The 2018-2019 Guide to

Preparing for Elite Coding Programs

CODESMITH

table of contents

Good engineers know how to code, great engineers have developed problem-solving and debugging skills. Think more creatively about the challenges you are trying to tackle to find efficient solutions.

page 28

Do you know the differences between entry level roles and senior roles? Find out the types of problems you solve in each role.

page 32

Learning to code doesn't only involve your programming knowledge; develop strong technical communication skills to get accepted into a coding program or land that dream job!

page 33

Codesmith: At a Glance

Codesmith at its core is a 12-week advanced JavaScript program, located in Los Angeles and New York City. We aim to not only build excellent coders, but to foster a community of high achieving problem-solvers.

As a center of software engineering excellence, the Codesmith team creates an elite JavaScript program, offering residents innovative lectures on the most complex and modern concepts and a lifelong network of dedicated engineers.

While building a tight knit community of engineers is critical to the Codesmith experience, the success of our graduates speaks to the quality of the projects built in the program and the core curriculum. Codesmith residents go on to work at a variety of companies:, ranging from the largest tech companies to new startups focused on blockchain, crypto, and machine learing.

- Average Salary of New York Graduates: $113k
- Average Salary of Los Angeles Graduates: $105k
- 25% hired into senior developer roles
- ~70% hired into mid level roles

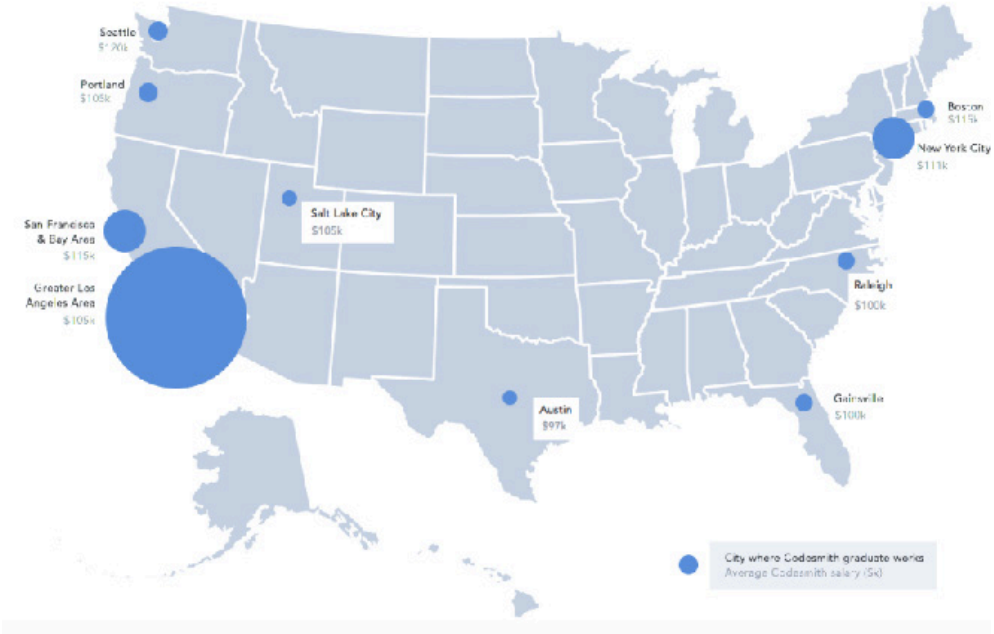
A key part of the success of our graduates is the program's emphasis on project building, most notably the production project.

Open Source Developer Tools Built By Codesmith Residents:

- Reactide: 8000+ Github Stars, 3rd on Top 25 Most Amazing React.js Projects
- React Sight: 2000+ Github Stars, featured on the 2017

Companies where Codesmith grads work

Codesmith grads work at over 120 companies across the USA



Top 25 Most Amazing React.js Projects

- WebDSP: Featured on the main stage at the Google I/O Annual Developer Conference

At the core of this program is the curriculum, teaching all aspects of engineering.

- Front-end development
- Back-end development
- Computer science fundamentals
- New technologies, such as machine learning.
- Technical communication
- Project management

It is no secret getting accepted to Codesmith is challenging, but we are constantly creating new resources to help you along the way.

- CS Prep: An accelerated, remote course, designed to give you a rundown of JavaScript and computer science fundamentals, and valuable interview prep for your technical interviews for coding programs or junior developer roles.
- CSX: Our free, online learning platform with 100's of hours of video lessons, challenges, and more

Codesmith and our committed team are excited to share this insider's guide to help you reach your software engineering goals. Enjoy!

Welcome to the Guide

Created by the Codesmith Team

Why are you pursuing a coding program? Before starting your personalized journey, be sure to think about what motivates you and which kind of program will help you reach your desired outcome.

The coding program space in 2018 can be overwhelming to say the least. There are so many differing opinions and hundreds of resources and programs to sort through. Even once you decide on a language to learn, there are many other factors to consider - location, price, outcomes, hours, difficulty - and the list goes on and on.

This guide will help you specifically map out a preparation plan to a web development program teaching JavaScript (the language most commonly used on the web). For the sake of this guide, we'll be focusing on JavaScript / Web Development programs. Its a great choice; JavaScript has been the most popular language for the past 6 years, and we don't see that slowing down anytime soon. Today, JavaScript is used to create websites, servers, mobile apps, games, desktop apps, and even interactive toasters through the Internet of Things. Focusing on this language sets you up for a wide variety of career opportunities.

We'll help you create a personalized study plan with resources at every step of the journey, so you spend your

time learning to code rather than waste time using and paying for unhelpful platforms.

To develop an appropriate plan for your preparation you must first understand your goals. We recommend taking a moment and thinking deeply about these questions -

(1) Why are you interested in a coding program?

(2) What type of learner are you?

Knowing why you're interested in a coding program is essential for creating a preparation plan. Coding programs result in graduates at incredibly varied levels. Are you aiming for a senior, mid level, or junior development job? If you're shooting for a certain type of software engineering position or industry, you need to make sure the program you are applying for prepares its grads to follow this path. You also need to think deeply about what experiences you want to have during your coming program. Are you motivated by learning the newest technologies, building impactful open source projects, or having face to face interactions with other students every day? Your answers to these questions will point you toward the perfect program for you.

