Answers to QA Assessment

1. Something is not displaying properly in the browser and you want to learn more about it. How do you go about doing that? (Looking for what tools and specific steps you would take to investigate the error)

To investigate an issue with something not displaying properly in the browser, I would follow these steps using various tools:

- Inspect the Element:

Use the browser's built-in developer tools (F12 or right-click and select "Inspect").

Navigate to the "Elements" tab to examine the HTML and CSS related to the problematic element.

- Check the Console:

Switch to the "Console" tab to look for any error messages or warnings. These messages can provide insights into JavaScript errors, network issues, or other problems.

- Review Network Activity:

Go to the "Network" tab to check if there are any failed network requests or resources that didn't load correctly (e.g., missing images, CSS files, or JavaScript files).

- Debug JavaScript:

Use the "Sources" tab to set breakpoints and step through the JavaScript code. This helps identify logical errors or issues with dynamic content generation.

- Check for Cross-Browser Issues:

Test the application in different browsers (Chrome, Firefox, Edge, Safari) to determine if the issue is browser-specific.

- Validate HTML and CSS:

Use online validators like the W3C Markup Validation Service to check for any syntax errors or invalid HTML/CSS that could cause rendering issues.

- Consult Logs:

Check server logs if the issue might be related to server-side processing.

- Search for Known Issues:

Search online for similar issues. Developer forums, Stack Overflow, and the browser's official documentation can be helpful resources.

2. Have you ever broken a piece of software? How did you break it? How did you fix it?

Answer:

Yes, I have broken a piece of software during my development work. Here's an example:

- How I Broke It: While refactoring a module in a web application, I inadvertently introduced a bug by overlooking a dependency between two components. This caused a critical feature to fail, leading to a runtime error.
- How I Fixed It:
    - Identify the Issue: I immediately noticed the problem during testing and used the browser's developer tools to trace the error.
    - Reproduce the Bug: I reproduced the issue to understand the conditions under which it occurred.
    - Analyze the Code: I reviewed the recent changes I made and identified the missing dependency.
    - Implement a Fix: I corrected the code by properly handling the dependency and tested the module again to ensure the fix worked.
    - Code Review: I submitted the fix for code review to get feedback from my peers and ensure no other parts were affected.
    - Testing: After the code review, I conducted thorough testing, including regression testing, to confirm that the bug was resolved and no new issues were introduced.

This experience taught me the importance of thorough testing and careful consideration of dependencies during code changes.

3. Describe your experience with code review.

I have substantial experience with code review, both as a reviewer and as a reviewee. Here are some key aspects of my experience:

- Reviewing Code:
    - Objective: My primary goal is to ensure code quality, adherence to coding standards, and the correctness of the functionality.
    - Process: I thoroughly examine the code for potential bugs, performance issues, and readability. I also check for proper documentation and test coverage.

    - Feedback: I provide constructive feedback, highlighting both good practices and areas for improvement. I suggest specific changes or improvements when necessary.
- Receiving Reviews:
    - Openness: I appreciate and welcome feedback from my peers as it helps me improve my coding skills and catch issues I might have missed.
    - Implementation: I carefully consider all feedback and implement the necessary changes. If I disagree with a suggestion, I discuss it with the reviewer to reach a consensus.
- Tools Used:
    - Platforms: I have used various platforms for code reviews, including GitHub, Bitbucket, and GitLab.
    - Automation: I leverage automated tools like linters and static code analyzers to catch common issues before the code review process.
- Benefits:
    - Learning: Code reviews are an excellent learning opportunity, allowing me to see different approaches to solving problems.
    - Quality Assurance: They significantly improve the overall quality of the codebase by ensuring multiple sets of eyes examine every change.