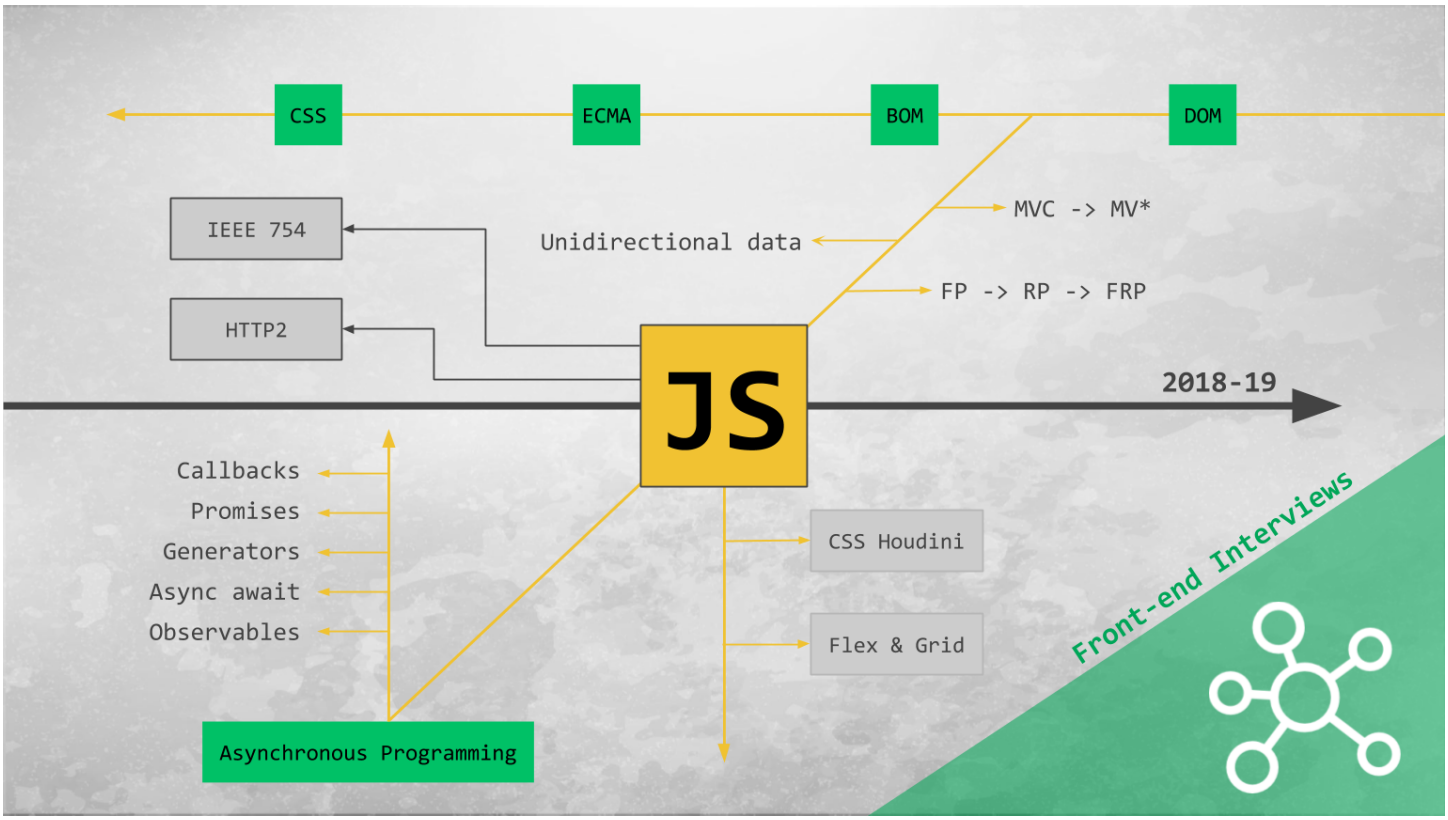
Front-end JavaScript Interviews in 2018–19