The learning style between programs differs greatly with the most obvious difference being in-person vs. online. We recommend being incredibly honest with yourself if online learning works for you. Additionally, some programs

Codesmith Graduate Spotlight

Daniel K.

Google

"When selecting who will be admitted into the program, they focus not just on the person's technical skill, but on their communication skills, empathy, and culture fit. The 12 week program is very demanding, and the level of support that I felt from the staff and my fellow students was what allowed me to grow as much as I did in the program."

Susan T.

Thrive Market

"What exceeded my expectations: How Codesmith taught us how to learn new topics in general and how to solve any problem, not just the ones we were given. Codesmith prepared us to deal with uncertainty, rapidly pick up new material and have confidence that no matter what challenge we may encounter, we'd be able to push through and solve it."

"We outline what we expect you to know at each level and the best resources to grow from your starting point."

can do a lot more hand holding and can be tutorial-focused, where others really challenge their students to struggle through challenges in order to gain a deeper understanding they can take with them after the program. If you're someone who gains a lot from tutorials versus overcoming blocks individually (less walkthroughs), this should help you shape your decision. You want to select a program to prepare that prioritizes your learning style and will push you to the outcome you want.

Many other questions are important to think through (hours per week, finances, location, etc.) but the most important thing to understand is your motivation and desired goal.

Now that you've thought through your drive a bit more, the next step is making sure you're prepared to succeed at the type of program you're shooting for. We organized our guide into levels of JavaScript experience, so feel free to skip to the level you think applies to your knowledge. In each level we outline what we expect you to know and the best resources to grow, beginning at that starting point.

Completing all the recommended resources doesn't necessarily guarantee you acceptance to the most elite programs, but it will help you put your best foot forward. Advanced programs, like Codesmith, require additional skills beyond pure JavaSrcript knowledge. Be sure to pay close attention to other practices, such as technical communication, which will help you take your understanding to the next level.


acer


DO
MORE.

Level 0: Never Coded

At this early stage in your coding path, we recommend focusing on understanding the fundamentals of writing code. The most important piece is persevering and practicing. Ideally do so with some friends or at a free Codesmith workshop (live-online or in-person). Nearly 50% of Codesmith's accepted students start their journey without a computer science degree or background as a developer and are admitted in under one year. You've got this!

Everyone's journey to a coding program begins somewhere, and never having coded before is a great place to start! At this level of experience you're able to carefully select your preparation path to make sure you're building up to the type of coding program that will put you in the best position for your desired outcome.

When just beginning to learn a language you should focus on the fundamental syntax of a the language, paying close attention to the new vocabulary you will be using to describe aspects of JavaScript. Challenge yourself to not use plain English but to begin developing your internal computer science glossary.

We recommend you start by diving into these resources on their beginner JS tracks -

Codecademy - Intro to JavaScript

The combination of walkthrough tutorials and small challenges to test understanding make Codecademy a great place to start your JavaScript learning.

Code School - JavaScript Road Trip Part 1

Code School combines challenges and videos on the console, variables, js files, and a very beginner introduction to HTML. This resource will help you grasp the larger ideas and how things tie together.

Team Treehouse - JS Basics
Treehouse is a very friendly way to start coding! They'll give you a great backstory on why things are important, with an extensive introduction that's perfect for those just starting out.

In additional to mastering syntax and following tutorials, becoming an 'expert googler' is key to your success as an engineer. A wonderful aspect of the technology industry is the amount of support you can find online. The three resources you should become familiar with are -

W3 Schools
W3 schools is your go to place for tutorials and syntax breakdowns. If you're ever questioning the breakdown of a concept, this should be the first place you visit.

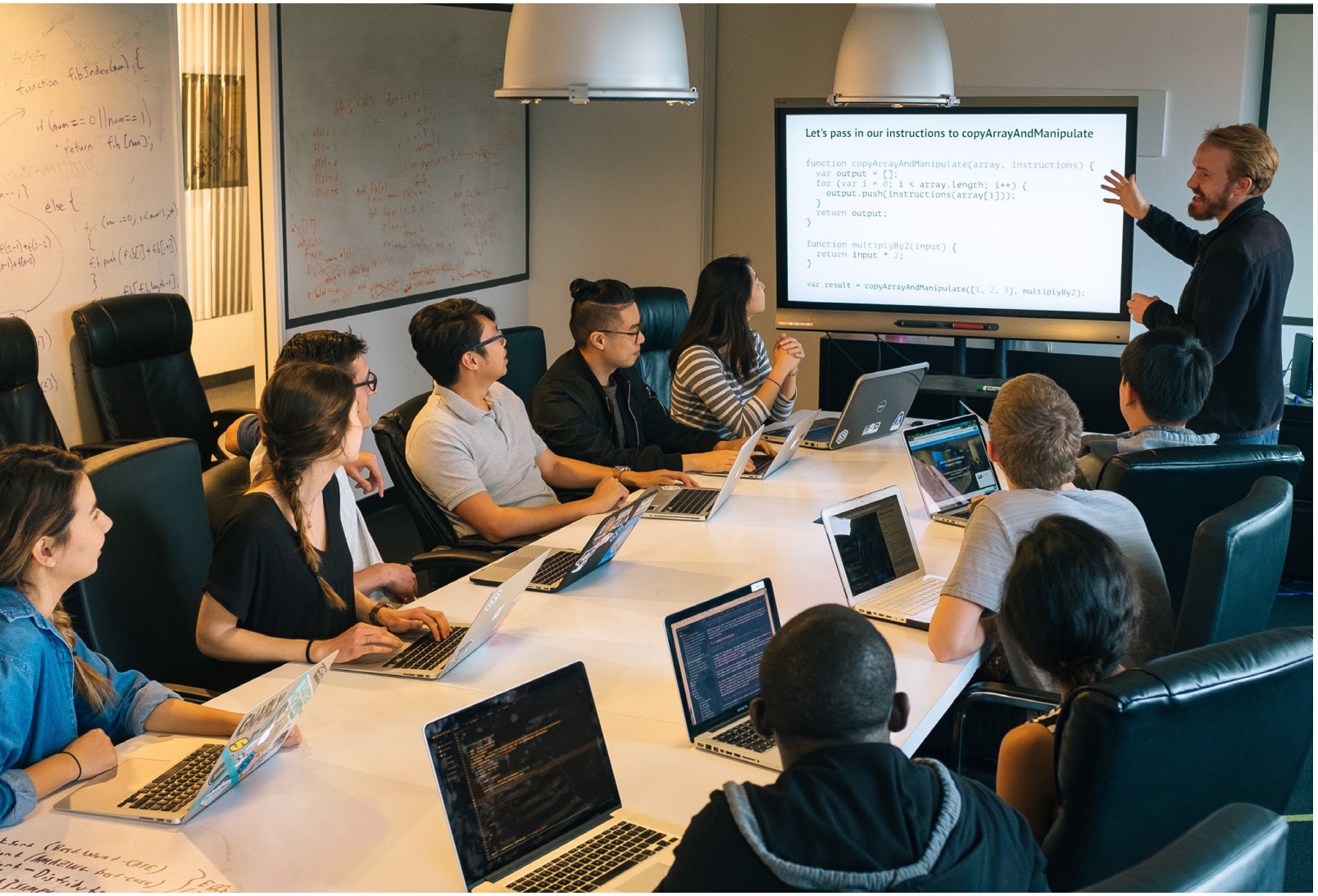
MDN
MDN web docs provide all the information you need to know on JavaScript, HTML, and CSS. They provide great background information, walkthroughs, and tutorials

