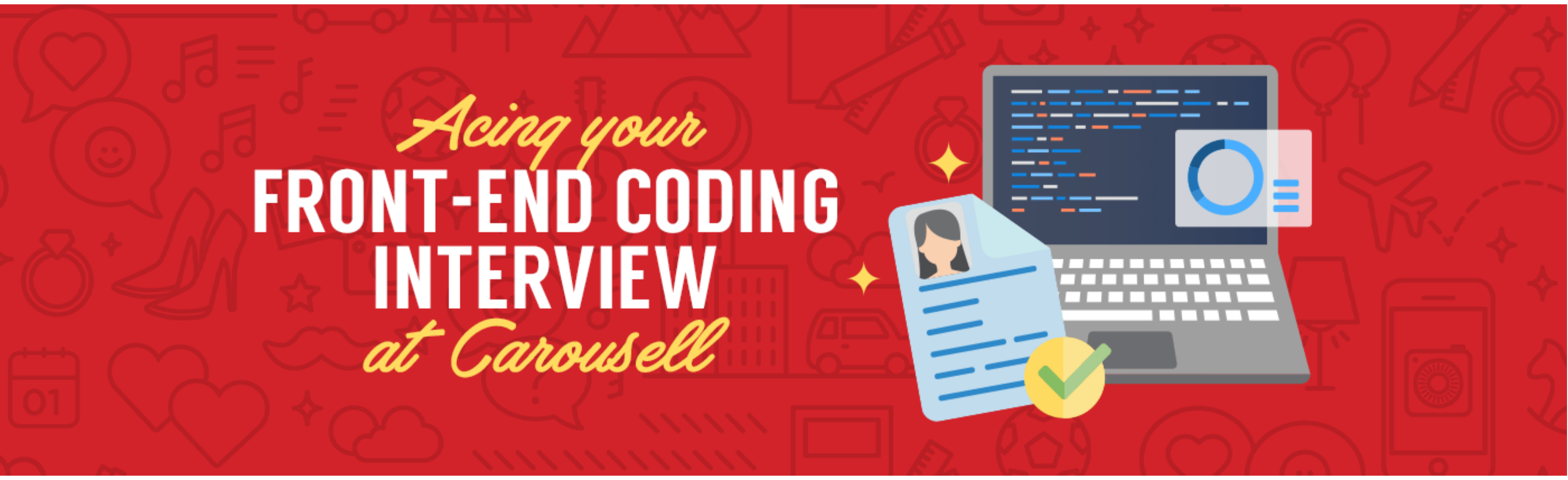


Acing Your Front-End Coding Interview at Carousell

Yao-Hui Chua Follow
Aug 7, 2019 · 9 min read

By Yao-Hui Chua

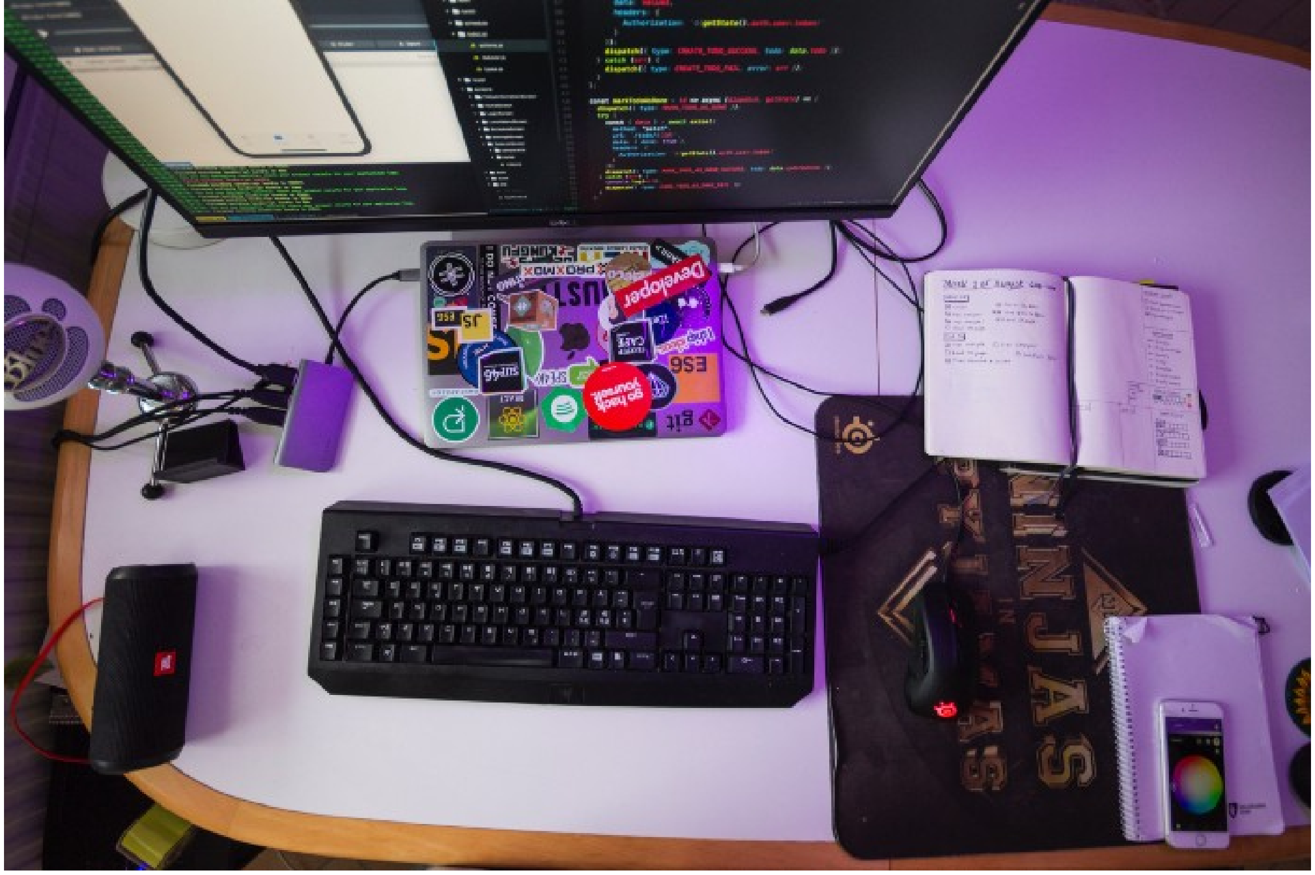
At Carousell, we've made it our mission to establish a world-class engineering team based in Asia.

