameworks
Code smells
Method decomposition
How to ask for help
...and so on
Testing
Best practices
Optimization
Architecture
Data modeling
Systems
Key Topics >>

The Scope of Knowledge that's Required in Each Phase -- Click to Enlarge

When you first start learning, the set of things you need to understand is narrow. Everyone, regardless of goals or language or background, needs to figure out what a `for` loop is, how to build conditional logic, and other basic structures of programming syntax. There ultimately aren't even that many of these fundamental concepts so the Scope of Knowledge during that phase is very narrow.

As soon as you get away from the basics, you see a rapid broadening of the Scope of Knowledge as you need to begin picking up things that are more difficult like understanding errors and *when* to use the code you know know *how*to use. This is different because there is no "correct" answer to a clear question... things get fuzzy.

When you progress into the third phase, the scope of knowledge balloons wider. You now need to understand what tools to use, what languages to learn, underlying CS fundamentals, how to write modular code, object-orientation, good style, and how to ask for help (to name just a few). Every trip to Google or Hacker News takes you down another set of rabbit holes and overwhelms you with more things you don't know but feel like you should.

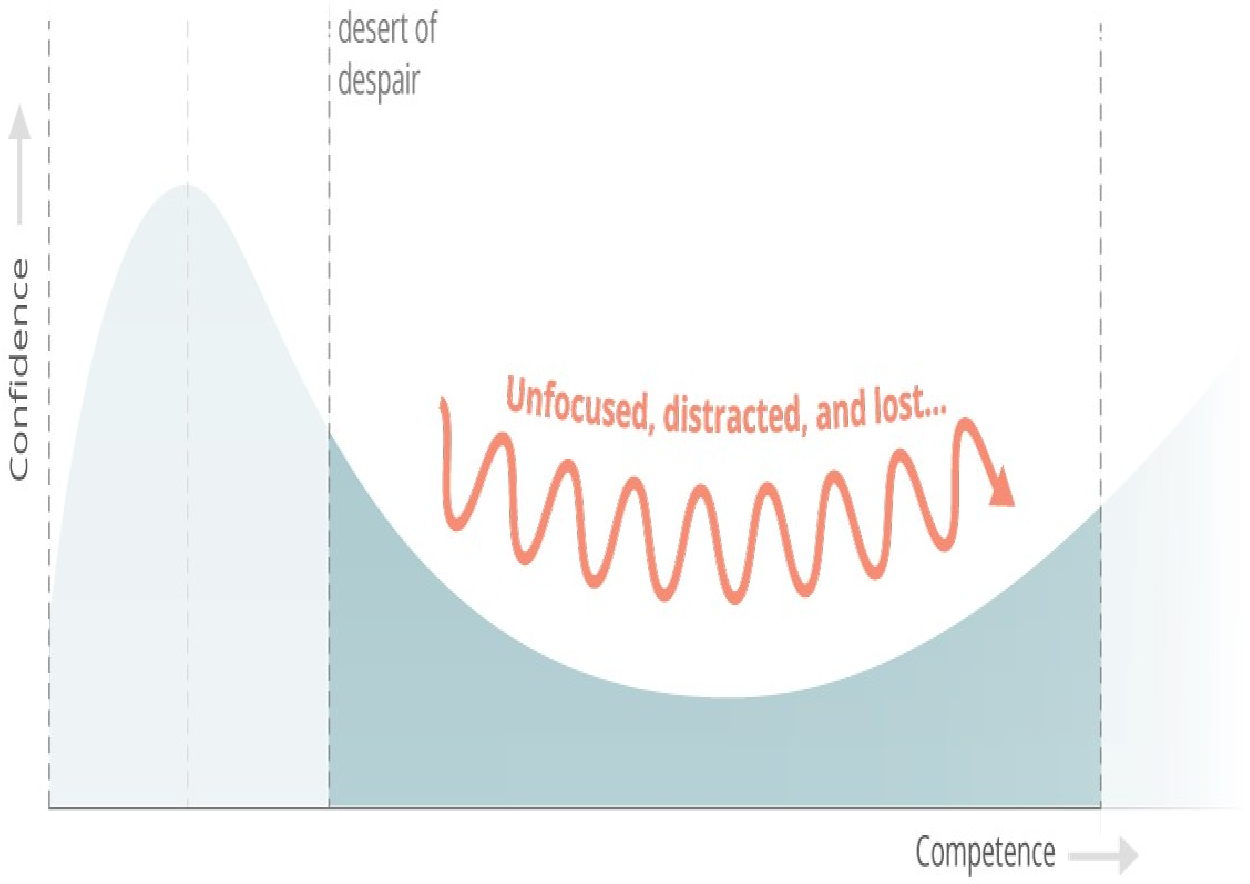
You don't know what you don't know.