Stack Overflow
If you have an error you can't seem to crack or something isn't working the way you think it should be, likely someone else has experienced this and asked about it on Stack Overflow. Come for the wealth of help and stay for the sassy comments.

We encourage you to look up every concept you come across on W3 and MDN and read the documentation behind it. This will give you the reasoning behind why these concepts are important and a couple examples to base the concept on. Gathering the background information before moving into challenges is a great way to learn code.

Another important part of learning to code is to surround yourself with others on this same path.

The CSX Slack
An engaging group to find pair programming partners, ask JavaScript questions to mentors, and connect with the supportive Codesmith community. Users chat about all things tech



from around the world!

WelearnJS Slack
Great for connecting with other learners over a variety of topics and resources!

Code Community Slack
One of the largest open source focused slack teams in the world! Great for learning more about the newest projects and hearing news from the community.

Once you have a firm grasp on what JavaScript syntax looks like and where to go for help, you're ready to move onto Level 1!

> **"Another important part of learning to code is to surround yourself with others on this same path."**

Level 1: Beginner

You are learning variables, data types, control flow (conditional statements), arrays, HTML and CSS basics.

At Level 1 you're still making sure that you have a complete grasp on the fundamental JS building blocks. Repetition is key here, and pay close attention to how HTML, CSS, and JavaScript work together to make up much of our modern web. Pushing through these blocks are best done at in-person/online Codesmith workshops where you can ask questions when you come upon them and really push your skills forward.

Having a complete grasp of the fundamental concepts in JavaScript is imperative for your long term growth and that starts with having a deep understanding of variables, data types, conditional statements, and arrays.

The best resources to cement this knowledge are ones that enforce repetition. It may seem silly at first to work through challenges that test these same concepts over and over again, but since these are the building blocks to web development you want to be sure you've got them down, and the best way to do that is through repetition, repetition, repetition.

CSX - Precourse Unit

The Precourse to the Codesmith JavaScript learning platform covers the fundamentals in depth. It's a great way to walk through each concept and test your knowledge with the precourse self assessment at the end. You also have access to mentors in the CSX slack to help you with any questions you may come across.

JavaScript the Easier Parts

This Codesmith workshop covers all primitive data types, mathematical operators and the order of operations, understanding and utilizing variables, and controlling complexity with if/else conditions. It also has pair programming through challenges that test your understanding and ability to communicate on these concepts.

Freecodecamp

Take advantage of the freecodecamp forums for questions and dive right into their intro to JS, HTML, CSS, and a bit of jquery. It uses a friendly in-browser text editor, so is a great place to start. "I really liked how lengthy it was. I was able to continually grow and see my progression over time" - Luis, Codesmith Graduate & Engineering Fellow

Codecademy - Learn HTML

Similar to their JavaScript resources, Codecademy is great for learning HTML. They break things down into easy to manage tutorials,including both instructions and problems to make sure you've grasped the concepts.

Level Up Tutorials - How to make your first website

A great resource for getting your first site online, with no prior experience with HTML or CSS necessary.

"Starts slowly going over the basics of HTML and CSS and then really ramps up and covers everything you'd want it to" -Ben, Codesmith Resident

There's an enormous number of resources out there for JS beginners, and these are just a few of them. Many junior programs will purely test these fundamental concepts on your technical interview,

so be sure to look at what the program you're shooting for requires. If they don't have it listed directly on their website, review the resources they recommend for preparation, and try to attend an info session to have the chance to ask questions like this. It's rare for web development programs to test your HTML and CSS upon entry, but be sure to ask if that's something you need to brush up on.

Once solving problems focused on data types, control flow, arrays, and the HTML CSS basics become second nature you're ready to move on to Level 2: Upper Beginner.

Level 2: Upper Beginner

You are working to master for loops, objects, array and object manipulation, type coercion, functions and execution contexts.

While many Codesmith applicants start attending Codesmith workshops at Level 0 or 1, Level 2 is where you should attend regularly. You've become comfortable with the basics, and you're ready to dive deeper into how JS works and solve more complex problems. At Codesmith, we prioritize applicants showing both how to make something work *and* why it works, so make sure you pay special attention to the "why."

Upper beginner is a stage on your journey to a coding program that can take awhile to move through, and so cut yourself some slack if you're not moving on as fast as you would like to.

" Functions are the most important thing to learn in JavaScript. Every library you work with, whether on the frontend or backend, is completely composed of functions."

- Michael K.

Software Engineer at Leaf Group
Codesmith Graduate

This is because you're working with computer science concepts that will come heavily into play in more advanced JS work, such as for loops, objects, array and object manipulation, type coercion, functions and execution context.

