Streamlining Web Feature Deprecation for Enhanced User

Anila Kosaraju
University of Massachusetts, Lowell
Anila_Kosaraju@student.uml.edu

***Abstract*—Web applications are constantly evolving, introducing new features to meet user demands and improve functionality. However, as these features age, security concerns may arise, necessitating their deprecation. The current process of feature deprecation across various web browsers lacks coordination, resulting in compatibility issues and prolonged security threats. This paper proposes a progressive security mechanism to streamline the deprecation of web features, providing enhanced security guarantees for users.**

1. INTRODUCTION

Web applications have become integral to our daily lives, offering a plethora of features to enhance user experience. However, as technology advances, so do the threats to user security. Many web features, once considered innovative, may become obsolete or pose security risks over time. The current approach to feature deprecation suffers from a lack of coordination among different browser vendors, leading to a fragmented user experience and delayed mitigation of security vulnerabilities.

2. THE CURRENT STATE OF WEB FEATURE DEPRECATION

Web feature deprecation is currently managed independently by each browser vendor, leading to a lack of cohesion in the deprecation process. Each vendor follows its own timeline and methodology for deprecating features, resulting in a fragmented user experience. Compatibility issues arise as developers struggle to keep pace with disparate deprecation schedules, ultimately leaving users exposed to security vulnerabilities. Furthermore, the absence of standardized communication channels between browser vendors and developers exacerbates the challenges. Developers often discover feature deprecations reactively, hindering their ability to proactively address potential issues. This lack of coordination not only complicates the development process but also poses significant security risks.

3. THE NEED FOR STREAMLINED DEPRICATION

The necessity for a streamlined approach to web feature deprecation stems from the shortcomings of the current disjointed process. A coordinated strategy is essential to mitigate the following challenges:

3.1 Security Vulnerabilities: Outdated features left unaddressed due to fragmentation in deprecation timelines create extended windows of exposure, leaving users susceptible to security threats.

3.2 Developer Friction: The current lack of coordination imposes a burden on developers who must navigate varying deprecation schedules. A streamlined approach provides predictability and clarity, reducing the learning curve and facilitating a smoother adaptation process.

3.3 User Experience Impact: Inconsistencies across browsers resulting from different deprecation timelines can lead to compatibility issues and an overall compromised user experience. A unified approach ensures a harmonious web environment for users.

To overcome these challenges, a progressive security mechanism is proposed to establish a standardized and collaborative framework for web feature deprecation, enhancing both security and user experience.

4. PROGRESSIVE SECURITY MECHNISM

The progressive security mechanism involves a phased approach to feature deprecation, allowing developers and users to adapt gradually. This includes:

4.1. Early Warning System: Implementing a standardized early warning system to notify developers about upcoming feature deprecations. Browser vendors can collaborate to issue warnings well in advance, giving developers sufficient time to update their applications.

4.2. Coordinated Deprecation Timeline: Establishing a shared timeline for deprecating specific features across major browsers. This ensures a synchronized and predictable process, minimizing disruptions for developers and users.

Table 1. Proposed Coordinated Deprecation Timeline (CDT)

Feature	Browser Vendor A	Browser Vendor B	Browser Vendor C
Feature X	Q1 2024	Q2 2024	Q1 2024
Feature Y	Q3 2024	Q3 2024	Q4 2024
Feature Z	Q2 2025	Q1 2025	Q3 2025

4.3. Mitigation Strategies: Providing clear guidelines and mitigation strategies for developers to transition away from deprecated features. This includes documentation, tooling support, and educational resources to facilitate a smooth migration process.

Table 2: Mitigation Strategies (MS) Checklist

Deprecated Feature	Mitigation Strategy 1: Documentation	Mitigation Strategy 2: Code Examples	Mitigation Strategy 3: Automated Tools
Feature X	Link to Documentation	Code Snippets	Migration Tool Available
Feature Y	Documentation PDF	GitHub Repository	No Automated Tool
Feature Z	Online Video Tutorial	Sample Projects	Migration Guide

5. BENEFITS OF THE PROPOSED MECHANISM

Implementing a progressive security mechanism for web feature deprecation offers several benefits:

5.1. Enhanced User Security: By streamlining the deprecation process, security vulnerabilities associated with outdated features are addressed promptly, reducing the window of exposure for users.

5.2. Improved Developer Experience: A coordinated approach provides developers with a predictable timeline and comprehensive resources, making it easier to adapt to changes without compromising functionality.

5.3. Consistent User Experience: Users benefit from a more consistent experience across different browsers, as deprecated features are phased out uniformly.

6. IMPLEMENTATION OF THE PROGRESSIVE SECURITY MECHANISM

To realize the proposed progressive security mechanism, a collaborative effort among major browser vendors is essential. This can be achieved through the establishment of a Web Standards Security Council (WSSC), consisting of representatives from leading browser development teams. The WSSC would serve as a platform for communication, coordination, and decision-making regarding feature deprecation.

