30 Q&A For Your
Next Frontend Interview

Bigson Dev

Adrian Bigaj

About This Guide

All of the questions included are based on my experience and could potentially occur in one of your interviews. I tried to cover questions for various levels of seniority. The Guide was created for anyone preparing for Frontend Technical Interview, please, treat it as a refresher. I added my own thoughts between the proposed answers to direct you on what recruiter would expect as an answer to the question.

How Come You Know All Of This?

I described some of my competences on bigsondev.com, mentioning 12 years of experience (5 as a Frontend Developer, 2 as a Frontend Recruiter, and 5 as a League of Legends Coach).

All these years were for me to learn how to share knowledge in the best possible way with you.

I hope you will smash your next Frontend interview and the below questions and answers with help with that!

> Proposed answers are my subjective opinion and not the ultimate exact solution to the question. They're robust enough to cover most of the topics, but I am not limiting you to answering in the same way. Treat it as a base fundamental on which you could expand when preparing an answer to a similar question you might be asked during the interview.

P.S. Please try to do Quiz or two, on BigsonDev - Library before checking the answers to test your current knowledge.

Table of Contents

15 Junior Questions & Answers

Q: What are Browser Developer Tools?

A: Tools that every Frontend Developer should know and extensively use daily for several reasons. Shortcut for opening them is Command + Option + J (on Mac - Chrome, tell the shortcut for the system you use and Browser) or right-click on the element followed by clicking on "Inspect". The reasons to use them are (just a couple of examples - add yours to the list):

- *Inspect elements to check the CSS, entire page HTML (DOM) structure to check which styles are applied, to inspire how other websites are styled and which structure they have (including JS code)*
- *Check the Network tab and see which requests are made (in XHR filter) or which Fonts, CSS, Images, etc.*
- *Debug code in the Sources tab as apps can be complex and console.log might not be enough to go through the entire flow*
- *Simulate Network conditions by using Network throttling or even testing the browser in Offline mode*

Q: What is CLI?

A: CLI stands for the Command-line interface. It is a text-based User Interface to view and manage files. But it's also a lot more, managing git commands, NPM, Node.js. These days there are a lot of CLI (e.g. Angular CLI) to fasten the work and expose a lot of "manual" work in automated ways through simple commands. I use iTerm on Mac as a Terminal Replacement with Oh My Zsh to extend basic Shell with themes and plugins, one of my favorites is a git plugin to simplify the entire workflow of it. (Try to name your Terminal, OS, and plugins/themes if you use any, or just advantages of using them, it will let you shine in the recruiter's eyes, tell about automating manual tasks through commands / most used commands, etc.).

Q: 1 + 5 = 12, 2 + 10 = 24, 3 + 15 = 36, 5 + 25 = ?

A: (Could be a Knowledge Test or just some simple logical task to solve on the board, this one is purely about finding a pattern and coming up with the answer quickly, try to find similarities. The first two numbers are always added together and represent half of the result, in other words, it's 6 = 12, 12 = 24, 18 = 36, 30 = ?. I guess you already know that it's 60, it is just harder to notice when 1 + 5 is used instead of 6. It's worth practicing logical tasks upfront so you won't be surprised).

Q: How would you describe Components in relation to the Frontend/User Interface?

A: Components are the building blocks of the software. In relation to User Interface, Atomic Design is a good reference, where we have Atoms, Molecules and Organisms. Atoms could be a Button, Title, Avatar, just the smallest possible chunks of code, abstracted into reusable pieces aka Components to avoid duplicating code (DRY - Don't Repeat Yourself). Molecules are based on composition, not inheritance of Atoms and could represent Navigation, Card, Stepper, Wizard etc. Organisms could be whole pages composed out of Molecules and Atoms. The main points of creating Components are scaling, staying in sync with Designers, adjusting/extending code in one place and being happy that everything is abstracted in a nice way.

Q: Which Client-side storages exist in the Browser?