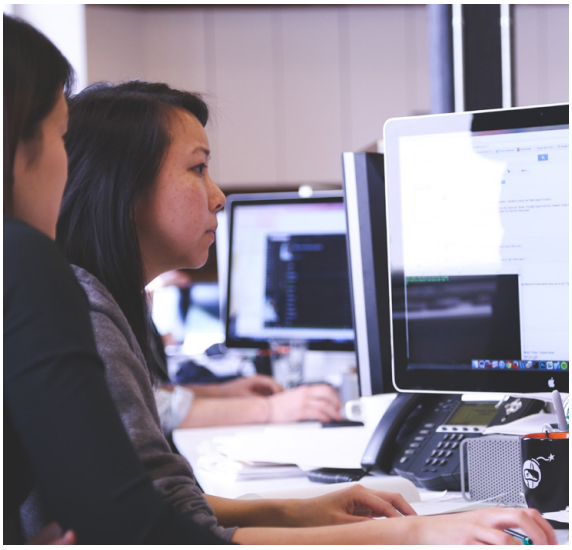
With this set of concepts, you'll want to focus on resources that first introduce you to these concepts and then portray them in unusual ways to really test your understanding. It's easy to fall into the trap of thinking you fully understand a concept when you're only seeing it in one setting

For example, you may understand the basics of for loops, but seeing them used with expressions beyond looping to the end of an array may be foreign to you. You may understand how to iterate through an array, but using Array.forEach or Array.

map isn't in your skill set. You may know the differences between how to loop through an array versus an object, but are yearning for a deeper understanding of why you need to loop through them differently. You might be aware of type coercion, but wondering why it's a thing in the first place and how it could ever be useful.

CSX - Functions & Execution Context Unit

This unit will give you a strong foundation on code execution, memory, and the call stack. With a combination of high quality videos and challenges, CSX is meant to really test your understanding. This is a fantastic, more advanced resource to use after you've been introduced to the concepts to test your understanding here.



JavaScript: The Easier Parts

Complimenting CSX: Functions & Execution Context- JavaScript: the Easier Parts covers the anatomy of for loops, how to understand arrays and objects, nesting loops, and how to create functions. Although you may have attened during Level 1, its common to see repeat students in Codesmith's events. The step-by-step lecture and challenges will help you build the foundation to work through problems on your own with the support of the wonderful community.

JS: The Weird Parts

This 11.5 hour course covers it all but it's especially thorough for those look-

ing for a strong introduction to JS. It's built for people with minimal knowledge and includes both starter and finished source code to help your understanding.
"He breaks it down in a way that assumes nothing, you go in with no background and earn how things work under the hood" - Luis, Codesmith Gradute & Engineering Fellow

Shayhowe - Learn to Code HTML & CSS

If you're looking to start at an introductory level and end with your first web page, this is the best place to do it. Shayhowe does a really good job building you up and leaving you with a greater understanding and a finished product you can be proud of.

JavaScipt30

JS30 by Wes Bos is a fantastic resource for combining basic HTML, CSS, and JavaScript with a focus on building projects.
"Good intro to JS while challenging you to make something every day. Much more exciting than just challenges" - Ben, Codesmith Resident

Watch & Code: Practical JS

Come for the explanations and stay for the detail, showing you how even the most basic parts of JS relate to building full scale applications. If you're lost on how the things you're working with now relate to what senior engineers do spend some time in this course!

Featured Course

CS Prep - A course both flexible & structured

In the Upper Beginner phase, you may start to waver from your dedicated learning schedule and feel you need some structure. Luckily, Codesmith offers a remote, live online course designed to jumpstart your coding journey. CS Prep will help you develop a deep, yet intuitive understanding of JavaScript, covering core concepts and engineering best practices.

Current Codesmith Resident and CS Prep graduate, Adam, says "my goal was to get myself immersed in all areas needed to be able to take the Technical Interview as part of the Codesmith admissions process -- but I really got so much more than that. The other students of this class were exceptional and you could tell they were extremely involved and interested; and they too were learning a lot of this stuff for the very first time as well. [It] makes it so much easier to work with people as they are helping each other along."

Apply to CS Prep to accelerate your learning journey, prepare for important interviews and connect with other engineers.

Level 3: Intermediate

You are learning hoisting, ES5 and above, the call stack, callbacks and higher order functions.

The Codesmith interview will focus on many intermediate concepts, specifically higher order functions because they are such a crucial part of development with JavaScript. The best preparation is to attend JavaScript: the Hard Parts and pair program. There are even candidates who have attended the workshops for 45 weeks! While you may not become that regular, learning by collaborating is the best way to grow as a developer. We've had applicants admitted at this level who show great potential and drive.

Congratulations you've made it past the beginner phase - welcome to Intermediate! We describe the intermediate phase as having a grasp on hoisting, ES5 and above (most importantly is ES6), the call stack, as well as callbacks and higher order functions.

JavaScript truly gains its power when it's used as a functional language and understanding how this works leads gives you access to JavaScript frameworks, libraries, and reactive website really taking your building capabilities to the next level.

Understanding the concept of hoisting is also a huge accomplishment at this stage in your journey. This concept is key in understanding how JavaScript is executing your code - an understanding that will allow you to make smart decisions when structuring

your code and determining things like whether a function declaration or function expression is more appropriate for your use case, or to properly track down errors where variables don't have values when expected them to.

At this point you might also have some knowledge of syntax that's been released post-ES5. What is this 'ES5' business, you ask? ECMAScript is a specification on how a programming language should be written and what kinds of features should be included in it. Most people learning JavaScript these days usually start out with using ES5 (5th edition of JavaScript) syntax - this is where your handy 'function' and 'var' keywords come from. ES6 significantly expands on JavaScript features and introduces new syntax that allow us to change the way we write JavaScript - for the better. ES6 gives us features like arrow functions, new keywords to declare variables (let and const), and even new ways to handle asynchronous activity in JavaScript (Promises). These are just a few features, there are also versions of the language beyond ES6 (ES7, ES8, etc.) that introduce even more useful features - turning JavaScript into a fairly robust language.

For more info on ES6, check out this article.

CSX - Callbacks & Higher Order Functions

Excellent instruction videos go under the hood of why functions like map, reduce and filter are powerful tools and keep our code DRY (don't repeat yourself). Then, test your knowledge of these concepts with in-depth challenges.

JavaScript: The Hard Parts - Callbacks & High Order Functions

This is one of the most popular JS MeetUps in Los Angeles and New York City because of the focus not only on how but also why JavaScript works the way it does. You'll then test the theories with time spent pair programming through challenges with an incredibly supportive community. People come back week after week to the same workshop because they keep picking up new things each time. The

record is 41 times, can you beat it?