Only when you've finally found some traction and left the desert does the scope again begin to narrow. By that point, you've found your chosen technology and its place in the ecosystem. You finally (pretty much) know what you don't know and can plot a path through it. You will continue to increase focus as you push onward and into the beginning of your career.

Phase III: The Desert of Despair

With an understanding of these factors, you can see that the Cliff of Confusion is really just a turning point. The pain caused by the toxic combination of a rapidly increasing Scope of Knowledge and a falling Resource Density results in what I call the "Desert of Despair".

In essence, this desert is where you know there's an end *somewhere* but you don't know how to get there:



The Desert of Despair

The desert is long and fraught with dangers. You'll find yourself drawn to "Mirages of Mania" along the way -- dozens of tempting resources which appear to hold the solutions you're looking for but which will deposit you, once again, in a place where lonely sand extends to each horizon line.

Maybe you sign up for a couple MOOC courses from Coursera or Udacity or edX. Or you find a tutorial which purports to take you all the way. You thought you learned the lessons of the Hand Holding Honeymoon -- that there are no easy answers -- but the temptation to seek salvation is too great and you fall for the promise that *this one* will get you to the finish where the others did not.

You can't learn this stuff in a week or a month or a single college class no matter what anyone says so stop falling for that!

There is a LOT more to learn than you probably expected. Even if you're able to get some apps running, it's hard not to feel lost in the greater scheme of becoming a true professional. It's difficult to measure your progress. How do you know what you need to learn or if you're even learning the right things?

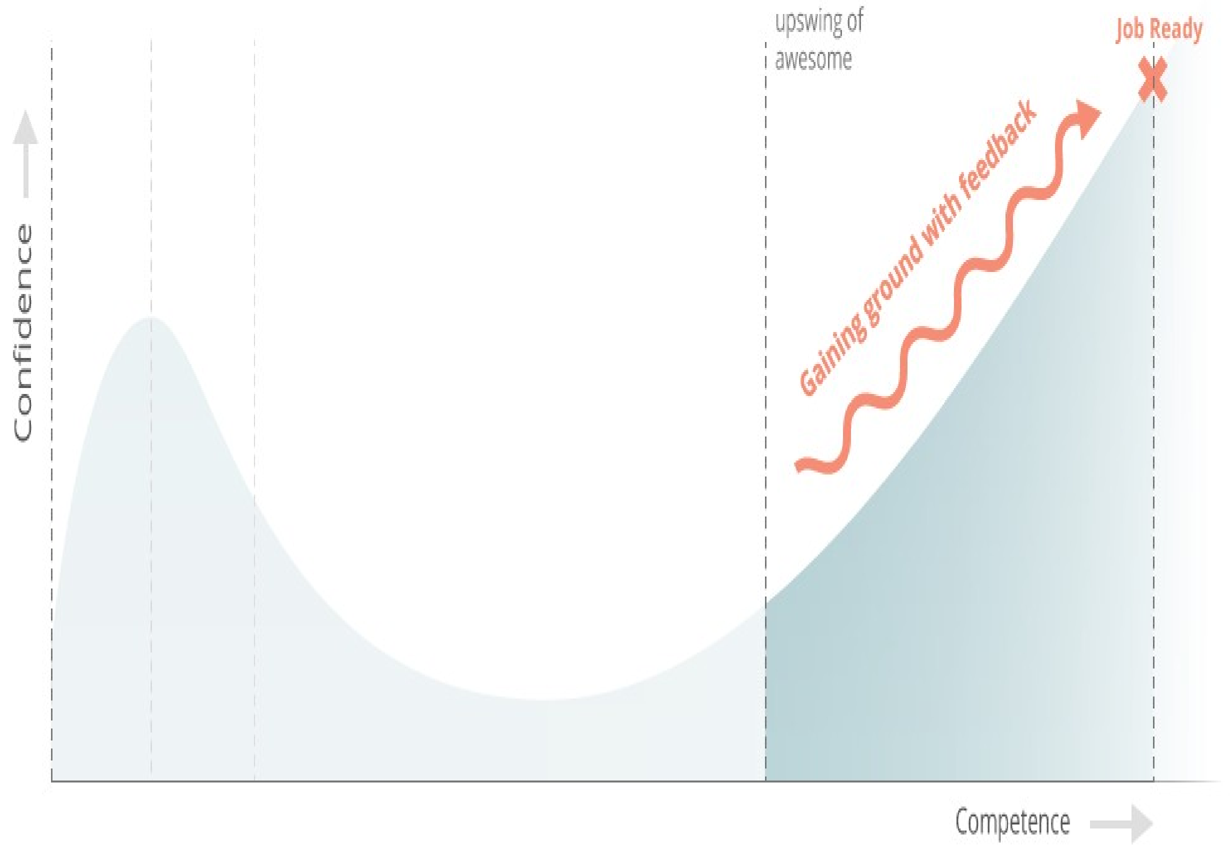
Even if you're pointing the right direction, it's hard to measure your progress. You might feel totally lost until the very moment when you're finally able to build something that looks and acts the way you expected it to. But, with enough perseverance and a good compass, you'll eventually get your first few "real" projects launched and you'll realize that you're finally starting to *get it*.