6.1. Early Warning System: The WSSC would oversee the implementation of an Early Warning System (EWS), a centralized platform that alerts developers about upcoming deprecations. Browser vendors would contribute to this system by providing insights into their deprecation roadmaps. The EWS would send notifications, emails, and in-browser alerts, ensuring developers are well-informed well in advance

6.2. Coordinated Deprecation Timeline: One of the primary challenges in feature deprecation is the lack of synchronization among browsers. The WSSC would work to establish a Coordinated Deprecation Timeline (CDT) that outlines when specific features will be deprecated across all major browsers. This timeline would be updated regularly, providing developers with a clear schedule for adjustments.

6.3. Mitigation Strategies: The WSSC, in collaboration with developer communities, would develop comprehensive Mitigation Strategies (MS) for each deprecated feature. These strategies would include documentation, code examples, and automated migration tools to facilitate a smooth transition. Additionally, the WSSC could organize webinars and workshops to educate developers on best practices during the deprecation process.

7. FLEXIBILITY AND DEVELOPER FEEDBACK

Recognizing the diverse nature of web development, the proposed mechanism should allow for flexibility. Developers should have the opportunity to provide feedback on deprecation proposals and express concerns or suggestions. A feedback loop, integrated into the EWS, would enable developers to voice their opinions, fostering a sense of community involvement in the deprecation process.

8. CASE STUDY: SUCCESSFUL FEATURE DEPRECATION IMPLEMENTATION

To illustrate the effectiveness of the proposed progressive security mechanism, a case study can be conducted on the deprecation of a widely-used but outdated web feature. This case study aims to showcase how the collaborative efforts of the Web Standards Security Council (WSSC), the Early Warning System (EWS), the Coordinated Deprecation Timeline (CDT), and Mitigation Strategies (MS) contribute to a seamless transition for developers and an enhanced security posture for users.

8.1. Selection of the Deprecated Feature:
- Identify a feature that is widely used but has become a security concern or is outdated.
- Ensure that the feature is cross-browser and used in diverse web applications.

8.2. Formation of the Web Standards Security Council (WSSC):
- Detail the process of establishing the WSSC with representatives from major browser vendors.
- Describe the collaborative decision-making process within the council.

8.3. Implementation of the Early Warning System (EWS):
- Demonstrate how the EWS operates in providing advanced notifications to developers.
- Highlight the channels used for notifications, such as in-browser alerts, emails, and documentation updates.

8.4. Coordinated Deprecation Timeline (CDT):
- Present the shared timeline developed by the WSSC for deprecating the selected feature across major browsers.
- Discuss how the CDT is regularly updated and communicated to developers.

8.5. Mitigation Strategies (MS) Execution:
- Provide a detailed checklist of mitigation strategies developed for the deprecated feature.

- Include links to documentation, code examples, and any available automated tools.
- Share insights into webinars and workshops organized by the WSSC to educate developers on best practices during the deprecation process.

8.6 Developer Engagement and Feedback Loop:

- Emphasize the importance of developer engagement in the deprecation process.
- Showcase the feedback loop integrated into the EWS, allowing developers to voice concerns and suggestions.
- Highlight how the WSSC incorporates valuable feedback into the ongoing deprecation strategy.

8.7. Security and User Experience Impact Analysis:

- Evaluate the impact of the coordinated deprecation on user security.
- Analyse how the streamlined approach contributes to a more consistent user experience across browsers.
- Provide statistical data on the reduction of security vulnerabilities associated with the deprecated feature.

8.8. Lessons Learned and Future Adaptations:

- Summarize the lessons learned from the case study.
- Discuss any unexpected challenges and how they were addressed.
- Propose future adaptations or improvements to the progressive security mechanism based on the case study findings.

9. FUTURE CONSIDERATIONS

As technology continues to evolve, the WSSC would play a crucial role in adapting the progressive security mechanism to future challenges. This may involve the incorporation of machine learning algorithms to predict potential security risks, or the exploration of automated migration tools to further simplify the transition process for developers.

10. CONCLUSION

In conclusion, the proposed progressive security mechanism represents a significant step towards streamlining web feature deprecation and enhancing user security. By fostering collaboration among browser vendors, providing early warnings, coordinating deprecation timelines, and offering comprehensive mitigation strategies, the mechanism aims to create a more secure, consistent, and developer-friendly web environment. The establishment of a Web Standards Security Council ensures sustained efforts and adaptability to the evolving landscape of web technologies, ultimately benefiting all stakeholders in the web ecosystem.

11. REFERENCES

[1] T. Van Goethem and W. Joosen, "Towards Improving the Deprecation Process of Web Features through Progressive Web Security," *2022 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2022, pp. 20-30, doi: 10.1109/SPW54247.2022.9833872.

[2] A. Mirian, N. Bhagat, C. Sadowski, A. Porter Felt, S. Savage and G. M. Voelker, "Web Feature Deprecation: A Case Study for Chrome," *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, QC, Canada, 2019, pp. 302-311, doi: 10.1109/ICSE-SEIP.2019.00044.

[3] F. R. Cogo, G. A. Oliva and A. E. Hassan, "Deprecation of Packages and Releases in Software Ecosystems: A Case Study on NPM," in *IEEE Transactions on Software Engineering*, vol. 48, no. 7, pp. 2208-2223, 1 July 2022, doi: 10.1109/TSE.2021.3055123.