CS Prep

CS Prep is a structured program that teaches core JS concepts and engineering best practices. It will help you develop a deep, yet intuitive understanding of JS and prepare you for advanced coding program technical interviews. There are two paths - live online (2 weeks of instruction) and self directed (4 weeks with 1 hour/week mentorship).

"I must admit, I was skeptical about Paired Programming. Mostly because I did not know what it was and did not imagine anything nearly as dynamic as the driver/navigator relationship. I really enjoyed that experience and I learned a lot, even while covering the basic concepts presented in that set of challenges. CS Prep is proving to greatly exceed my expectations."
-Joel, CS Prep Graduate

funfunfunction YouTube - Higher Order Functions

In a great youtube series on functional programming, Mattias Petter Johanasson (MPJ), goes into what higher order functions are with live coding on how to use them.

"Entertaining, more advanced content, He's very personal, breaks it down and shows lots of examples"
-Ben, Codesmith Resident

JavaScript.isSexy

Excellent walkthrough of how to manage different scenarios with callbacks and higher order functions. No video or challenges, but really great descriptions.

"CALLBACKS ARE IMPORTANT FOR UNDERSTANDING EVENT-BASED AND ASYNCHRONOUS PROGRAMMING IN JAVASCRIPT. I USE THESE ALL THE TIME TO TRIGGER CERTAIN ACTIONS ON CLICK EVENTS ON THE FRONT END OR EMITTED FILE EVENTS ON THE BACK END."

- Derek M.

Software Developer at RED Interactive Agency
Codesmith Graduate

Eloquent JavaScript - Chapter 5: Higher Order Functions

Incredibly thorough walkthroughs with tons of detail, especially focused on why functional programming is important. For some this resource can be a bit dry, but others, incredibly informative - worth a shot.

Many more beginner programs will test your understanding on Intermediate concepts for their technical interview. Make sure you're well versed on the importance of high order functions.
resources at every step of the journey, so you spend your time learning to code rather than waste time using and paying for unhelpful platforms.

Codesmith Graduate Spotlight

Brandon D.
Hulu

"I was free to ask for help and work with others on what I had trouble with and at the same time others could always come to me with their questions as well. And that was important because the program was dense and without that kind of community, I couldn't learned what I did, and I wouldn't have been able to built what I did."

Bita D.
UNITE US

"The Codesmith team will go above and beyond to provide the highest quality education and help you find an amazing job with an amazing salary. The hours will be long, but the people you will spend your days with are amazing, both personally and professionally."

Level 4: Upper Intermediate

You are working on closure and basic object-oriented programming in JavaScript, including prototype chain and prototypical inheritance.

The Codesmith interview will test concepts from this level, especially closure. At Codesmith we value compact and readable code which in JavaScript is primarily accomplished through closure. On the Codesmith technical interview you're able to look up anything you'd like - we're as much assessing your debugging skills as your JavaScript skills, so make sure you're comfortable with reading documentation. Codesmith applicants with exceptional problem-solving skills have been accepted at this experience level.

Closure is probably the most misunderstood feature of JavaScript yet one of the most powerful. The concept makes up a core part of the Codesmith interview (and of developer interviews in general).

Closure allows us to understand professional-grade functions like once, memoize and even password-protected functions. Closure is also what enables us to build iterators in Javascript (a brand new ES6 feature) and are really useful for writing asynchronous code - the part of JavaScript that powers web apps.

By introducing the execution context and variable environment, we will be able to understand classic closure functions like once, memoize and even password-protected functions.

CSX - Closure, Scope, & Execution Context Unit

This unit in CSX really pushes you to know closure in depth, while giving you all the tools to get there. You'll come out of it having a full understanding of classic closure functions like once, memoize, and event password-protected functions. Paired with the in-person/online workshop below you're sure to master this difficult concept.

"OOP offers a developer the chance to encapsulate their code and reuse large pieces of functionality throughout the program."

- Xavyr M.
Engineering Fellow
Codesmith Graduate

JavaScript: the Hard Parts - Closure, Scope and Execution Context

In this interactive workshop led by Codesmith CEO and Frontend Masters instructor, Will Sentance, covers these concepts that 80% of JavaScript engineers do not understand but are at the core of every single successful application. Cement your knowledge with pair programming through challenges after you're introduced to why closures function the way they do.

You Don't Know JS: Scope and Closures

Kyle Simpson's book and github repo is a go to place for all sorts of JS concepts, but his bit on closures and scope is especially informative. He'll conversationally walk you through how closures work and why they are important with in an in depth look at a common closure scenarios.

One of the best ways to improve problem-solving is to work through coding challenges. These will also improve your readiness of interviews, where these types of questions are popular during interviews).

Codewars

Codewars is one of the best sites for coding challenges. A really cool feature of Codewars is that you also get to see the best solutions voted on by the community.

HackerRank

HackerRank is known for its tough algorithm questions. Many companies use it for their interviews and it's a great place to practice more advanced ways of problem-solving.
"Great for doing lots of algorithm challenges, and is definitely more advanced which helped me prepare for more advanced coding program interviews" - Alyssa, Codesmith Graduate

LeetCode

Similar to HackerRank, LeetCode is great for preparing for algorithms and technical interview questions. Frequent contests with rewards will keep you pushing your skills to the next level.

"A lot of challenges with more edge cases than Codewars, really good for challenging your mind to think of new ways to view the concepts you're testing yourself on" -Jake, Codesmith Graduate

PythonTutor

PythonTutor lets you visualize your code running line-by-line much in the way Codesmith's JavaScript: The Hard Parts workshops do on the whiteboard.

Chrome DevTools 101: Debugging JavaScript

When you get an error in your code, Chrome Devloper Tools are your friend. In this intro course, you'll learn about inspecting elements, pausing code execution and how to analyze your errors to debug your JavaScript quickly.

JavaScript: the Hard Parts

Utilize the dedicated pair programming time. By working through problems with a partner where you are expected to explain your strategy before coding, you can expect to discover new approaches to prolems, as well as develop a more organized problem-solving approach.