A: Cookies, Web Storage (localStorage and sessionStorage) and IndexedDB. It's worth mentioning that Web Storage has much more capacity than Cookies do (it would be the next question anyways very likely). The difference is between 5MB (Web Storage) and 4KB (Cookie). Web Storage is limited to only one domain, it would require tricky solutions such as iframes on other domains to make it cross-domain. Cookie can deal with that by setting appropriate headers to make it available for cross-domain purposes. Another important note is about sending data to the server, when using Web Storage it won't be sent with every request, but with cookies every request will attach additional data due to the existing cookies. (Bonus points if you mention anything about IndexedDB).

Q: What is VCS?

A: Version Control System, such as Subversion (SVN) or Git. These days Git is the most common VCS due to its simplicity and flexibility. VCS is basically a way to track project changes and allow multiple developers to work in parallel. It's the most efficient way to do code reviews, share code with others and avoid conflicts or repeating the same solutions. (Worth to mention standard Git flow and concept of branches). Master branch is always the first branch created in the repository, developers usually work on feature branches. There are branches for different environments such as staging, QA, production (sometimes referred as release branches and have different naming conventions). Hot ins and shortcuts you use in your IDE?"). It's an Integrated Development Environment to fasten the workflow of writing code. I use VSCode (www.code.visualstudio.com) as it's super fast compared to WebStorm or other Intellij products (you can mention the editor of your choice). It has an amazing community and is growing quite fast. Smart select, multiple cursors, multiple tabs to compare different files, replacing entire sentences in the Editor are things that I like. If it comes to the extensions, all kinds of syntax highlighting extensions, Auto Import, Debugger for Chrome, Snippets that allow you to quickly write code and repeat the same starting templates. Eslint, Prettier, MDX, Typescript and others. (You should mention plugins, shortcuts that you use on a daily basis and some specific features to your editor). There is also that great functionality for pair programming of Live Sharing which is really helpful when working with others / mentoring.

Q: What is an IDE?

A: (I'm pretty sure this question might sound trivial to you, but it could also be asked like that - "What plugins and shortcuts you use in your IDE?"). It's an Integrated Development Environment to fasten the workflow of writing code. I use VSCode as it's super fast compared to WebStorm or other Intellij products (you can mention the editor of your choice). It has an amazing community and is growing quite fast. Smart select, multiple cursors, multiple tabs to compare different files, replacing entire sentences in the Editor are things that I like. If it comes to the extensions, all kinds of syntax highlighting extensions, Auto Import, Debugger for Chrome, Snippets that allow you to quickly write code and repeat the same starting templates. Eslint, Prettier, MDX, Typescript and others. (You should mention plugins, shortcuts that you use on a daily basis and some specific features to your editor). There is also that great

functionality for pair programming of Live Sharing which is really helpful when working with others / mentoring.

Q: Given the following code:

```
for (var i = 0;  i < 5; i++) {
setTimeout(() => console.log(i), i * 1000)
}
```

What will show up in the Console?

A: The result will be 5, 5, 5, 5, 5. (logged with delay of one second each). The reason for that is each function in the loop will be executed when the entire loop has completed. Loop quickly iterates to 5 and stops executing at this point as the loop only exits when the condition breaks. While i is incremented to 5, console.log will occur with the delay and log number five, five times. The solution to this problem could be to use let instead of var or an IIFE (closure).

Q: After executing the following command:

```
git pull
```

What happens under the hood?

A: git pull runs two commands, git fetch (to sync with the remote origin) followed by git merge (to apply all the changes from the remote branch), e.g. if I am working on feature-branch-abc and this branch was checked out from develop, git pull will merge the changes from the latest remote develop branch into my feature-branch-abc (which could cause potential conflicts). It's recommended to pull often as dealing with massive code changes and more conflicts later on can consume a lot of time. (Dealing with smaller problems is easier than with a big one, ain't it?)

Q: What Data Types exist in JavaScript?

