Preparing for a Front-End Web Development Interview in 2017

Published on January 10, 2017

David Shariff | Follow
Engineering Manager @ Amazon Prime Now - I'm Hiring!

387 25 71

I've been interviewing a lot of web developers and software engineers focusing on front-end development for the last few years at Amazon and Yahoo, in this post I wanted to share some tips to help folks better prepare.

Disclaimer – *this article isn't intended to be an exhaustive list of topics you could be asked during your front-end interview, but can be thought of as a baseline of knowledge.*

Interviews are hard, and as a candidate, you are typically given 45-minutes to show what you can do. As an interviewer, this is equally as hard to assess if the person is a good fit in such a short amount of time. There are no one-size fits all for interviews, interviewers are typically given an area to cover but apart from that the questions they ask are at their own discretion.

From being on both sides of the table as an interviewee to interviewer myself, this post tries to cover the most important areas of front-end development you're likely to be asked.

Common Misconceptions

One of the biggest mistakes I see candidates make is preparing for trivia style questions such as "*What is the box model?*" or "*Tell me the difference between == and === in JavaScript?*". While knowing the answers to such trivia type questions is great, it doesn't really tell the interviewer much.

Instead, what you can expect is a more hands-on interview where you'll write code using JavaScript, CSS and HTML. During your interview expect to implement UIs, build a widget or write commonly used utility functions from a library such as Lodash and Underscore.js. For example:

- Build the layout and interactions of common web applications, such as the Netflix browser site.
- Implement widgets like a date picker, carousel or e-commerce cart.

Speaking of libraries, another mistake I see is folks needing to solely depend on the latest framework to solve the interview questions asked. You may be thinking why reinvent the wheel if I can use jQuery, React, Angular et al in production, why can't I use it during an interview? Well, technologies, frameworks, and libraries change with time – I'm more interested to see you know the underlying fundamentals of front-end development than needing to depend on higher level abstractions. If you can't answer interview questions without them, I'd expect you could at least thoroughly explain and reason about what the library is doing under the hood.

Overall, you should expect the majority of your interviews to be very hands-on coding and practical.

JavaScript

You need to know JavaScript and you need to know it inside out. The more senior level you are interviewing for the higher the expectations are in terms of depth of knowledge in the language. At a minimum, the following are topics within JavaScript you should know well:

- Execution context, especially lexical scope and closures.
- Hoisting, function & block scoping and function expressions & declarations.
- Binding – specifically call, bind, apply and lexical this.
- Object prototypes, constructors and mixins.
- Composition and high order functions.
- Event delegation and bubbling.
- Type Coercion using typeof, instanceof and Object.prototype.toString.
- Handling asynchronous calls with callbacks, promises, await and async.
- When to use function declarations and expressions.

DOM

How to traverse and manipulate the DOM is important, and this is where most candidates struggle if they have been depending on jQuery or have been writing a lot of React & Angular type apps recently. You might not do this on a daily basis since most of us are using an abstraction of sorts, but without using a library you should know how to do the following:

document.getElementsByClassName.

- Traversal up and down – Node.parentNode, Node.firstChild, Node.lastChild and Node.childNodes.
- Traversal left and right – Node.previousSibling and Node.nextSibling.
- Manipulation – add, remove, copy, and create nodes in the DOM tree. You should know operations such as how to change the text content of a node and toggle, remove or add a CSS classname.
- Performance – touching the DOM can be expensive when you have many nodes, you should at least know about document fragments and node caching.

CSS

At a minimum, you would be expected to know how to layout elements on a page, how to target elements using child or direct descendant selectors and when to use classes vs IDs.

- Layout – sitting elements next to each other and how to place elements in two columns vs three columns.
- Responsive design – changing an element's dimensions based on the browser width size.
- Adaptive design – changing an element's dimensions based on specific break points.
- Specificity – how to calculate a selector's specificity and how the cascade affects attributes.
- Appropriate namespacing and naming of classnames.

HTML

Knowing which HTML tags that best represent the content you are displaying and associated attributes should be back of the hand knowledge.

- Semantic markup.
- Tag attributes, such as disabled, async, defer and when to use data-*.
- Knowing how to declare your doctype (most people are not writing new pages every day and forget this) and what meta tags are available to use.
- Accessibility concerns, for example, making sure an input checkbox has a larger responding area (use label "for"). Also, role="button", role="presentation", etc.