Harshal Patil

Nov 12, 2018 · 10 min read ★

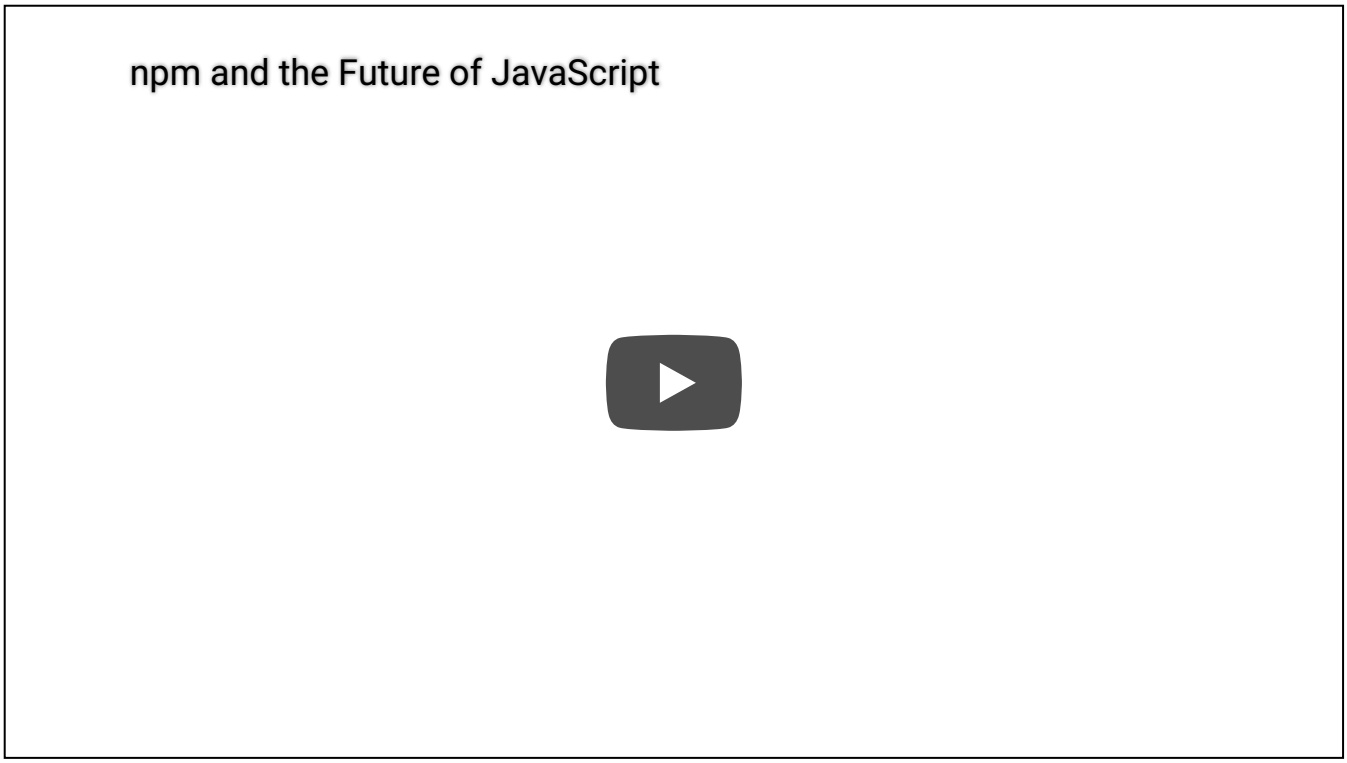


Front-end JavaScript interviews in 2018–19

Things changed with ES2015 for JavaScript. The specification is huge. With native support for modules coupled with ever-evolving functional programming patterns, new JavaScript entirely feels a different breed of a language. And the juggernaut continues every year with ES2016, ES2017…

Yet when it comes to interviewing or hiring a front-end developer, there is a significant gap between expectations, reality and needs. As Laurie Voss puts it in an NPM video:

> 97% of the code in a modern web app comes from NPM.



Do we really write code or simply assemble it like Lego blocks?

At the end of 2018, we still see a majority of the interviewers asking for AJAX calls and candidates answering it in jQuery. Some of the classic examples of this gap are:

- CSS grid and flexbox are now widely supported. Yet, the interview is still focused on making **floats** and **inline-block** center of CSS discussion for multi-column layouts. Not to forget the never dying passion for Bootstrap or Foundation grid systems.
- Module bundler is a norm for large-scale applications. Yet, when it comes to architecture, we talk about minification and concatenation. How many times do we really discuss Webpack during the interview?
- If 97% of the code is coming from NPM, yet interview is focused on sorting array or iterating over object. Worse, we are still interested in finding out what `typeof null` is. **Why can't it be about understanding rational approach in choosing right libraries, frameworks or tools which are available in abundance?**

- We are still making interviewees do classical inheritance fit on top of prototypes instead of validating the need for such badly conceived ideas. We have many more functional patterns. Certainly, a good debate can shape up with use of JavaScript classes, newly introduced private and static fields. It will lead to a much better understanding of interviewee's opinions, critical decision making, etc.
- Caching discussion is still confined to Cache control headers and CDN. Things like IndexDB, HTTP2 or Service Workers are just passer-by.

