Richtlinie zum „Software-Sprint“

[BrowserEQv2 – BrowserEQ v2]

Schlussbericht

Zuwendungsempfänger:

Berrak Nil Boya

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS24S27 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Was war Deine Motivation? Welches Problem wolltest Du mit Deinem Projekt lösen? Wie war die geplante Vorgehensweise zur Problemlösung (auch Angabe der wichtigsten Meilensteine)?

Meine Motivation für BrowserEQ entstand aus dem Bedürfnis, die Audioqualität von Livestreams und Online-Medien verbessern zu können. Viele Streams, sei es während Literaturfestivals oder Spiele-Fundraising-Veranstaltungen, leiden unter schlechter Audioqualität mit Rauschen oder störenden Geräuschen, was das Verständnis und die Teilnahme erschwert.

Das Ziel von BrowserEQ v2 war es, auf dem Erfolg der ersten Version aufzubauen und eine umfassendere, leistungsfähigere Audio-Equalizer-Weberweiterung zu entwickeln, die allen Nutzern - unabhängig von ihrem technischen Wissen - hilft, Audioinhalte im Browser zu optimieren.

Die wichtigsten Meilensteine waren:

1. Neugestaltung der UI/UX für neue Funktionen

2. Umstellung von Vanilla JavaScript auf Vue.js 3 mit TypeScript

3. Implementierung von Voreinstellungen (Speichern/Laden)

4. Hinzufügen von Mono/Stereo-Downmix und Panning-Optionen

5. Integration verbesserter Zugänglichkeit

6. Hinzufügen von Echtzeit-Aufnahme/Re-Sampling

7. Erweiterung der Browser-Kompatibilität (Firefox, Edge, Safari)

Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint“

Wer ist die Zielgruppe für Deine Lösung? Wie profitiert sie von den Ergebnissen Deines Projekts? Welche Bezüge gibt es zu den Themenfeldern und Zielen des Software Sprints?

BrowserEQ v2 trägt zum Bereich "Civic Tech" bei, indem es die digitale Teilhabe fördert. Die Weberweiterung macht Online-Kommunikation, Community-Beteiligung und audiovisuelle Medien für Menschen mit unterschiedlichen Bedürfnissen zugänglicher.

Insbesondere unterstützt das Projekt Menschen mit Hörproblemen oder sensorischen Empfindlichkeiten, die ohne solche Tools möglicherweise von wichtigen Online-Diskussionen, Bildungsangeboten oder kulturellen Veranstaltungen ausgeschlossen wären.

Als Open-Source-Lösung ermöglicht BrowserEQ zudem die gemeinschaftliche Weiterentwicklung und stellt eine Alternative zu proprietären Audio-Lösungen dar, die oft mit hohen Kosten verbunden sind.

Ausführliche Darstellung der Ergebnisse

Welche konkreten Ergebnisse hast Du erzielt? Konnten alle Meilensteine erreicht werden? Welche zusätzlichen Erkenntnisse hast Du aus der Projektarbeit gewonnen, auch im Hinblick auf die Begleitung durch die Open Knowledge Foundation?

BrowserEQ v2 ist eine umfassende Audio-Equalizer Web-Erweiterung, die alle Arten von webbasierten audiovisuellen Inhalten für Benutzer zugänglich und verständlich macht. Die Anwendung erfasst den Audiostream der aktuellen Browser-Registerkarte und ermöglicht dem Benutzer, diesen Stream mit einer benutzerfreundlichen Oberfläche zu steuern.

Die wichtigsten technischen Errungenschaften und Verbesserungen gegenüber der ersten Version sind:

1. **Vollständige Neuimplementierung in Vue.js 3 mit TypeScript anstelle von Vanilla JavaScript, was die Codequalität und Wartbarkeit verbessert**
2. **Optimierung der Audio-Verarbeitung für bessere Leistung durch effizientere Algorithmen und Datenstrukturen**
3. **Einführung von speicherbaren und ladbaren Voreinstellungen mit einer Sammlung von Standardvoreinstellungen**
4. **Implementierung von Mono/Stereo-Downmix und Panning-Optionen für verbesserte Audiosteuerung**
5. **Umfassende Verbesserung der Audio-Zugänglichkeit**
6. **Integration von Echtzeit-Aufnahme und Re-Sampling-Funktionen**

Im Bereich der Browser-Kompatibilität wurden einige Herausforderungen identifiziert. Nach eingehender Analyse der verschiedenen Browser-APIs musste die ursprüngliche Strategie angepasst werden. Aufgrund zahlreicher Implementierungsprobleme mit der

GetUserMedia API in Chrome wurde die Entscheidung getroffen, stattdessen die TabCapture API zu verwenden. Diese Entscheidung hatte jedoch Auswirkungen auf die Firefox-Kompatibilität, da Firefox derzeit keine Unterstützung für die TabCapture API bietet.

