UNIVERSITÉ
CÔTE D'AZUR

Université Côte d'Azur

Devtools WebAudio - 2019 https ://github.com/FontanyLegall-Brandon/devtools-WebAudio

Auteur :
Brandon Fontany–Legall

17 juin 2019

Table des matières

Chapitre 1

Présentation du projet

1.1 Présentation du sujet

Dans le cadre de la modification du navigateur Mozilla avec notamment sa réécriture en language Rust [1], il nous est demandé de migrer l'outil WebAudio présent dans Mozilla vers une extension web externe au navigateur. Cet outil permet d'afficher, sous forme de graph, les éléments Webaudio d'une page dans le debugger du navigateur dans l'objectif d'aider les développeurs dans leurs tâches de conception.
Pour résumé, il faut créer un code permettant d'analyser du code WebAudio et de l'injecter à l'aide d'une extension dans les pages web permettant l'observation des différents éléments audio. Il faut ensuite les traduire en graph pour affichage dans le debugger.

1.2 Présentation du Groupe

Dans le cadre de ce projet, je suis le seul(Brandon Fontany-Legall) en conception et mise en oeuvre du projet. Je suis accompagné de Mr Michel Buffa, professeur titulaire de l'université Côte d'Azur. Pour finir, en mentors, nous avons Luca Greco, Paul Arduino, Nick Fitzgerald et Yulia Startsev qui sont des employés de Mozilla qui sont là pour répondre aux possibles questions.

1. Rust est un nouveau langage de programmation open source créé par Mozilla et une communauté de volontaires, conçu pour aider les développeurs à concevoir des applications ultra-rapides et sécurisées

Chapitre 2

État de l'art

2.1 WebAudio

WebAudio est une API client-side permettant l'ajout de capacités audio avancées pour les spécifications <audio> et <vidéo> qui ont été introduites en HTML 5.

L'API ainsi que son design a a été proposée et discuté par Chris Rogers et Robert O'Callahan en 2010[12] à la suite de la proposition de l'Audio Data API(Mozilla)[1] formulée par Dave Humphrey. Une premiere spécification est alors mise en place en 2011 sur le site du W3C[6] pour une premiere implementation sur Firefow/Webkit browsers en 2012. Pour finir, l'AudioWorklet(voir 2.1.3) fait son apparition sur Google Chrome en 2018.

2.1.1 Intérêt de WebAudio

WebAudio est une API extrêmement jeune permettant, du fait qu'elle soit client-side, de faire de l'audio de façon **fluide sans latence** sur un navigateur web. Il devient notamment possible de faire du son uniquement à l'aide de son navigateur internet comme le mixage de son, la création de son, le branchement sur son ordinateur d'appareils musicaux comme des guitares avec comme simple logitiel un navigateur internet,etc... Les possibilités sont multiples.

FIGURE 2.1 – Audio Graph Example

Dans ce contexte, il devient important d'avoir des outils qui permettent d'aider à la programmation et notamment via le debugger du navigateur directement.

2.1.2 AudioNode

Dans WebAudio, les opérations sont définies par des noeuds formant des graphs audio. Il existe un grand panel de noeuds prédéfini dans WebAudio comme par exemple le Gain, Filter, Compressor etc. mais aussi des noeuds spécifiques comme les noeuds sources(file, stream, sound card...) et les noeuds de destination(speakers). Voici un exemple de graphique WebAudio :

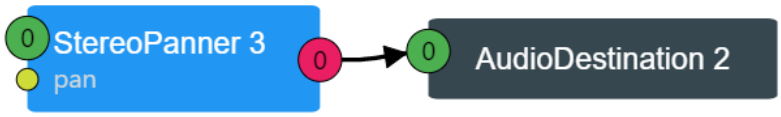

FIGURE 2.2 – Audio Graph Example

2.1.3 AudioWorklet

L'AudioWorklet[2] est le nouveau modèle pour traiter l'audio. Il permet aux développeurs de fournir des scripts (Javascript ou WebAssembly) pour traiter l'audio au fil du temps avec des noeuds audio personnalisés. Plus précisément, le modèle introduit des noeuds spécifiques AudioWorklet qui sont exécutés par des AudioWorkletProcessor exécutés dans un thread spécialement alloué pour l'audio avec une latence faible (exécuté en utilisant des buffers de 128 frames).

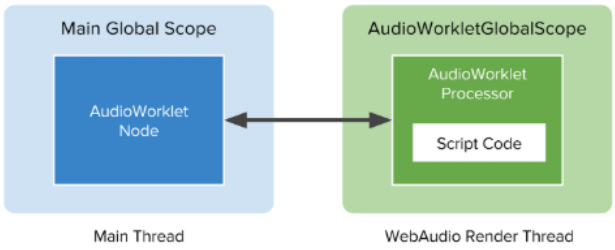

FIGURE 2.3 – Audio Worklet

Ce qui est intéressant est le fait que nous pouvons définir des noeuds personnalisés en définissant nos propres noeuds avec notre propre processor dont voici ci-dessous une figure explicative sans rentré trop dans les détails.