The list is countless and the gap between interview assessment and actual needs of the job is evident. At one side we have front-end technologies making great leaps into future and on the other side, we have a large community which has not yet embraced this new way. A fractured community is never a good sign. It is a path of disaster. A gap always creates a new something that has a capacity to destroy everything we built so far. I cannot imagine Java Developers using GWT to write another Facebook.

An interview is a great place to make this change happen and to get everyone together. If you make interview sound like an interview, then it leads nowhere but inflated egos.

> *For the interview to succeed, it has to be a discussion. It has to be a place to exchange thoughts. It should challenge people to think and to objectively analyze the given problem. It is about understanding the decision process that one makes. It is about knowing one's passion for technology and problem-solving. It means knowing your possible future colleague. All those north-south pole puzzles, tricks or* `typeof null` *do not really make an interview.*

Below is the list of some of the questions we ask during the interview discussion. We hope that this list helps both interviewers and interviewees set the expectation, needs and realities in the right context.

> TLDR; We need to step up as Interviewers.

. . .

Basic JavaScript questions

1. Make the following code work:

```
const a = [1, 2, 3, 4, 5];

// Implement this
a.multiply();

console.log(a); // [1, 2, 3, 4, 5, 1, 4, 9, 16, 25]
```

2. Following code returns `false` in JavaScript. Justify why it happens:

```
// false
0.2 + 0.1 === 0.3
```

3. What are the different Data types in JavaScript?

Hint: Only two types — primary data types and reference types (objects). There are six primary types.

4. Solve the following asynchronous code problem.

Retrieve and calculate the average score for each student belonging to a classroom with some Id say 75. Each student can take one or more courses in a given year. Following APIs are available to retrieve the required data.

```
// GET LIST OF ALL THE STUDENTS
GET /api/students

Response:
[{
    "id": 1,
    "name": "John",
    "classroomId": 75
}]

// GET COURSES FOR GIVEN A STUDENT
GET /api/courses?filter=studentId eq 1

Response:
[{
   "id": "history",
   "studentId": 1
}, {
   "id": "algebra",
   "studentId": 1
},]
```

```
// GET EVALUATION FOR EACH COURSE
GET /api/evaluation/history?filter=studentId eq 1

Response:
{
    "id": 200,
    "score": 50,
    "totalScore": 100
}
```

Write a function that accepts the classroom Id against which you are going to calculate the average of each student in that classroom. The eventual output of this function should be the list of students with average score:

```
[
  { "id": 1, "name": "John", "average": 70.5 },
  { "id": 3, "name": "Lois", "average": 67 },
}
```

Write the required function using plain **callbacks**, **promises**, **observables**, **generators** or **async-wait**. Attempt to solve this problem using at least 3 different techniques.

5. Implement a simple data binding using JavaScript Proxy

Hint: ES Proxy allows you to intercept a call to any object property or method. To start with, DOM should be updated whenever an underlying bound object is changed.

6. Explain JavaScript concurrency model

Are you familiar with any other concurrency model that is used in other programming languages like Elixir, Clojure, Java, etc?

Hint: Look for Event loop, task queue, call stack, heap, etc.

7. What does `new` keyword do in JavaScript?

Hint: In JavaScript, `new` is an operator used to instantiate an object. The aim here is to understand what is happening in terms of scope and memory.

Also, lookout for `[[Construct]]` and `[[Call]]`.

8. What are the different function invocation patterns in JavaScript? Explain in detail.

Hint: There are four patterns, function call, method call, `.call()` and `.apply()`.

9. Explain any new upcoming ECMAScript proposal.

Hint: As in 2018, `BigInt`, partial function, pipeline operator, etc.

10. What are iterators and iterables in JavaScript? Any built-in iterators that you know?

11. Why JavaScript classes are considered bad or an anti-pattern?

Is it a myth? Has it suffered Chinese-whispers syndrome? Are there any use cases where they are useful?

12. How to serialize the following object in JSON?

If we convert the following object to JSON string, what would happen?

```
const a = {
    key1: Symbol(),
    key2: 10
}

// What will happen?
console.log(JSON.stringify(a));
```

13. Are you familiar with Typed Arrays? If yes, explain their need and differences as compared to traditional arrays in JavaScript?