Für eine vollständige Firefox-Implementierung wäre eine signifikante Umstrukturierung des Audio-Stream-Erfassungsteils notwendig, um stattdessen die GetUserMedia API zu nutzen. Diese Herausforderung wird in den weiteren Entwicklungsphasen adressiert werden, da sie eine grundlegende Codemodifikation erfordert. Trotz dieser technischen Hürde konnten die meisten geplanten Meilensteine erfolgreich umgesetzt werden, und die Erweiterung funktioniert zuverlässig auf Chrome.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Welcher Nutzen ergibt sich für die Zielgruppe aus den Ergebnissen Deines Projekts? Welche weiter-gehenden Effekte ergeben sich aus der Open-Source-Stellung der Ergebnisse? Gibt es Ideen für die Weiterentwicklung Deiner Lösung und Pläne zu deren Umsetzung?

Hat die Arbeit in dem Projekt Dich in Deiner persönlichen, fachlichen Weiterentwicklung unterstützt?

BrowserEQ v2 richtet sich an alle, die Webbrowser zum Streamen audiovisueller Inhalte verwenden, insbesondere an Personen ohne tiefgreifende Audiokenntnisse. Die Zielgruppe profitiert von:

1. **Verbesserter Audioqualität bei Livestreams, Online-Meetings und Medienwiedergabe**
2. **Zugänglichkeitsoptionen für Menschen mit Hörproblemen oder sensorischen Empfindlichkeiten**
3. **Benutzerfreundlicher Oberfläche, die keine Audio-Expertise erfordert**
4. **Möglichkeit, störende Geräusche zu reduzieren, die Konzentration oder Gesundheit beeinträchtigen können**

Die Open-Source-Stellung ermöglicht:

- **Gemeinschaftliche Weiterentwicklung und Anpassung an spezifische Bedürfnisse**
- **Transparenz und Vertrauen in die Datensicherheit, da keine versteckten Funktionen implementiert werden können**
- **Alternative zu kostenpflichtigen, proprietären Audio-Lösungen**
- **Wissensaustausch und Bildung im Bereich der Audio-Webentwicklung**

Zukünftige Weiterentwicklungsmöglichkeiten umfassen:

- **Benutzer-Voreinstellungen für spezifische Anwendungsfälle**
- **Verbesserte Visualisierung von Audiofrequenzen**
- **Kollaborative Funktionen zum Teilen von Voreinstellungen**
- **Vollständige Firefox-Unterstützung durch Anpassung der Audio-Capture-Methodik**
- **Weitere Barrierefreiheitsfunktionen wie Tastaturnavigation und WebMIDI API-Unterstützung**

Das Projekt hat meine persönliche und fachliche Entwicklung in den Bereichen moderne Web-Frameworks, Audio-APIs und barrierefreie Webentwicklung maßgeblich gefördert.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Gab es Arbeiten bzw. Lösungsansätze, die nicht weiter verfolgt wurden? Was waren die Hinter-gründe, und wie bist Du alternativ vorgegangen?

Bei der Implementierung der Audio-Verarbeitung traten zunächst Kompatibilitätsprobleme mit älteren Browserversionen auf. Der ursprüngliche Ansatz, eine vollständige Abwärtskompatibilität zu gewährleisten, wurde nach eingehender Analyse aufgegeben zugunsten einer fokussierten Unterstützung moderner Browser mit Fallback-Optionen.

Zudem wurde ein anfänglicher Versuch, ein komplexeres System zur dynamischen Kompression zu implementieren, aufgrund von Performance-Problemen vereinfacht. Stattdessen wurde eine effizientere Lösung mit vordefinierten Kompressionseinstellungen entwickelt.