Take it up a notch: Master debugging

At the upper intermediate level, you will have probably had quite some experience debugging your code. This in itself is quite an important skill. Often times our first pass at writing code is not necessarily the best way of doing it, or perhaps our first pass doesn't work at all! This is completely normal - even seasoned developers face this issue on a daily basis. However, **stellar debugging skills are one of a few things that differentiate an experienced developer from a novice.** This is the skill that's responsible for pushing you forward in times of struggle. As soon as you get stuck, you immediately start walking through your application logic, logging values to the console, and carefully examining your data flow. Once you've investigated the issue further, you find yourself looking at documentation on methods or libraries that aren't working quite the way you expected them to - and then coming back to your code with your new found knowledge to implement a fix for what wasn't working. This skill plays a large role in your journey to becoming an autonomous problem solver. Developers who have truly honed this skill can jump into any codebase or tackle any problems that they've never seen before with confidence since **they have a refined workflow around breaking things down and connecting the dots through debugging.**

At this stage you might also be putting a lot more thought into problem-solving strategies. This is the portion of your workflow where you allow the problem **you're faced with to digest a bit, and think of a game plan before moving in any given direction**. Having a structured workflow around coming up with strategies can vastly accelerate the speed at which you move through problems. Before each challenge, you're now asking yourself, "**What does it mean to succeed in solving this problem?**". Once you've answered that question, you then move on to examining your inputs, determining what your outputs should be, and creating actionable pseudo code based on a strategy you've formulated to get from your input to your output. Once you've got your pseudo code written, the actual code flows naturally. This process, once refined, coupled with solid debugging skills and sprinkling in some domain knowledge will make **you a truly autonomous problem solver - a force to be reckoned with.**

Level 5: Advanced

You are learning basic recusion, basic asynchronous JavaScript, and building a strong technical communication skillset.

At level 5, you already have many of the pieces you need for admission to Codesmith. Now, make sure you have refined your technical communication by pair-programming or completing a mock interview with a fellow Codesmith applicant. Complete your application, schedule your interview and best of luck!

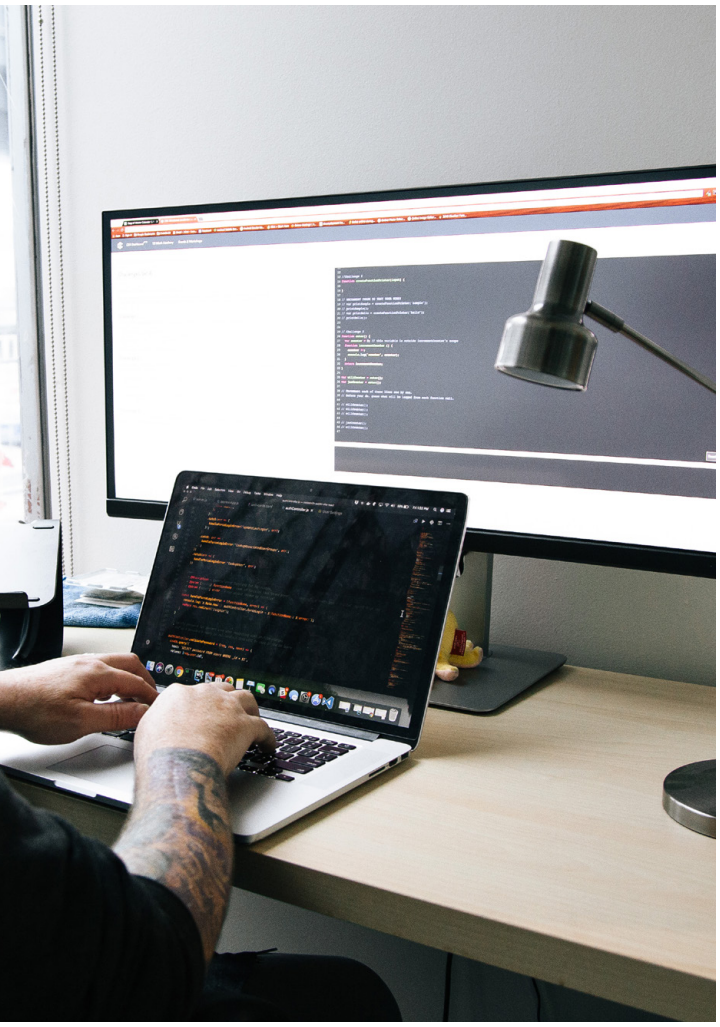
Asynchronous JavaScript is perhaps the most powerful part of web development. It lets us write performant code to speak to a server, to interact with APIs, and many more features that are available to JavaScript.

It's a complex topic covering concepts like the eventloop, callback queue and promises. Codesmith does not interview candidates for the program on asynchronous JavaScript. However, it is a significant part of Week 2 of the program and questions on asynchronous JavaScript are the most typical questions Codesmith graduates receive in mid or senior software engineering interviews, so it's a crucial topic to understand.

"BUILT ON PROMISES FOR TIMING PURPOSES AND IS ONE OF JAVASCRIPT'S STRENGTHS TO NOT HAVING TO HALT CODE EXECUTION TO WAIT FOR AN OPERATION TO FINISH."

- Jeffrey M.

Web Developer at CNN Money
Codesmith Graduate



CSX - Recursion Unit

Recursion is a crucial concept in computer science, and this CSX unit goes incredibly in depth so you can be sure to understand it. With solution videos for the majority of challenges that walk you through the logic, rather than just the solution, this is the best resource to test and grow your recursion knowledge.

Recursion - A Deep Dive

Recursion is definitely a concept best learned with pair programming and the ability to ask clarifying questions. This engaging workshop will start slow and make you grasp the underlying concepts and then test your knowledge with guided challenges through some of the most common technical interview questions on recursion.

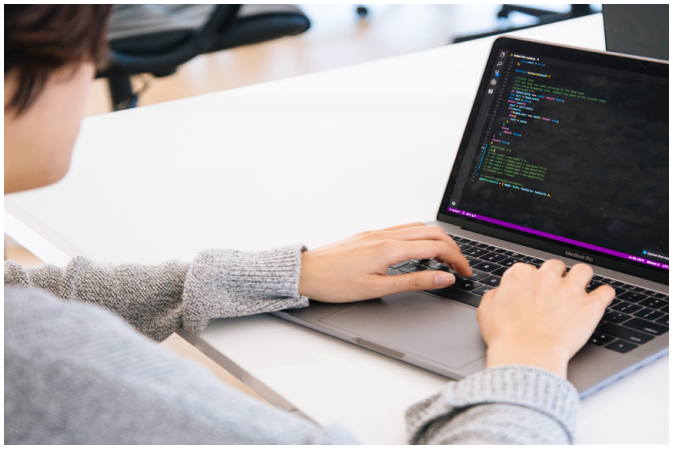
CSX - Asynchronous JavaScript Unit

In this unit, you'll be walked through async, the backbone of web development. You will work with timeouts and practice some of the most crucial async chal-

lenges to understand and to truly grasp the concept.