14. How does default argument work?

If we have to use the default value of the `timeout` when calling `makeAPIRequest` function, what is the correct syntax?

```
function makeAPIRequest(url, timeout = 2000, headers) {
   // Some code to fetch data
}
```

15. Explain TCO — Tail Call Optimization. Is there any JavaScript engine that supports Tail Call Optimization?

Hint: As of 2018, there are none.

. . .

JavaScript Front-end Application Design Questions

1. Explain one-way data flow and two-way data binding.

Angular 1.x was based on two-way data-binding whereas React, Vue, Elm, etc. are based on one-way data flow architecture.

2. Where does unidirectional data flow architecture fit with regards to MVC?

MVC has a solid history of ~50 years and have evolved into MVP, MVVM and finally MV*. What is the co-relation between the two? If MVC is an architectural pattern, what is Unidirectional data flow? Are these competing patterns to solve the same problem?

3. How is client-side MVC different from server-side or classical MVC?

Hint: Classical MVC is the Smalltalk MVC meant for a desktop application. In web applications, at a minimum, there are two distinct data MVC cycles.

4. What are the key ingredients that make functional programming distinct from object-oriented or imperative programming?

Hint: Currying, point-free functions, partial function application, higher order functions, pure functions, isolated side effects, record types (unions, algebraic data types), etc.

5. In the context of JavaScript and Front-end, how does functional programming relate to reactive programming?

Hint: No correct answer. But in crude terms, functional programming is about coding in small, writing pure function and reactive programming is coding in large, i.e. data-flows between modules, connecting components written in FP style. FRP — Functional Reactive Programming is another different but related concept.

6. What are the problems solved by immutable data structures?

Are there any performance implications of immutable structures? What are some of the libraries in JS ecosystem that provide immutable data structures? What are the pros and cons of these libraries?

Hint: Thread safety (Do we really need to worry about this in browser JavaScript?), Side-effect free functions, better state management, etc.

7. Should large-scale applications use static typing?

1. How does TypeScript or Flow compare to transpile-to-JS languages like Elm, ReasonML or PureScript? What are the pros and cons of these approaches?
2. What should be the prime criteria for selecting a particular type system over the others?
3. What is type inference?
4. What is the difference between statically typed language and a strongly typed language? What is the nature of JavaScript in this regards?
5. Do you know any language that is weakly typed yet statically typed language? Do you know any language that is dynamically typed yet strongly typed?

Hint: Structural vs Nominal type system, type soundness, tooling/ecosystem support, correctness over convenience.

8. What are the prominent module systems in the JavaScript world? Comment on ES Module system.

List some of the intricacies involved when achieving interoperability between different module systems (Mostly interested in ES Module and CommonJS interoperability)

9. How HTTP2 will impact JavaScript application packaging?

List fundamental characteristics that separate HTTP2 from its predecessors.

10. What improvements does Fetch API provide over traditional Ajax?

1. Are there any drawbacks/pain points of using Fetch API?
2. Is there anything that Ajax can do while fetch cannot?

11. Comment on pull-based vs push-based reactive system.

Discussion concept, implications, uses, etc.

1. Add lazy vs eager evaluation to this discussion.
2. Then add singular and plural values dimension to this discussion.

3. Finally, talk about synchronous and asynchronous nature of value resolution.
4. Provide the example for each combination that is available in JavaScript.

Hint: Observable is a lazy, push-based, plural value construct with both async/sync schedulers.

12. Discuss issues associated with Promises in general.

Hint: eager evaluation, awkward cancellation mechanism, masquerading `map()` and `flatMap()` with `then()` method, etc.

. . .

Front-end fundamental and theory questions

1. What is the use of Doctype in HTML?

Specifically what will happen in each of the following scenarios:

1. Doctype is absent.
2. HTML4 Doctype is used but HTML page uses HTML5 tags like `<audio>` or `<video>`. Will it cause any error?
3. Invalid Doctype is used.

2. What is the difference between DOM and BOM?

Hint: BOM, DOM, ECMAScript, and JavaScript are all different things.