Sure it's been hard up until now, but maybe this web dev stuff isn't so bad after all... Everything's coming up Milhouse!

Phase IV: The Upswing of Awesome

You've made it through the desert and your confidence is growing. Your Google-fuis excellent and you're finally able to understand those detailed industry blog posts and screencasts. Maybe you've gone deep into a particular language or framework and you have confidence that you can build and launch a functioning application.

This is the "Upswing of Awesome":



The Upswing of Awesome

All may seem well to the outside but you know deep down that you're not there yet.

You can make that application work but what's happening beneath the surface? Your code is duct tape and string and, worst of all, you don’t even know which parts are terrible and which are actually just fine. Your periodic flashes of brilliance are countered by noob mistakes and, worse, a creeping suspicion that you still don't have a damn clue what you're doing.

This is a bipolar phase. You feel like half of you is a bulletproof developer and the other half is a thin veneer of effectiveness covering a wild-eyed newbie who is in way too deep. The further you progress, the more a gnawing sense of uncertainty grows that someone is going to "out" you as a fraud.

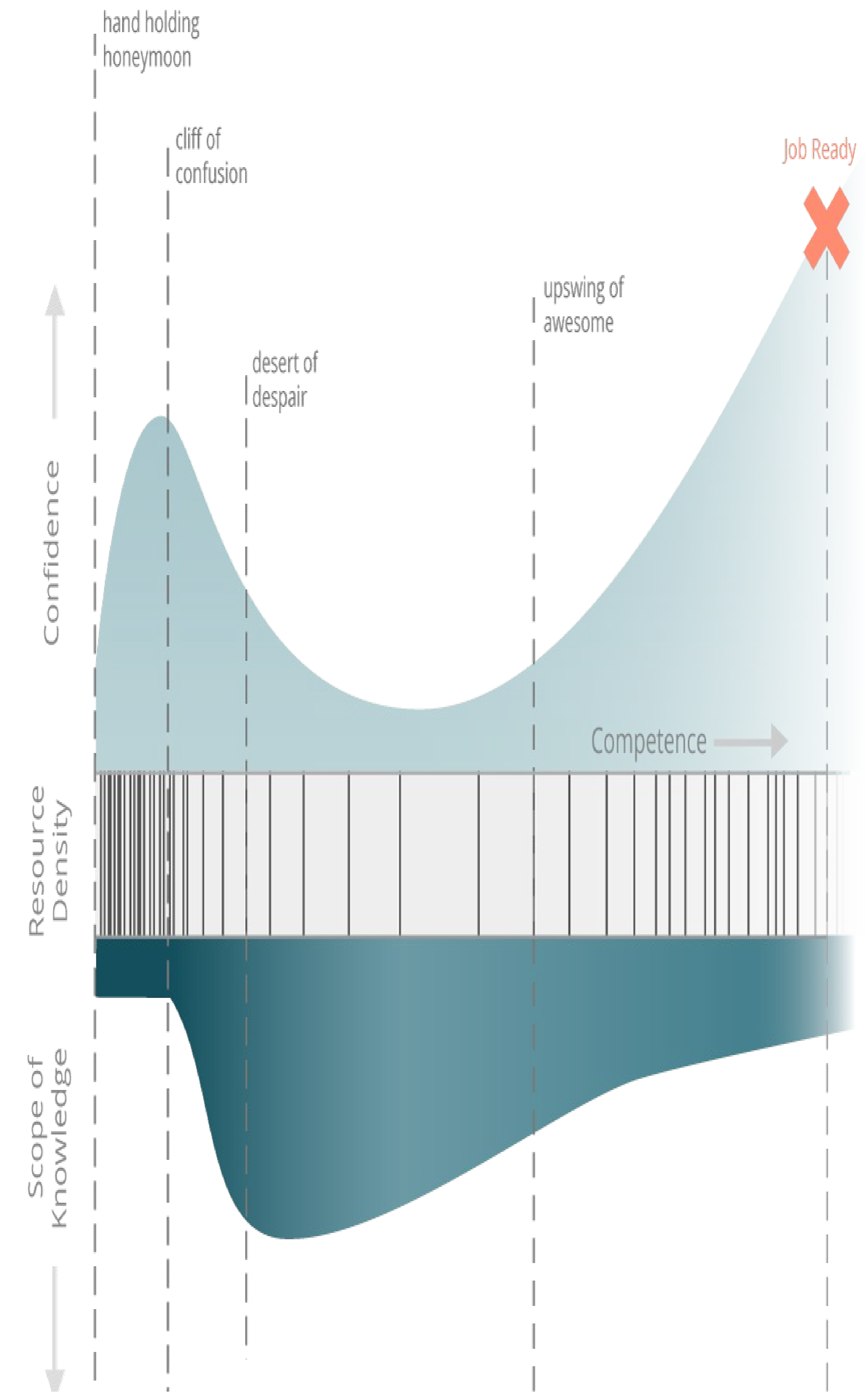
You feel like you should be a developer already but the distance between the code you're writing and a "professional" work environment couldn't feel further away...

Eventually, though, you'll make it. There's too much momentum not to! The Desert of Despair is behind you and the Cliff of Confusion is a distant memory. You're finally, *truly,* on the upswing. You're learning faster and more intelligently than ever before and, eventually, you will have absorbed enough best practices that your swiss cheese knowledge coalesces into a production-grade skill set.

The Upswing of Awesome always takes longer than you expect it to and it feels interminable because you're *so close*... but you will get there. If you're persistent enough in the right ways (the topic of a future post for sure), you will convince someone to pay you to keep learning. The job is yours.

What it All Looks Like

So now you've seen the road ahead and the reasons why it can be difficult. When you combine all four phases we just covered with the factors that define them, it looks something like the following chart:


hand holding honeymoon
cliff of confusion
desert of despair
upswing of awesome
Job Ready
Confidence
Competence
Resource Density
Scope of Knowledge