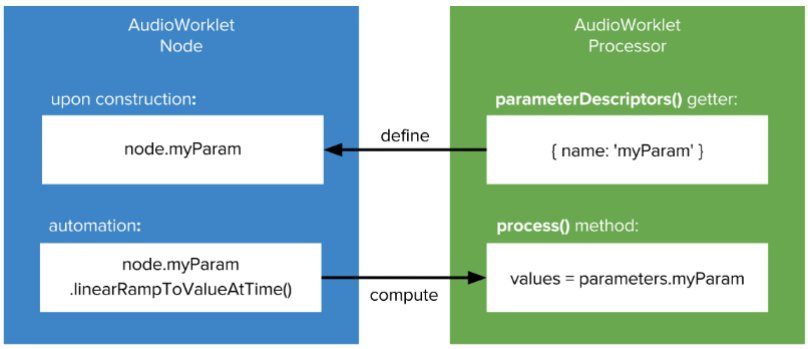

FIGURE 2.4 – Custom Audio Worklet

2.2 Web Extension

Les extensions Web[5] sont des outils externes permettant d'étendre/modifier les capacités d'un navigateur. Ils sont dans la plupart du temps créé par des indépendants n'ayant pas de lien direct avec le navigateur en question. Les extensions pour Firefox sont réalisés en Javascript via l'utilisation de l'API WebExtensions. Trois outils pour notre projet nous intéressent plus particulièrement : l'api devtools, le script de contenu et le script d'arrière-plan.

2.2.1 API Devtools

Dans notre cas, nous recherchons à enrichir le débugger du navigateur avec notre outil de visualisation de graph audio. L'api devtools rentre donc en jeu nous permettant d'étendre les outils de développement intégrés

du navigateur. La page devtools qui est tout simplement une page html dans notre extension va alors être chargée lorsque les devtools du navigateur sont ouverts (ctrl + shift + i ou F12).

2.2.2 Script de contenu

Le script de contenu[3] est la partie de l'extension qui va s'exécuter dans le contexte de la page web actuel. Le script de contenue est totalement different du script d'arrière-plan(voir 2.2.3) ou encore des scripts qui font partie du site. Le script de contenu peut accéder au contenu des pages et y être exécuté, c'est pourquoi il faut être extrêmement prudent sur les extensions que nous installons sur notre navigateur qui peuvent être très dangereux ! Après définissions du script et sa spécification dans le manifest.json, l'extension va demander au navigateur de charger le script de contenu lors du chargement de la page. Le script pourra ensuite s'exécuter et renvoyer les possibles donnés nécessaires au bon fonctionnement de l'extension.

2.2.3 Script d'arrière plan

Le script d'arrière-plan est un script qui est exécuté par le navigateur dès que le navigateur est lancé. Il va notamment être en charge du fonctionnement de l'extension à l'aide de l'API Web Extension.

2.2.4 Interaction

Il est important de comprendre que le script de contenu et le script d'arrière-plan n'ont donc pas la même portée. En effet, le script de contenue ne peut accéder qu'à un sous-ensemble des API WebExtension[4] mais peut accéder aux éléments de la page courante. De son côté, le script d'arrière-plan peut accéder à l'intégralité de l'API WebExtension mais ne peut ni accéder aux éléments de la page courante, ni à l'API devtools. Il va donc joueur un rôle d'intermédiaire entre le content-script et l'api webextension via l'utilisation de messages.

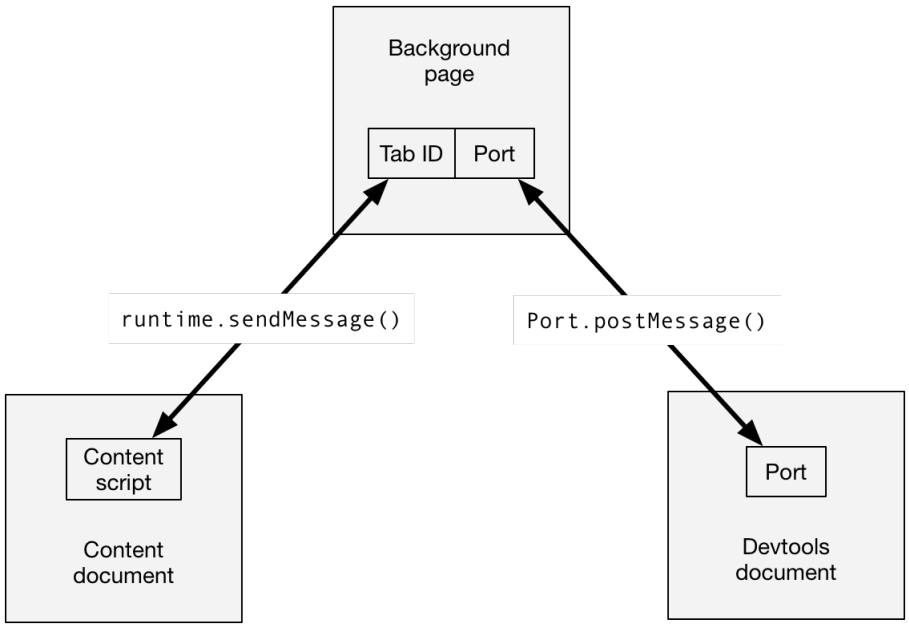

FIGURE 2.5 – Interaction[14]

2.3 CallWatcher

Le CallWatcher est une API propre à Mozilla avec laquelle nous pouvons enregistrer les appels de fonctions ainsi que les appels aux constructeurs. Ceci utilise notamment le patron de conception du décorateur(voir figure ci-dessous) qui redéfinit la fonction en appelant la fonction de base avec un petit ajout permettant de logger l'appel.