However, it is tricky to design an interview process that evaluates engineers fairly, regardless of their history. The current state of the web is tremendous in scope; candidates who come to interview with us all too often have different kinds of expertise.

Thus, some of us from Carousell's front-end web team have spent the last couple of months rethinking our company's interview process. In this article, we'll outline our updated expectations regarding what the core technical requirements for front-end engineers should be.

If you're a prospective candidate, our hope is that the following information serves as a useful study guide and steers you towards success during your live-coding interviews with us.

· · ·


A front-end developer’s workspace (from Oscar Yildiz).

An Overview

As a candidate, you’ll want to:

- Master your JavaScript fundamentals.
- Be comfortable working with commonly-used Web APIs.
- Be able to solve algorithmic problems of moderate complexity.
- Have a deep understanding of your own preferred set of front-end tools.

At face value, each of these requirements will seem fairly generic. We’ll carefully deconstruct what they mean in the sections below.

· · ·

Doubling Down on JavaScript

First and foremost, we expect our candidates to have a strong command of JavaScript. Given that JavaScript *is* the language of the web, this is pretty much a given, but there are many finer details worth highlighting.

For instance, we believe that candidates shouldn't need to worry about committing arbitrary prototype methods to memory. It's normal to us if you can't immediately recall the exact signature of a `.reduce()` callback. In fact, we're perfectly fine with you taking a couple of minutes to look up MDN if you ever trip up on these details during a live-coding session.

What we do care more about though, is your knowledge of JavaScript's core mechanisms. Specifically, you should strive to:

- Have confidence in answering questions related to asynchronicity, prototypes, context objects (`this`), scopes, and closures.
- Have a high-level understanding of components in the JavaScript engine that are involved in code execution (e.g. the call stack, the event loop, task queues).
- Be familiar with popular ES6/ES7 features and the abstractions they provide over preceding versions of JavaScript.

Jake Archibald speaks about the event loop, a critical piece of the runtime puzzle.

These qualities are important to us because they ultimately determine how you think your application code will run. Your knowledge of object binding might affect how you decide if arrow functions are necessary in your class-based components, while your knowledge of asynchronicity, on the other hand, could determine how you expect native functions such as `.setTimeout()` to behave.

In order to assess your knowledge of JavaScript, our live-coding rounds will require you to implement a commonly-used pattern or function. Examples include:

> *Implement a debouncer.*

Sample solution here .