JavaScript: The Hard Parts - Asynchronous JavaScript
This workshop is the definitive guide to understanding asynchronous Java-Script under the hood - covers the event loop, callback & microtask queues, browser APIs, and promises. In this incredibly descriptive workshop you'll break down async piece by piece so you truly understand the gravity of the concept and its relation to all things web development. You'll leave the workshop being able to tackle challenging async problems with ease.

Jake Archibald's Blog
Microtask Queue
This post is Google Developer Evangelist Jake Archibald's excellent overview of the microtask queue and promises



Choosing the right program for you

Most coding programs can be divided into two major tiers - junior programs and elite programs. Within these tiers, programs are not exact equivalents, but have somewhat similar admission requirements, curriculum structure, and graduate outcomes. You'll need to do the research on each program's specifics (pay special attention to graduate reviews) and decide for yourself which program is right for your path.

Junior Programs

Many beginner and intermediate programs will test your understanding on basic programming concepts for their technical interview. Make sure you're well versed on functions, objects and arrays.

The admissions process expects you to have an understanding of beginner and some intermediate programming concepts including functions, variables, objects and arrays.

These programs will typically give you a foundation in basic web technologies such as HTML, CSS, and web development frameworks such as Angular or React. Intermediate programs may also include more backend technologies such as Ruby on Rails or Node, but will typically not go under-the-hood of concepts or introduce computer science concepts. Students will have the opportunity to build a portfolio of projects in the core web development stack of technologies

Flatiron School is an example of a beginner level pr. Founded in NY and now part of WeWork, Flatiron teaches web development technologies including Angular and Ruby on Rails. The admissions process expects you to have an understanding of beginner and some intermediate programming concepts including functions, variables, objects and arrays.

Elite Programs

Upper Intermediate and advanced programs expect a level of experience with programming - although the admissions process and expectations very significantly by program.

While Codesmith may require the most advanced level of preparation of any program, the admissions process is still designed to value your potential as a software engineer. Expect to include concepts like closure, JavaScript runtime, and spend extra time focused on excellent technical communication in preparation for the Codesmith interview. Checkout resources like JavaScript: the Hard Parts and CSPrep to help prepare.

For programs like Fullstack Academy, App Academy and Hack Reactor, you should expect to understand higher order functions and have strong problem-solving experience.

These types of programs may cover some of the basics but will typically focus on more intermediate to advanced concepts. Hack Reactor and Codesmith, particularly, will also emphasize strong problem-solving training. Some programs also provide a Hiring event at the program - notably Codesmith and Fullstack Academy.

Look out for new technologies being introduced into these programs. Hack Reactor recently introduced Blockchain and Codesmith introduced Machine learning.

Technical Communication

While writing code with proper syntax is necessary, your ability to communicate technical ideas and challenges is critical to becoming a software engineer. Prepare for coding school interviews and job interviews by perfecting your technical communication.

Technical communication is by far the most overlooked skill when it comes to determining success in this field of work. Let's establish one thing before moving forward: great software is not built by great engineers, it's built by great teams. As engineers, we deal with highly technical concepts on a daily basis, and the vocabulary that comes with these concepts is quite bewildering - making it difficult to properly communicate your thoughts to another developer on your team. Additionally, you'll likely have to convey your thoughts to non-technical team members as well. If you can do this effectively, and couple this skill with your technical expertise, you're guaranteed to be a highly valued asset on any team. In fact, Codesmith doesn't accept applicants who cannot demonstrate decent technical communication skills, even if they are a JavaScript wizard. Technical expertise is definitely not the only skill required to be truly successful - you've got to function in team settings as well. To reach this level of communication, you should internalize a few things:

Get the vocabulary right! At Codesmith, we aim for perfect technical clarity in our communication. If we're discussing a piece of code, we'll get into all the relevant details on what's happening when it's run from start to finish - all the while using the exact terminology (ex. function expression vs. function declaration, or parameter vs. argument). When you get to this level, no one who is familiar with the vocabulary will misunderstand you.

Consider your audience. If you're speaking to another developer on your team, they'll likely have an understanding of the tech stack your team uses and the vocabulary that comes with it. How-

ever, if you're speaking to a non-technical team member or even a more junior developer, you may need to explain what's happening in other ways in order for them to create a working mental model of the situation.

Context is everything. Regardless of who you're communicating with, start by probing their understanding of the situation. This allows you to formulate a narrative that gives your audience the complete picture of what's happening. For example, you might be discussing how a certain function behaves, but what the other person might not realize is that this function is part of a larger workflow. Making sure they have all the context they need is key in order to be a great communicator.

CSX - Pair Programming Unit

Pair programming is the best way to practice your technical communication. As the Navigator, you're required to walk your partner through your ideas methodically, with an emphasis on explaining each step. This focus on accurate and descriptive communication can't be reached as a solo coder, and does wonders for your ability to explain concepts and walk others through your ideas.

The best way to get pairing practice is to attend a Codesmith workshop in person (or online if you're unable to attend in person). Each workshop is a mix of lecture and time spent pairing with mentors you can ask questions to.

Pairing will push you much further in the depth of understanding you have on a concept than just coding yourself would. You have to have clear reasoning on why you're instructing your partner what to implement, and to be able to back that up with conceptual evidence

Self Recording

If you don't have access to others in your learning journey or want to spend extra time outside of pairing to practice your technical communication, recording yourself is a good way to do this.

In a sense you're taking on the Navigator and Driver roles when you're doing a recording. You should challenge yourself to sudo code and lay out the approach you are going to implement to solve the problem. You should then implement your code slowly and talk aloud line by line what you're doing and why you're doing it that way.

The best way to grow from this exercise is to play back your recording and analyze your performance. Were there times you were unclear? Were you using the technical term rather than plain English? Did you skip a step in your explana-

tion?