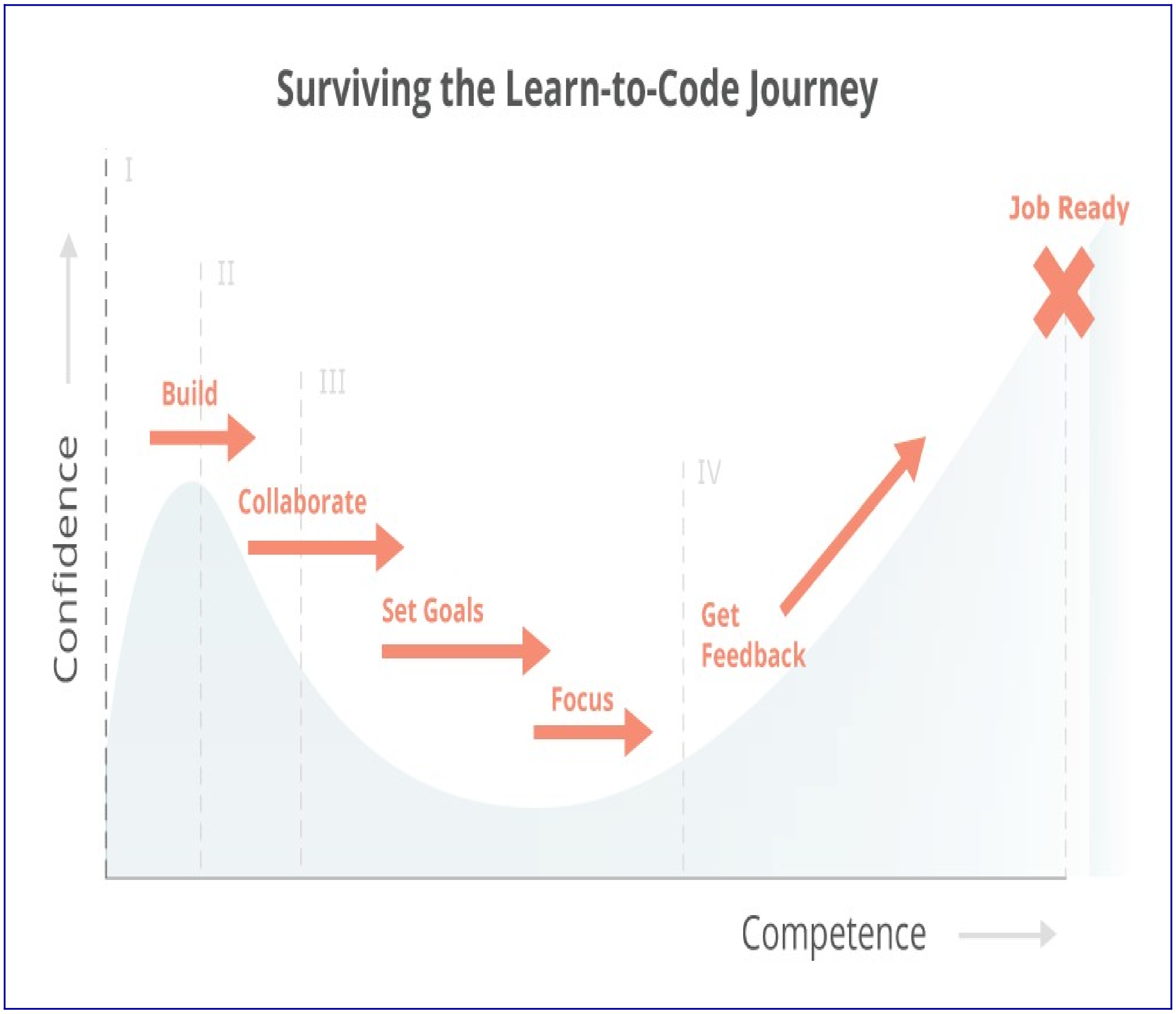
The Whole Shebang -- Click to Enlarge

It's one thing to know the path and another to walk it. Let's get you started on the right foot.

How to Make it Through Alive

The journey seems intense and, frankly, it often is. It's important that you understand what you're in for, particularly if you go it alone. *But you don't have to.*There are ways to short-circuit most of these problems. Learning to code is rarely as easy as people make it out to be but it's also rarely as difficult as it seems in the depths of your despair.

In this section, I'll introduce the key tactics you can use to keep yourself pointed in the right direction.



Your Progression Through the Phases -- Click to Enlarge

I: Surviving the Hand-Holding Honeymoon

The plethora of available resources in the Hand-Holding Honeymoon make it a lot of fun. They do a great job easing you into the kind of logical thinking you'll need to cultivate over the coming phases. It's a great time to start learning to code so try to enjoy it and keep these two tips in mind:

1. **Start by trying out different resources to find how you learn best and what sorts of projects are the most interesting to you. Maybe it's Khan Academy's quick challenges, Codecademy's in-browser exercises, Chris Pine's Learn to Program book or Code School's wacky try Ruby experience. Be open minded at the start and ignore anything about what you *should* learn... all code is the same at this phase.**
2. **Then pick one resource and stick with it once you've found your fit. Work through to the end of their introductory course arc, which should give you all the foundational knowledge you need to write basic scripts and apps. Then get ready to start building on your own.**

II: Surviving the Cliff of Confusion

Almost everyone will experience the Cliff of Confusion because the only way to become a developer is to, well, develop. You can pretend to be building by signing up for tutorials (or tutorials which masquerade as "complete" courses), but you're just putting off the inevitable. Tutorials are a good way to bridge from more high-touch introductory offerings but you'll need to wean yourself off the pacifier and face the real world at some point.