> *Implement a rate-limiter.*

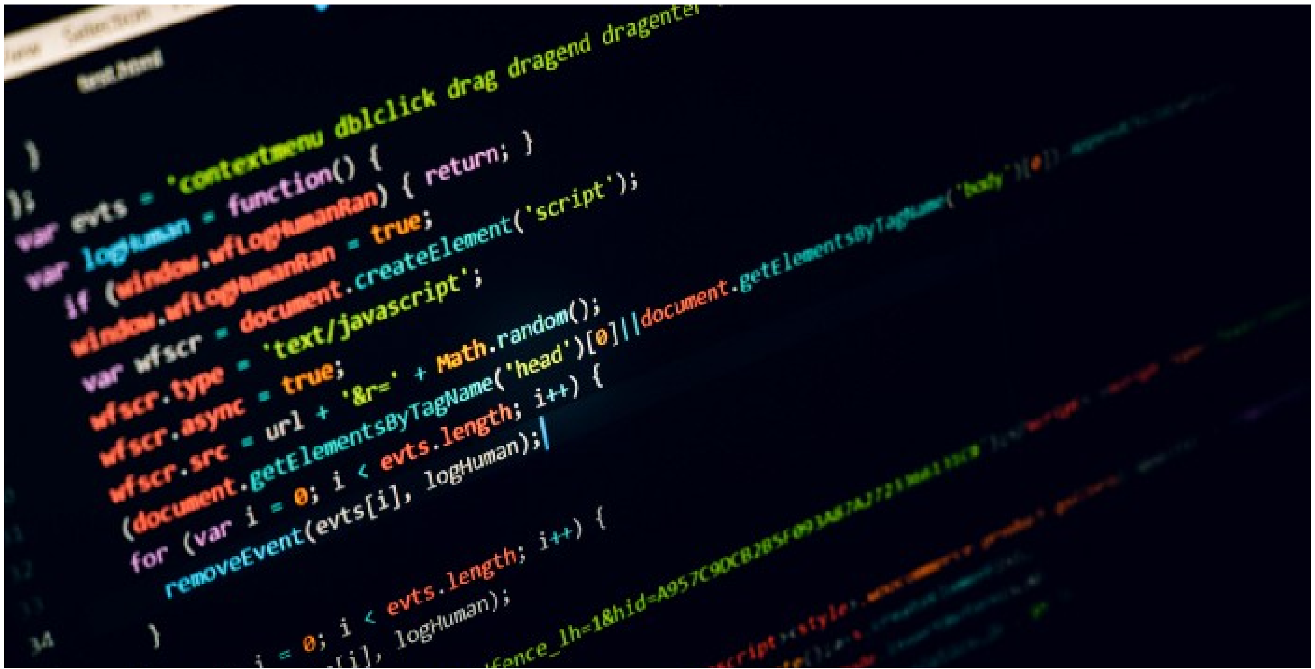
Sample solution here.

You'll need to have a decent grasp of closures and higher-order functions to tackle the above problems. Any reasonable working solution is welcome; what matters most is your ability to communicate your approach well and demonstrate your fluency in the language.

If you're not confident in any of the topics highlighted above, we recommend taking the time to go through resources like Eloquent JavaScript and You Don't Know JS. One great way to test yourself quickly is to try your hand at some of the questions contained in the Front-End Interview Handbook.

While reviewing these materials, it'll also be a good idea to make a habit out of running snippets of code in REPL environments to see some of the nuances of the language in effect.

• • •



DOM APIs in action (from Dlanor S).

A Renewed Focus on Web APIs

JavaScript was created so that developers could add interactivity to their HTML. Frameworks such as React, Angular and Vue provide us with powerful abstractions over the DOM and allow us to quickly compose interactive UIs with ease. One obvious downside, however, is that the convenience afforded by these abstractions can cause engineers to lose touch with the underlying browser APIs.

Framework and patterns will come and go, but front-end engineers will always be working with the browser. Thus, at Carousell, we expect our candidates to be familiar with the native features provided by the platform. After all, these APIs are the very same APIs used by some of the most popular front-end frameworks of today.

In our view, any front-end candidate should:

- Have the ability to select, navigate, and manipulate DOM elements without the assistance of abstractions such as jQuery.
- Have a sound understanding of how browser events can be handled.
- Recognise the functional and semantic roles played by different HTML elements.
- Understand the relevance of concepts such as inheritance and specificity in the context of styling elements.
- Know how to retrieve and display payloads served by backend APIs.

To test your abilities, we'll ask you to build a simple UI feature right before our eyes.

> *Implement a carousel.*

Your mission, should you choose to accept it, is to make this widget work.

For such questions, we'll provide most of the boilerplate HTML/CSS so that you can get productive quickly. Your main concern should be to update the code so that the interactive parts of the feature work as expected.

To reiterate, we don't encourage deliberate memorisation, since we allow candidates to look up documentation sites as required during their interviews. One resource we recommend for getting well-acquainted with the nuts and bolts of browser events is this comprehensive set of tutorials on JavaScript.info.

You'll soon realise that this is a domain that demands a fair bit of experimentation. Taking the time to build and play with widgets in lightweight environments such as CodePen or CodeSandbox will go a long way in reinforcing your learning.

• • •

A climbing net… or an undirected graph (from Clint Adair).

A Continued Emphasis on Algorithms

The question of whether or not front-end engineers need to be good at working with data structures and algorithms is a slightly contentious one. Our answer is **yes**, but it comes with several caveats.