A: There are 8 Data Types in JavaScript. Number, String, Boolean, BigInt, Symbol, undefined, null (Additionally, null is used when an object has nothing more in the prototype chain, e.g. it does not inherit from anything anymore). The last one is Object which is the most complex one. Worth

to mention that Array and Function constructors are also derived from Object constructor.

Q: Given the following code:

```
console.log(42 == '42')
console.log(42 === '42')
```

What will show up in the Console?

A: In JavaScript == is an Equality operator, === is an Identity operator. Equality operators always try to cast the left side and right side into the same type. On the other hand, Identity operator requires both sides to be of the same type as a prerequisite. Hence, when comparing 42 == '42', string '42' is coerced into number 42 which results in true. With the second example, 42 is a number, but '42' stays as a string and the result is false as number and string are different types.

Q: Given the following code:

```
theAnswerToLife = 42;
var theAnswerToLife;
console.log(theAnswerToLife);
```

What will show up in the Console?

A: If var would be changed to let or const, we would get "Uncaught ReferenceError: Cannot access ,theAnswerToLife' before initialization", but var doesn't have Temporal Dead Zone and its declaration is hoisted up, the result will be 42. Imagine that JavaScript Interpreter is moving var theAnswerToLife to the first line and theAnswerToLife = 42 is moved to the second line instead.

Q: What is the structure of DOM?

A: DOM stands for Document Object Model and it's defining HTML elements as objects. When a web page is loaded, Browser creates the DOM model of the page. This model is constructed as a tree of objects. Tree model was a potential choice because of reflecting relationships between the HTML elements (parent and children metaphor) and representing a hierarchy between them. Tree data structure provides

an efficient insertion and searching and is very flexible to allow manipulations on nodes.

Q: What is a Polyfill?

A: Code which provides the fallback solution for some specific functionality that is expected to work in latest browsers, but might not be available in older browsers e.g. in IE11 and needs to be supported. Each time there is something new in ECMAScript, such as anonymous functions, null coalescing operator, or CSS specific features, e.g. Flexbox, CSS Grid, a new question arises - "Can I use it?" It's recommended to check browser support and compare it with project requirements, if the feature is worth it then polyfill should be applied. Sometimes it's better to use the older, more stable solution and wait for the feature to be supported in all browsers (at least 2 latest versions).

Q: Given the following HTML code:

```
<div id="i-rule" class="no-i-rule">Hello CSS Specificity</div>
```

And the following CSS code:

```
#i-rule {
background-color: red;
}
div {
background-color: green;
}
.no-i-rule {
background-color: blue;
}
```

What will be the color of div?

A: This question refers to CSS Specificity rules. There are various calculators available on the Web to calculate the "power" of selectors. For not an experienced Frontend Developer it might seem that the last .no-i-rule selector will override all the previous ones, which is not true. CSS Specificity rules are represented in numbers, where id selector has a number of 100, class, attribute and pseudo-class selector has a number

of 10, element and pseudo-element selector has a number of 1. Given the above explanation, the background color of this particular div will be red. There is an "!important" declaration which overrides all of the Specificity rules, but it's not recommended to use in most scenarios. It's worth noting that selectors can be combined to create more powerful Specificity rules.

10 Regular Questions & Answers

Q: Given the following code:

```
console.log(0.1 + 0.2);
```

What will show up in the Console?

A: The answer is 0.30000000000000004. A bit surprising. It all comes down to that there is only one number type in JavaScript: (Technically there is a BigInt newly introduced, but this scenario is not connected with it), double-precision 64-bit binary format IEEE 754 value. "Floating numbers can't store properly all decimal numbers, because they store stuff in binary". It's not a JavaScript problem, but more general, a computer one. It's worth remembering that responsibility for formatting numbers is on Developers, e.g. how they are displayed to the end-user. In complex payment systems it's often important to understand business logic behind and how many decimals should be shown (hopefully all the calculations are done under the hood in the BE).