Asking yourself these questions and rerecording after giving yourself feedback is an awesome way to work on your technical communication.

Teaching & Mentoring

Now that you've got some experience under your belt, you now have the opportunity to teach and mentor others with less experience than you. This is an extremely powerful learning tool. Hang on... learning? Aren't we talking about teaching? The answer is yes, to both! Often times, you believe you understand something... until you try to explain it to someone else. Before you know it, you've caught yourself in a nonsensical rant. Teaching and mentoring others is a great way to sharpen not only your technical communication, but also your understanding of technical concepts.

At Codesmith, this is as much a responsibility as an opportunity. Remember, great software is built by great teams, and there's no better way to foster great teams than by establishing a culture of mentorship and collaboration. Every resident accepted to Codesmith's immersive program shows great promise in this aspect - raising the level of all other residents in the program and really bringing out the best version of themselves in the process.

Beyond Coding Programs

So you finished your coding program, now what? There are several routes to persue post-graduation, so be sure to educate yourself and find a role that aligns with your skillset and your personality.

An entry level developer typically has the foundations of good problem-solving skills or may have built some projects in the web development stack but is not yet able to take on a full feature and will typically work on smaller basic tasks including HTML/CSS and will be focused on learning the core technologies of the team.

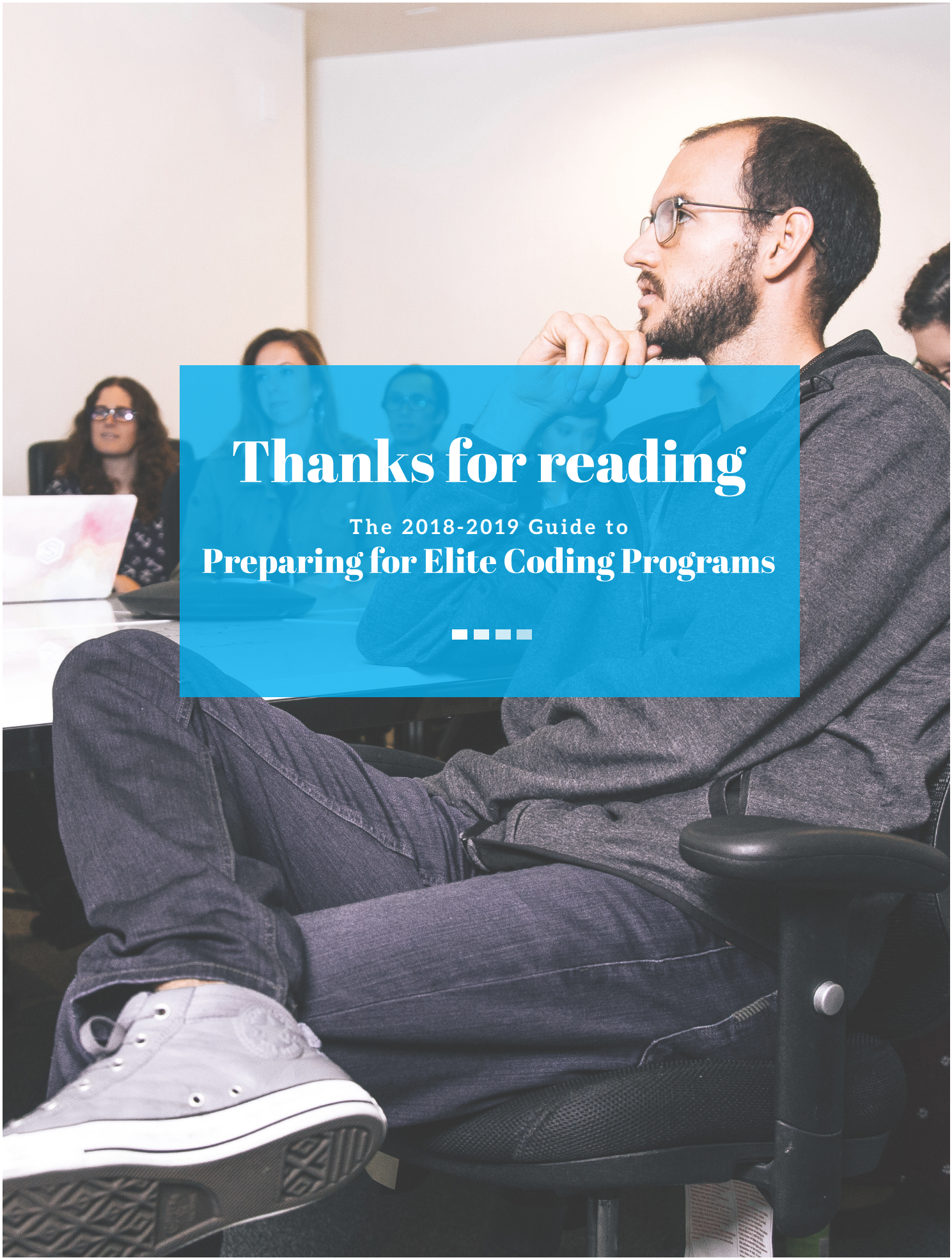
A junior developer is able to work on features in areas of technology that they have directly worked with before. They may not have yet developed the under the hood principles, but have good knowledge of how to implement some web technologies. They may not yet have developed computer science foundations to write more performant or efficient code. Around 5% of Codesmith graduates receive offers as junior developers

A mid-level developer is able to take any new feature or task and implement a solution to it even they have not seen the technology before. They have *autonomous* problem-solving skills and

a knowledge base on the core web technologies under the hood, important computer science concepts and stack of web development like React and Node. Around 70% of Codesmith graduates receive offers as mid-level developers.

A senior-level developer is able to take any new feature or task and empower a team to implement it through clear, empathetic and high quality technical communication. Additionally, senior developers may typically have greater responsibility for larger technical decisions around architecture or technologies chosen. Around 25% of Codesmith graduates receive offers as senior developers

Lead engineers are responsible for the overall architectural direction of a platform including judgments on technologies and managing challenges such a technical debt (the burden of managing and upgrading code written at an earlier date). Lead engineers not only manage architecture but also lead mentorship, ongoing training and dynamics of the team.



Thanks for reading

The 2018-2019 Guide to

Preparing for Elite Coding Programs