Warm-Up Retrospective

Date: 10/20/2024
Project Name: Refactoring Bigfoot Library to JavaScript

---

1. What Went Well

Achievements:

- Thoroughly reviewed the Bigfoot and Littlefoot libraries, including their design, structure, and the improvements required for a successful refactor.
- Successfully identified the differences between jQuery and JavaScript, focusing on how to transition features from jQuery-based to native JavaScript, ensuring a cleaner and more modern implementation.
- Eliminated the original dependency on JavaScript libraries like jQuery, allowing for cleaner integration, better performance, and a more modular codebase.
- Refactored the Bigfoot library into JavaScript, enhancing maintainability and scalability.

Team Dynamics:

- Team members demonstrated excellent collaboration during the early phases of the project, especially in understanding and dissecting the existing code, which laid the groundwork for a smoother refactoring process.
- Despite short notice for some meetings, the team remained highly motivated, with the majority of members attending even if meetings were scheduled just a few hours prior.
- Open and constructive discussions around dependency removal allowed for multiple perspectives, resulting in over 8 ideas and 5 formal ADRs, showcasing the team's analytical and creative problem-solving abilities.

Tools & Processes:

- The use of version control, combined with frequent and thorough code reviews, ensured that all team members were able to contribute meaningfully and that code quality remained high.
- The team effectively used issues and branches to manage the repository, ensuring that work was clearly separated into independent branches. This modular approach facilitated parallel work streams and avoided integration issues.
- Even those team members with little experience in repository management quickly adapted and became proficient, contributing to a well-managed and organized codebase by the end of the warm-up phase.

---

2. Challenges Faced

Technical Challenges:

- Refactoring an outdated language base posed significant challenges, especially when transitioning the Bigfoot library from jQuery to modern JavaScript. This required substantial effort in understanding legacy patterns and rethinking them using modern practices.
- Complex portions of the library required different designs than those used originally, as JavaScript and jQuery handle DOM manipulation and asynchronous logic differently.
- The removal of the original JavaScript dependency introduced some unexpected technical roadblocks, delaying certain aspects of the refactoring and requiring additional research and rework.

Team Communication:

- Some team members struggled initially with aligning on the overall structure and design of the refactored code, leading to occasional misunderstandings during implementation.
- Varied levels of JavaScript experience within the team resulted in differing development speeds, as less experienced members required more time to familiarize themselves with the language and coding patterns.
- Frequent changes in meeting plans, particularly around offline meetings, made it difficult for everyone to attend simultaneously, resulting in information discrepancies and a lack of continuity for some team members.

Time Management:

- The team faced tight deadlines, which combined with other academic commitments, put significant pressure on the group. This was particularly evident toward the end of the warm-up phase, where balancing coursework and refactoring work became challenging.

---

3. Team Analysis

A. Individual Contributions

Strengths:

- Several team members demonstrated expertise in multiple areas, including GitHub management, repository structuring, and JavaScript, helping guide the rest of the team through the refactoring process.

- Team members stepped up and offered to help where needed which helped to get the project going
- Everyone attended class which showed our commitment to this class and to each other
- [someone could add their strengths here]

Weaknesses:

- Some team members lacked sufficient experience in web development, particularly with JavaScript, which resulted in a slower learning curve and impacted their ability to contribute effectively in the early stages.
- Conflicting schedules. For most meetings, not all members were able to attend due to conflicting schedules. This made communication more difficult.
- [someone could add their weakness here, if don't mind]

B. Team Operations

Strengths of Team Operations:

- The team consistently maintained a regular meeting schedule and utilized collaborative tools such as Google Docs for real-time communication, decision tracking, and documentation.
- Decisions were made in a collaborative manner, with every member having the opportunity to provide input. This approach ensured balanced ownership of tasks while keeping the team aligned on overall goals.
- Setted general due dates which helped us to have steady progress and not procrastinate
- Most team members lived in grad housing which made for easier coordination and establishing meeting locations

Weaknesses of Team Operations:

- There was occasional ambiguity regarding ownership of specific refactoring tasks, which led to duplication of efforts or delays when roles were unclear.
- The team could have benefited from more structured documentation of decisions made during discussions, which would have reduced follow-up confusion and avoided the need for clarification in future meetings.

C. Moving Forward

What Should Be Done:

- Assign clear ownership for specific parts of the codebase to ensure accountability and reduce the risk of duplicated work.
- Provide additional resources and workshops on JavaScript for team members with less experience, ensuring they are better equipped to contribute equally in the future.
- Improve documentation processes for team decisions and meetings, so that everyone is aligned and fully informed, even when certain members are unable to attend.

---

4. Lessons Learned

Technical Insights:

- Refactoring large, complex libraries like Bigfoot requires deep knowledge of both the original codebase and the target language (JavaScript). A thorough understanding of how each language handles critical features like DOM manipulation, dependency management, and event handling is crucial.
- Learning to effectively analyze and restructure an unfamiliar repository is a valuable skill, but one that is often under-practiced. This project gave the team an opportunity to sharpen this ability, which will be useful in future work.

Team Collaboration:

- Regular check-ins and collaborative discussions were essential for keeping everyone aligned. However, more structured task assignments could have prevented task overlaps and improved efficiency.
- Instead of rescheduling meetings when conflicts arose, an alternative approach (like keeping the meeting on schedule and sharing detailed minutes with absent members) proved effective in minimizing disruption.

Process Improvements:

- Setting clearer milestones and assigning concrete task owners at the start of the project would have helped the team manage its time and resources more efficiently. Better planning could have mitigated some of the time management challenges experienced during the warm-up phase.

---

5. Action Items for Improvement

Immediate Actions:

- Improve the documentation process for ongoing tasks and decision-making, ensuring that detailed meeting notes are available for everyone.
- Allocate time for team members less familiar with JavaScript to deepen their knowledge through team-based learning sessions and independent projects.
- Encourage members to take on smaller independent projects over weekends to solidify their understanding of key concepts, particularly in web development.

Long-Term Changes:

- Consider implementing pair programming or mentoring sessions to help less experienced team members build their skills in future phases of the project.
- Define clearer task ownership and establish a process for monitoring progress to avoid bottlenecks and ensure that everyone is contributing effectively.

---

6. Closing Thoughts

Final Reflections:

- Despite the challenges faced, the team made significant progress during the warm-up phase. Each member gained valuable experience, not only in refactoring but also in team collaboration, repository management, and working with modern JavaScript.
- Through this project, the team developed a strong foundation for communication and collaboration, which will make future projects smoother and more efficient.

What's Next?

- The next steps involve solidifying the team's understanding of the refactored code and optimizing the workflow for the upcoming phases of the project. By addressing the challenges outlined and implementing the suggested improvements, the team is poised for continued success in following formal assignment.