Q: What is the difference between:

```
const friends = [];
friends.push('Jimmy');
```

and

```
const updatedFriends = [...friends, 'Jimmy']
```

A: It should be mentioned that the first example mutates the friends array, the second example uses spread syntax to copy contents of the friends array, add 'Jimmy' string to it and with [...friends, 'Jimmy'] creates the updatedFriends array that is free of side effects and mutations to the friends array. This small example touches a concept of Functional Programming, being declarative, avoiding side effects and introduces to pure, clean code. (You could potentially go crazy about that, but the above answer should be a minimum to fulfill the question).

Q: Which tools can help with code quality?

A: This one is deep, but start with ESLint, Prettier, Testing (Jest, Cypress), TypeScript, properly configured CI & CD and local dev setup (at least some husky hooks to run tests, prettier, checks before pushing to the repo) and explain that running tests in pipeline jobs will help with catching potential bugs before pushing to production. (Learn about these tools and start using them in your projects, as recruiter could follow on any of them as the next question)

Q: How can you apply a bug fix on different branches?

A: Let's say that you were fixing a critical bug, found a solution for that and it needs to be applied on different environments (e.g. staging, QA, production), each of them have a separate branch. You could mention git cherry-pick as a potential solution, it takes a commit from your current branch and applies it onto another branch, which in that case would be 3 different branches

Q: Given the following code:

```
const mysteriousFunction = (callback, wait) => {
  let timeout;
  return (...args) => {
    const context = this;
    clearTimeout(timeout);
    timeout = setTimeout(() => callback.apply(context, args), wait);
  };
}
```

What is the mysteriousFunction doing?

A: During the interview, you could be presented with a set of different code challenges. This one is a debounce function which clears an execution of callback if the user keeps clicking e.g. on a button before waiting a fixed amount of milliseconds set in the "wait" parameter. Additionally you could mention a throttle function which splits an amount of clicks into delayed executions over time, e.g. if a user clicks ten times on a button and the "wait" parameter is set to 1000, each callback will occur every one second, ten times. (Play with debounce, throttle, additionally read about memoization and practice code challenges, algorithms to become better at it before the interview).

Q: What is GraphQL?

A: (Other questions from this area could be - "What is a REST API", "Explain CRUD Principle"). GraphQL is a modern alternative to the REST API, where instead of hitting multiple endpoints and always fetching all the data, the client has power to choose which exact data he needs through Queries. Additionally, there is only one endpoint /graphql (usually) and everything is done using mentioned Queries, Mutations and sometimes Subscriptions. GraphQL (e.g. react-apollo) has a strong cache mechanism and saves a lot of time compared to REST implementation. It's straightforward to specify what data needs to be fetched and thanks to that response from the server can be minimized immensely. (I highly recommend playing around with e.g. Gatsby and GraphQL as these technologies are being heavily used in modern companies).

Q: Why is state management needed in modern web applications?

A: Modern web applications are composed of multiple features that often require a lot of re-rendering, transactions and instant feedback to the user. They include large forms, steps, wizards and a lot of moving around. Modern web applications are called SPA (Single Page Application) where native browser refresh is not cool anymore, and instead, e.g. in React, custom routing libraries are used to create a feeling of using a native app (no reloading, quick view changes). Under the hood, there might be a lot of logic to change views, save the form data, update data, delete data, edit data and so on. State management libraries such as Redux, Mobx or sometimes even React hooks can try to solve that by introducing a friendly, straightforward API which helps working with state changing horror. (Try to learn at least one of them, Redux is highly recommended still and has a fantastic concept of declarative, functional approach, no mutations etc.)

Q: Given the following code:

Hello.jsx

```
export const Hello = () => <div>Hello World!</div>;
```

Will it execute Properly?

A: It's a tricky question as we have a React Stateless Component there. It seems that everything is implemented properly, but React import is

missing. Without it, it won't be possible to do JSX Transpilation to React. createElement..., as even if it's just a Hello World! , it's still a React component. (These days almost all components are Stateless due to the new Hook functionality in React, but try to remember that it's JSX under the hood and still needs to be transpilled appropriately).

Q: Why is TypeScript so popular?