Eine Lösung für das schwerwiegende Problem, den Zustand der Audioverarbeitung zu erhalten, konnte zu diesem Zeitpunkt nicht gefunden werden. Mit Manifest v3 in Google Chrome-Erweiterungen ist es nicht mehr möglich, Audioaufnahme und -verarbeitung in Hintergrundskripten durchzuführen, wie es früher möglich war. Das bedeutet, dass die aktuelle Audioverarbeitung nur so lange angewendet wird, wie das Popup geöffnet ist. Eine Lösung dafür war die Erstellung einer alternativen, kompakten Benutzeroberfläche, die der Benutzer entweder über den Optionsbildschirm der Erweiterung oder während der Laufzeit über einen UI-Schalter auswählen kann.

Folgende Lösungsansätze wurden recherchiert und ausprobiert:

- **Erstellung eines schwebenden Popup-Fensters, das auch beim Verlust des Fokus im Vordergrund bleibt**
- **Verwendung der Sidepanel-API**
- **Verlagerung der Audioverarbeitung in die Content-Scripts**
- **Und viele weitere Ansätze, die jedoch aufgrund zahlreicher Bugs oder Implementierungsdetails der Browser-Hersteller zu keiner erfolgreichen Lösung führten.**

Weitere Informationen zu diesen Herausforderungen finden sich in folgenden Diskussionen und Issues:

- **Chromium Extensions: MV3 getUserMedia Audio Capture Issues**
- **Chromium Extensions: Audio Capture in MV3**
- **Chromium Issue 40184152: Audio Processing Limitations**
- **Chrome Extensions Samples: MV3 Audio Capture Issue**
- **Reddit Discussion: Capturing Audio with MV3**
- **Chromium Issue 40184923: Audio Processing Comment**
- **Chromium Issue 40094084: Audio Stream Handling**
- **Chromium Issue 40504498: Extension Audio Processing**

Wie bereits im Abschnitt "Ausführliche Darstellung der Ergebnisse" erwähnt, traten auch bei der Browser-übergreifenden Kompatibilität erhebliche Schwierigkeiten auf, insbesondere bei Firefox. Die Notwendigkeit, auf die TabCapture API anstelle der

GetUserMedia API umzusteigen, führte zu Kompatibilitätsproblemen, da Firefox derzeit keine Unterstützung für die TabCapture API bietet. Weitere technische Details zu diesen Herausforderungen können unter StackOverflow: Error Capturing Tab Content in Chrome Extension MV3 gefunden werden.

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Wo können sich Interessenten detailliert über Deine Projektergebnisse informieren (z.B. Webseite, GHitHub, Veröffentlichungen)?

Interessierte können sich über das Projekt informieren unter:

- GitHub-Repository: https://github.com/berraknil/BrowserEQ-V2
- Projektwebseite: https://berraknil.github.io/BrowserEQ-V2/
- Dokumentation und Nutzerhandbuch: https://github.com/berraknil/BrowserEQ-V2?tab=readme-ov-file#browsereq-v2---audio-verbesserung-für-deinen-browser

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Gab es im Projektverlauf Ereignisse, die eine Anpassung der Planung erforderlich machten – z.B. Mehr- oder Minderaufwand bei der Bearbeitung von Teilaufgaben?

Das Projekt wurde weitgehend innerhalb der geplanten 950 Arbeitsstunden umgesetzt. Die Implementierung der browserübergreifenden Kompatibilität erforderte neben der Neuschreibung der Kernlogik aufgrund der Änderungen an den Hintergrundskripten in Manifest v3 und der Tatsache, dass die GetUserMedia-API aufgrund von Fehlern nicht verwendet werden konnte, einen erhöhten Aufwand, da verschiedene Browser unterschiedliche Implementierungen der Audio-APIs haben. Diese Herausforderung wurde durch eine Umstrukturierung des Arbeitsplans und eine Priorisierung der wichtigsten Funktionen kompensiert, ohne das Gesamtbudget zu überschreiten.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Gab es Entwicklungen anderer Personen oder Institutionen, die Einfluss auf Deine Arbeiten und die Zielsetzung hatten? Wenn ja, worin bestand dieser und wie bist Du damit umgegangen?

Während der Projektlaufzeit wurden ähnliche Browser-Erweiterungen veröffentlicht, jedoch keine mit dem gleichen Funktionsumfang und Open-Source-Ansatz. Die Weiterentwicklung von Web-Audio-Standards und Browser-APIs hatte positiven Einfluss auf die technische Umsetzung des Projekts. Durch kontinuierliches Monitoring der Entwicklungen konnten neueste Standards und Best Practices implementiert werden, was die Qualität und Zukunftssicherheit der Anwendung verbessert hat.