We don't believe there's any real need for our candidates to be able to conjure sudoku solvers or LRU caches on the spot. We don't believe in setting our candidates up for failure and placing them under absurd time constraints. That said, we do think that our candidates should be able to:

- Understand and use typical approaches to algorithmic problems, such as memoisation, recursion, binary search, and dynamic programming.
- Be comfortable with common data structures, such as stacks, queues, and hash maps.
- Estimate space and runtime complexities with good accuracy.

Difficulty-wise, you can anticipate questions that hover around the level of Product of Array Except Self and Letter Combinations of a Phone Number. If you've made a serious effort to brush up on your computer science fundamentals, you really shouldn't have too much trouble with this segment.

. . .

A Broader Definition of Front-End Mastery

We take a good look at every candidate's resume and include in our evaluation process. Towards the tail end of every live-coding interview, we give candidates a chance to talk to share about their past experiences as feature developers and platform contributors. It matters to us that we give candidates the opportunity to represent themselves uniquely for the work that they've done, and we believe it matters to them too.

We think so too.

While we think it's important for candidates to have strong fundamentals, we don't expect you to be familiar with every single framework or library out there. On the job, we spend plenty of time with:

- **UI frameworks and libraries** (React, CSS Modules, Storybook) to create and organise our own set of design system components.
- **Data layer abstractions** (Redux, Redux-Saga, Reselect) to fetch and coalesce various data entities together.
- **Type systems** (Flow) that help mitigate a lot of the pain points of working with a dynamically-typed language.
- **Build systems** (Webpack) which enable us to implement performance practices such as code-splitting and lazy-loading.
- **Testing frameworks and utilities** (Jest, Enzyme) that verify if our application's data flows and UI changes unfold as expected.

Do we assume that you'll be familiar with the exact same set of tools? Definitely not. When we probe candidates about their previous experiences, we try to ask more general questions that might eventually develop into deep discussions about their own areas of expertise. Topics might include:

- **On architecture:** If you've worked with Progressive Web Apps in production, how do you determine the kind of network caching policy to use for your requests? If you've implemented server-side rendering before, what kinds of challenges have you faced with client-side hydration?
- **On performance:** What practices do you abide by in order to reduce page load times? How do you automate the tracking of your app's performance?

- **On build systems:** How have you configured or chosen your bundler to support your app's custom needs?
- **On design systems:** What do you think are some good practices when it comes to organising the different types of UI components in your app? How do you avoid the feared *apropcalypse*?
- **On web infrastructure:** How do you monitor and respond to rises and drops in error rates? How do you detect memory leaks in your client application and on your servers?

You don't need to be prepared to give a lengthy response to all of these questions. At the end of the day, we're just hoping to see your passion for the web ecosystem shine through — and that this passion is anchored by careful deliberation over the tools you've worked with.

Our colleague, Yishu See, reflects on how we've approached component design.

. . .

A Few Caveats

Note that while we're attracted to candidates with strong technical foundations, it's also important to us that you speak with clarity and listen with the intent to understand. Every interview is a two-way conversation; an eagerness to share your ideas and accept constructive criticism signals to us that you'd be a pleasure to work with.

To be clear, our interview format is far from being completely set in stone. Hiring outcomes aren't just about conversion rates, we'll also have to measure how well our hires actually perform on the job.

In addition, you might have noticed that our requirements thus far lean more heavily towards JavaScript/HTML than CSS. In future iterations, we'll work to find a better balance between the different aspects of front-end development.

It also goes without saying that the requirements of the position will evolve along with the shifting demands of the industry. For many, a solid grasp of the networking stack and a good understanding of the browser's internal architecture are considered must-haves for any modern front-end engineer. We'll continue to review such considerations on a frequent basis.

. . .

Up for an on-site?

Wrapping Up

We'd like to stress that *any* developer who has taken the time to prepare themselves in the domains described above is very likely to excel as a candidate. Our goal, ever since we started redesigning our interview process, has been to make our expectations as transparent as possible so that aspiring applicants have a clear path to success.

If you’re scheduled to interview with us, we’d like to wish you all the best and we look forward to meeting you!

If you’re not, then you might be interested to know that Carousell is hiring.

Many thanks to my colleagues from the front-end web team for accommodating multiple back and forths on this subject. Aaron Lam, in particular, worked closely with me on this project. Another shout-out is owed to our Head of Recruitment, Charlotte Lee, who provided Aaron and I with the structural support we needed to make this revamped process a reality.

Kudos to Natalie Tan, Wei Jie Koh, Bang Hui Lim, Yishu See and Stacey Tay for reviewing draft versions of this piece and sharing their comments.

Thanks to Charlotte Lee, Stacey Tay, and Natalie Christian Tan.

JavaScript Front End Development Coding Interview Engineering