A: It has a huge community and amazing support for the vast majority of npm libraries (at least the big ones), support for available methods, components. Also, when working in a larger team, writing a code without static typing can cause a lot of technical debt and potential bugs. Static typing was a missing feature in JavaScript for a long time and TypeScript is a great solution for that. (TypeScript might not be needed in your personal projects or even smaller, not large scale projects. However, it's good to be aware of it, even if it could be considered an additional boilerplate).

Q: What is Virtual DOM?

A: DOM computations are not too performant. Doing direct, complex re-renderings, state changes on the UI would cause freezes and big lag spikes for the end users, especially without an in-depth knowledge about DOM manipulations. Modern frameworks such as React, Vue have a solution for that which is called a Virtual DOM. It's an in-memory representation of the real DOM, all the updates are batched together and applied first to the Virtual DOM. At the end, one, last, combined update is applied on the real DOM to minimize the blocking of UI. (Try to practice some direct manipulations on DOM and see how cumbersome the native API is, then try out e.g. React and see how simple it is).

5 Senior Questions & Answers

Q: Which trade-offs are in Cypress?

A: (This one is a common question for Senior Frontend Developers not only about Cypress, but there are always trade-offs in different solutions, libraries, frameworks, etc. Being on this level, it's expected from a candidate to understand the advantages as well as the pain points.). There are a couple of them, one specific for our team was that Cypress can't execute tests for multiple origins. We had an application where clicking on the specific button, the user was moved to another application and we couldn't test that. In Cypress there is no support for multiple tabs and it's limited to only one Browser at the same time. For a long time, there was a lack of other browsers. (Good to mention your personal experience with the library instead of reciting a formula).

Q: Given the following code:

```
const foo = undefined ?? 'default placeholder';
```

What is the ?? operator called?

A: (Again, with Senior experience, this question could have many forms. It's basically checking if a candidate is still up to date with the latest trends and willing to learn new things). It's a Null Coalescing Operator introduced in ECMAScript 2020, 11th Edition. It only checks for null and undefined, omitting other falsy values. E.g. if 0 would be used instead of undefined, foo will be set to 0 instead of 'default placeholder'. (You should never stop learning, especially in the Frontend, it's exciting as it's growing rapidly, but also challenging and requires a lot of effort and love to stay up to speed, still, it's worth and rewarding IMO).

Q: What is the purpose of sideEffects property in Webpack?

A: (In a lot of projects, recruiters look for Senior Frontend Developers for a bunch of reasons. But, the most common reason is to align on MVP with the client, boilerplate project, setup CI/CD, development flow for future developers to make it stable, scalable from the start. You need to acknowledge these tools to make it a breeze). This question is all about the Tree Shaking concept, Webpack 4 sideEffects property

was introduced to tell Webpack that our code is free of side effects and unused imports can be removed which is also known as Dead-code elimination, to decrease the size of the bundle. (There could be more questions about chunks, lazy routing, bundles, other optimization techniques, or just how to effectively set up a new project, get ready to beat the interview).

Q: What are Micro Frontends?

A: Individual, semi-independent applications for better scalability for the main application. It's a hot topic in the Frontend World, especially in large-scale applications and complex systems. They're the Microservices, but e.g. in React and not on the server-side. (It's worth mentioning the monorepo, what it is, and why it could be considered. All the thoughts about routing, dependencies between teams will be welcome and an open, brainstorming discussion to share the knowledge between you and a recruiter).

Q: What are peerDependencies?

A: (This question checks your knowledge about building a library, e.g. components library, as peerDependencies are usually used there)
There are three types of dependencies in general. devDependencies, dependencies, and peerDependencies. The last one is used when building the library, e.g. components library in React. To avoid duplicating instances of React, it should be added as a peerDependency of the library package.json. The consumer of this library should provide React as a dependency in the project's package.json. (It's highly recommended for a Senior Frontend Developer to create at least one, custom, small library, push it to npm, and use it in some other project to see how the entire flow works, worth doing some open-source project as well).