System design interviews for folks working on the backend typically involve MapReduce, designing distributed key-value stores or require knowledge of CAP theorem and the likes. While your everyday front-end engineer shouldn't need to have in-depth knowledge of how to design such systems, you shouldn't be surprised when asked to design the front end architecture of common applications. These questions are usually vague, along the lines of "design a site like Pinterest" or "tell me how you would build a shopping checkout service?". Below are areas to think about:

- Rendering – client-side (CSR), server-side (SSR) and universal rendering.
- Layout – if you're designing a system used by multiple development teams, you need to think about building components and if you require teams to follow a consist markup to use said components.
- State management such as choosing between unidirectional data flow or two-way data binding. You should also think about if your design will follow a passive or reactive programming model, and how components related to each other for example Foo–> Bar or Foo –>Bar.
- Async flow – your components may need to communicate in real-time with the server. The design you propose should consider XHR vs bidirectional calls. If your interviewer asks you to support older browsers, your design will need to choose between hidden iFrames, script tags or XHR for messaging. If not, you could propose using websockets or you might decide server-sent events (SSE) are better.
- Separation of concerns – Model-View-Controller (MVC), Model-View-ViewModel (MVVM) and Model-View-Presenter (MVP) patterns.
- Multi-device support – Will your design use the same implementation for the web, mobile web, and hybrid apps or will they be separate implementations? If you were building a site like Pinterest, you might consider three columns on the web but only one column on mobile devices. How would your design handle this?
- Asset delivery – In large applications, it's not uncommon to have independent teams owning their own codebases. These different codebases probably have dependencies on each other and each usually has their own pipeline to release changes to production. Your design should consider how assets are built with dependencies (code splitting), tested (unit and integration tests) and deployed. You should also think about how you will vend assets through a CDN or inline them to reduce network latency.

Front end system design is a large topic that deserves more attention and I plan to write another blog post dedicated to it.

Web Performance

look at your code or design and its performance implications. It used to be enough to put CSS at the top of a document and JS scripts at the bottom of a page but the web is moving fast and you should be familiar with the complexities in this space.

- Critical rendering path.
- Service workers.
- Image optimizations.
- Lazy loading and bundle splitting.
- General implications of HTTP/2 and server-push.
- When to prefetch and preload resources.
- Reduce browser reflows and when to promote an element to the GPU.
- Differences between the browser layout, compositing and painting.

Data Structures & Algorithms

This is probably controversial but having a basic understanding of Big-O time complexities and common runtimes such as O(N) and O(N Log N) won't hurt you.

Single page apps are more common now and understanding things like memory management helps. For example, if you were asked to build a client-side spell checker, knowing common data structures and algorithms is going to make your life a lot easier.

I'm not advocating you need a CS degree, but the industry has moved on from building simple web pages. There are a lot of resources online where you can pick up the basics fairly quickly.

General Web Knowledge

You will be expected to have a grasp of the technologies and paradigms that make up the web.

- HTTP requests – GET and POST along with associated headers such as Cache-Control, ETag, Status Codes, and Transfer-Encoding.
- REST vs RPC.
- Security – when to use JSONP, CORs, and iFrame policies.

Summary

be phased by the depth of knowledge needed but keep an open mind to learning all the intricate pieces that make up the final end product your customers will use.

In addition to the technical topics covered here, you will be expected to talk about your past projects, describe interesting situations and the trade-offs you made.

I'm sure I've missed many areas of a front-end interview, I'd love to hear your experiences or let me know if I've missed anything of importance that you see being asked these types of interviews.

This article was cross-posted, please follow @davidshariff on Twitter for more posts. Note - views in this article are my own and do not represent those of my employer.

Thanks to ShihChih Huang of Facebook and Preethi Kasireddy of Coinbase for reviewing this post before publishing.

Report this

David Shariff
Engineering Manager @ Amazon Prime Now - I'm Hiring!
1 article
Follow

25 comments Newest

Leave your thoughts here…

Sheila Capps ··· 8mo
Program Manager & Trainer - passionate about people, inclusion, and great product
Daniel Capps
Like Reply

Joe H. ··· 8mo
Senior Software Engineer
And there's already talk of using ES7, when only 45% of ES6 features are supported in browsers. SMH.
Like Reply

There are 23 other comments. Show more.

Top stories from Editors' Picks