Three tips for making the transition to building on your own:

1. **Work with someone else, even another beginner. You'll be surprised how much easier it is to debug an impossible error when sharing two pairs of eyes.**

2. **Read other people's code to get comfortable with good patterns. Try to understand why the author did what they did. You wouldn't try to become a novelist without reading books as well, would you? We'll focus on this in an upcoming post but, for now, keep your eyes open for any small problems or projects that other people have written solutions for.**
3. **Start small and build constantly. You should have interesting big projects in mind for the future, but you'll need to get comfortable debugging and searching for resources with bite-sized challenges. There's really no substitute for experience.**

III: Surviving the Desert of Despair

Once you've become comfortable debugging, your biggest problem becomes the fire hose of required knowledge and a total loss for how to learn it all... the Desert of Despair. In this case, what you really need is a strong path forward. The Mirages of Mania represent all the interesting side paths and rabbit holes and get-skilled-quick schemes which ultimately waste your time.

So the keys to getting out of the Desert of Despair are:

1. **Have a strong goal for what you want to accomplish because otherwise you will end up chasing your tail learning all kinds of interesting but ultimately unproductive things. If you have the time to spare, by all means skip this...**

2. **Find a strong path which leads directly to the goal you've set and verify that it will actually get you there. This is where you need to dig deeper than the marketing slogans and smiling faces on course websites or book jackets to ask "will this help me accomplish the goal I've set or not?"**
3. **Focus and avoid distractions because, if you're the kind of person who's interested in learning to code, you're also the kind of person who gets interested by learning *all kinds* of other awesome things. When coding gets difficult, you need to be able to push forward instead of just trying out the next cool-looking thing.**

If you're able to identify a path and stick with it, you'll eventually push forward to the next phase instead of spending months or years chasing mirages across the shifting sands of the this desert.

IV: Surviving the Upswing of Awesome

The Upswing of Awesome is one of the trickiest transitions. You can *develop applications* but you really want to *become a web developer*. Getting past this phase and into a job requires you to do three things:

1. **Seek and follow best practices for programming. You need to understand the difference between *a solution* and *the best solution*. Best practices are a major difference between hacking on your own and building production quality code in a real job setting.**

2. Check your assumptions because you've probably skated by with some gaping holes in your knowledge that you didn't even know you had. You need to diagnose and fix these holes.
3. Tackle the unsexy skills that are rarely addressed but highly important for transitioning into a professional setting. This includes things like testing, data modeling, architecture and deployment which are really easy to breeze past but which are totally fundamental to good development.

The key to accomplishing these things and pushing through the Upswing of Awesome is to get feedback. Students who have learned entirely on their own may be productive but rarely have the kind of legible, modular, and maintainable code that makes them attractive in a professional setting. You need to work with other humans who will challenge your assumptions, ask piercing followup questions, and force you to fix the leaks in your bucket of knowledge.

So... Can it be Done?

This all may sound overwhelming but I promise that many others have persevered and survived this journey before you. By understanding the road ahead, you're already in a good spot to take it on with a focused plan and access to the right kind of help.

Obviously there isn't space in this particular post to dig as deeply into each phase of the journey as we'd like or to provide the kind of granular how-to advice you deserve. That said, this is a journey with which we're quite familiar and about which we're highly passionate so we want to help in any way we can.

Our core program is specifically designed to bridge this whole process but, if you're interested in following along on your own, we'll be addressing it publicly and in depth during future blog posts as well.

Sign up below if you'd like to come along for the ride as we dig deeper into everything here -- from finding a mentor to bridging the gap to a fulltime job in web development. Because, though it's a challenging road ahead, you don't have to walk it alone.

Good luck!

Special thanks to Peter DePaulo, Javier Noris, Michael Alexander, Andy Brown, Saul Costa, Phil Nachum and Quincy Larson for sharing your experiences and for helping to debug these thoughts.