MockIT

Candidate: Jan Kowalski
Technology: JavaScript
Sub-technologies: React, Node.js, TypeScript
Level: Junior
Conversation Language: English
Date: 01/01/2024

General Impressions

During the meeting, you demonstrated a good knowledge of JavaScript and TypeScript, as well as basic knowledge of React and Node.js (Junior level). You are not familiar with the more advanced mechanisms of React and Node.js. You are deficient in code testing.

When solving problems, you should focus more on the simplicity of the solution you are creating, analyze possible solutions and explain your ideas more clearly. Your ideas were simple but reasonable and your commitment and previous experience are promising. In our opinion, you qualify for a Junior position.

Experience and projects

When applying for the junior level, private projects are important. It is very good that you have them. It's even better that the projects are ambitious and that you use a lot of meaningful technology in them.

Comments:

- Sometimes you use variables in the camelCase convention and sometimes snake_case. It would be good to standardize this. The same goes for code organization and file naming.
- Your methods are often responsible for several/some things making them very long and unreadable.
- During the interview you mentioned that you are involved in open-source projects. It would be good to mention this in your resume.

Technical skills

You answered most of the questions comprehensively and correctly. However, there were some where your answer was too general or you did not give one.

1. **React and the general frontend**
    - You know the basic page rendering methods - in your spare time you can also read about ISR and SSG and learn the difference between them
    - You are not familiar with the concept of memoization in React
    - You have a basic knowledge of hooks in React however you are not familiar with the concept of custom hooks
    - You don't fully understand the difference between DOM and Virtual DOM
    - You are familiar with Redux and Context API

2. **Craftsmanship**
    - You are trying to make reusable components, you know some good coding practices, you have mentioned DRY, KISS principles, we just encourage you to have a broader understanding of SOLID principles
    - You know what to do and how to act quickly when you are blocked by a task
3. **Backend**
    - You know the basics of NodeJS
    - You know ORMs and what their advantages are
    - You know how OAUTH works and what JWT tokens are used for
    - You know the differences between REST and GQL
    - You do not know the assumptions of Node.js (event loop)
    - You are not fully aware of all the functionalities of the next() method
4. **Typescript**
    - It is apparent that you know and understand Typescript
    - You could not point out the antipatterns when using TS
5. **Javascript**
    - You did a great job with the JS questions (deep clone objects, constructs introduced in ES6, pointing out incorrect code snippets)
6. **Testing**
    - You mentioned some tools to test your code but didn't say what type of tests you can create with them

Problem solving

During the interview, you received such a task:

Transform the array arr = [1, 1, 2, 4, 9, 1, 3, 5, 3, 7] so that it contains only unique values

You decided to solve the task by adding an auxiliary array and base the whole algorithm on two loops (loop within a loop). Your solution worked but it is not the best way. The computational complexity of the algorithm you wrote is O(n2). It is much better to use an auxiliary object instead of an array (objects store data in the form key => value). This way you could remove the inner loop and the algorithm would have O(n) complexity. An even better approach is to use the **Set()** method.

Proposed solution:

const uniqueArray = (arr) => Array.from(new Set(arr));

Knowledge of computational complexity goes beyond the Junior level, however, it is good to keep this in mind and optimize algorithms. In addition, using an auxiliary object in this case should be natural for you.

Communication

1. **Form of response**
    - Usually comprehensive and elaborate if you know the topic well.
2. **Cooperation with the recruiter**
    - In several situations I had the impression that if you had asked me the details of the question asked you would have answered better.
    - Discuss every ambiguity with the recruiter, ask questions. Lack of questions to the recruiter works against you.

3. **Dialog**
    - While writing the algorithm, there was a lack of communication on your part about what you were currently thinking about or what the code you were writing represented.
    - Your soft communication skills are at a high level, which made you good to talk to.
    - Your openness and ability to listen highlight your ability to collaborate effectively in a team.

Additional comments

Focus on deepening your knowledge in React and Node.js. Study good programming practices. It's also a good idea to learn how to use testing tools/libraries like Jest or Cypress, which are often required in frontend job offers. At your leisure, take a look at the resources available here: https://github.com/getify/You-Dont-Know-JS. Other notes/concepts you should pay attention to are:

- Choosing the right application architecture, how to organize the project, limits of responsibility of modules, advantages/disadvantages of different approaches
- Computational complexities
- Methods built into Javascript
- Writing SRP-compliant code (application after analyzing your projects)