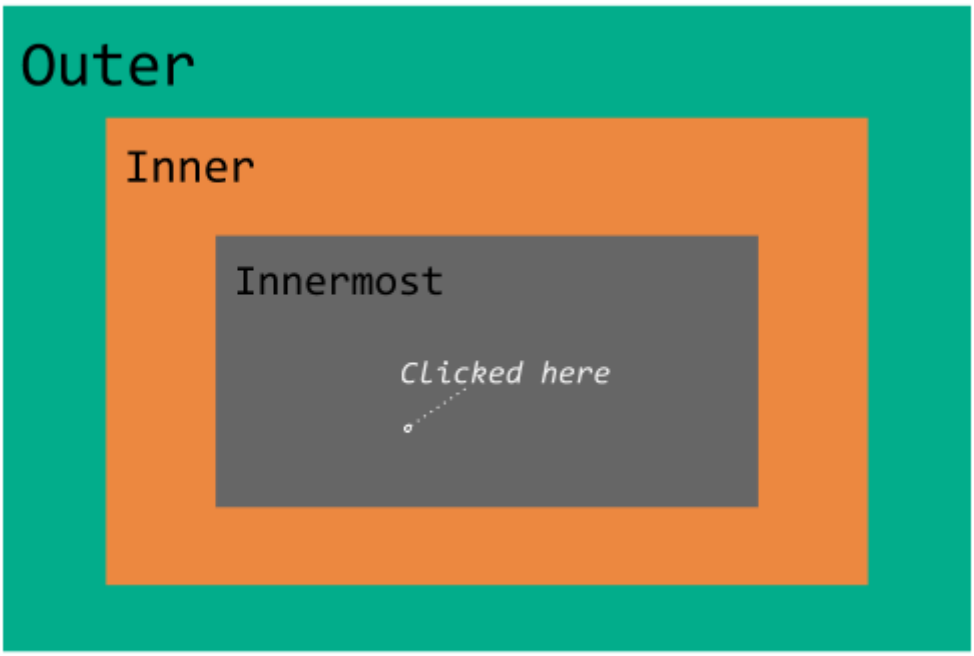
3. How does event handling work in JavaScript?

As shown in the next diagram, we have three `div` elements. Each of them has a click handler associated with them. Handlers perform following tasks:

1. Outer div click handler prints `hello outer` to console.
2. Inner div click handler prints `hello inner` to console.
3. Innermost div click handler prints `hello innermost` to console.

Write a code for assigning these handlers in such a way that the following sequence should always be printed whenever innermost div is clicked?

```
hello inner ⟶ hello innermost ⟶ hello outer
```



Event bubbling and capturing

Hint: Event capture and Event bubble

4. What are the different ways to upload a file to a server using Single Page Applications?

Hint: XMLHttpRequest2 (streaming), fetch (non-streaming), File API

5. What's the difference between CSS re-flow and repaint?

Which CSS properties cause re-flow and repaint on change?

6. What is CSS selector specificity and how does it work?

What is the algorithm to calculate the CSS specificity.?

7. How CSS pixel is different than hardware/physical pixel?

Hint: A pixel is not a pixel is not a pixel — ppk.

8. What is the sectioning algorithm?

Hint: It is also known as HTML5 outline algorithm. Particularly, important when it comes to building websites with semantic constructs.

9. If you have used CSS flex/CSS grid, why did you use it? What problems did it solve for you?

- With CSS grid, how do percentage `%` and `fr` unit differ?
- With CSS flexbox, sometimes flex-items/children do not respect width/height set by the flex container? Why does this happen?
- Can a Masonry layout be created with a CSS grid? If yes, how?
- Explain CSS grid and CSS flexbox terminology?
- How does floating element (`float: left | right;`) render inside CSS grid and flexbox?

Hint: Columns of equal heights, vertical centering, complex grids, etc.

10. When should CSS animations be used as opposed to CSS transitions? What is the deciding criteria?

11. If you are reviewing code for CSS, what are the common gotchas that you look for in a code?

Examples: Use of magic number like `width: 67px;` or using `em` instead of `rem` unit, writing media queries before general code, abusing ID and classes, etc.

12. How to detect touch events in JavaScript?

1. Do you believe that detecting device support for touch events is a bad idea? If yes, why?
2. Compare touch events and pointer events.
3. When a device supports both touch and mouse events, then what is or should be the correct event order for these events in your opinion?

13. What is the use of `async` and `defer` attributes defined for script tag?

1. Are they really useful now that we have HTTP2 and ES Modules?

. . .

The list presented is just a glimpse of how possibly infinite points we can discuss during the interview. There are many things like Web Components, CORS, Security, Cookies,

CSS transforms, Web Assembly, Service Workers, PWA, CSS architecture, etc. that we did not consider. We also did not cover framework or library specific questions.

Hope this guide helps the community set itself in the right direction when it comes to interviews.

JavaScript Front End Development Web Development CSS HTML