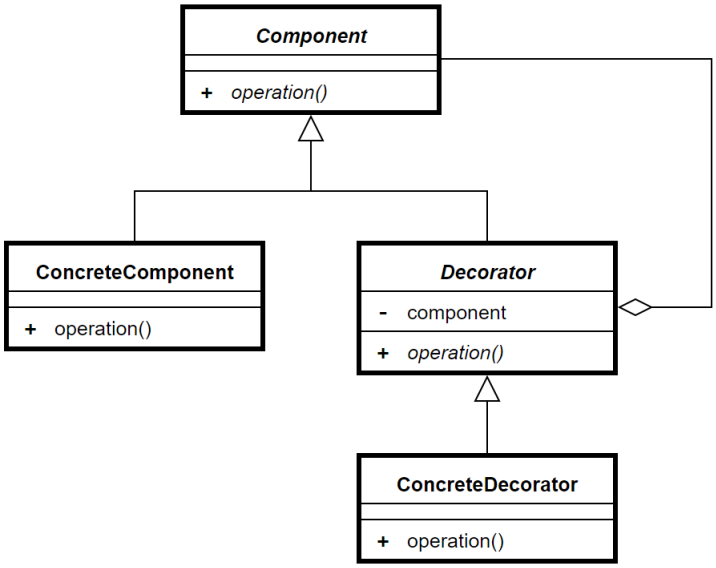

FIGURE 2.6 – Decorator

2.4 WeakReference

Les WeakReference[11], références faibles en français, est une feature expérimentale qui, pour l'anecdote, provient à l'origine d'une faille de sécurité de Mozilla permettant d'espionner le Garbage Collector[7]. Un objet à référence faible est un objet qui est weak reachable. C'est-à-dire que très peu d'objet pointe vers ce dernier voir pas du tout l'inscrivant dans la liste des prochains objets qui vont être finalize par le garbage collector. Nous stockons donc, si possible, les références des objets qui ont de grandes chances d'être finalisés par le garbage collector pour traquer le garbage collector. Ceci permet notamment dans notre cas de savoir quels noeuds Audio vont doivent être supprimer du graph Audio.

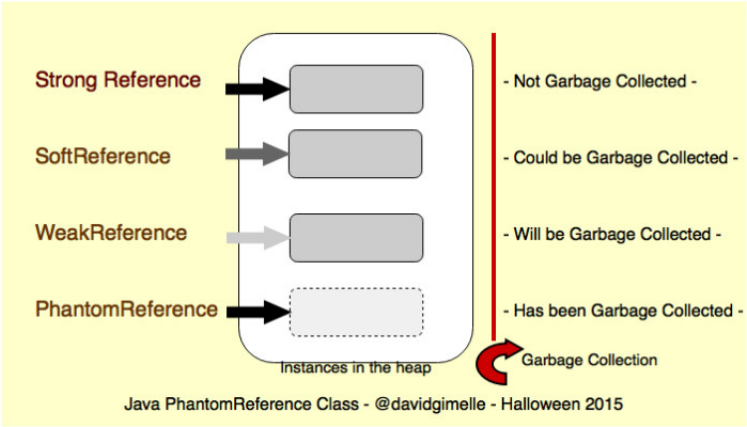

FIGURE 2.7 – Java Reference Class[13]

2.5 DagreD3

Dagre est une librairie JavaScript qui permet de créer des graphiques orientés client-side. Dagre-D3 est une librairie front-end à dagre basée sur le D3 renderer qui s'occupe du rendu graphique des graphiques. (Voir Dagre-D3 : https ://github.com/dagrejs/dagre-d3/wiki)

Chapitre 3

Travail effectué

3.1 Conception

En résumé, pour le projet, nous devons créer une extension web s'affichant dans le devtool du navigateur. Elle doit donc dans un premier temps recueillir les informations dans le content-script à l'aide du CallWtacher et l'envoyer au background-script qui fera des traitements sur le Json reçu. Une fois le json formalisé au format Dagre, le background doit l'envoyer au script du devtools qui va créer le devtools via l'API devtools.Une fois le devtools créer, il doit appeler Dagre-D3 avec le Json pour dessiner le graphique.Pour finir, les mêmes étapes devront être faites dans le cas des références faibles pour la synchronisation du graph en temps réel.

3.1.1 Création du graphique

La création du graphique suit dont le graphique ci-dessous, c'est-à-dire qu'il doit se faire via les données du CallWatcher, mises à jour par les WeakRefs, et dessiné par Dagre-D3.

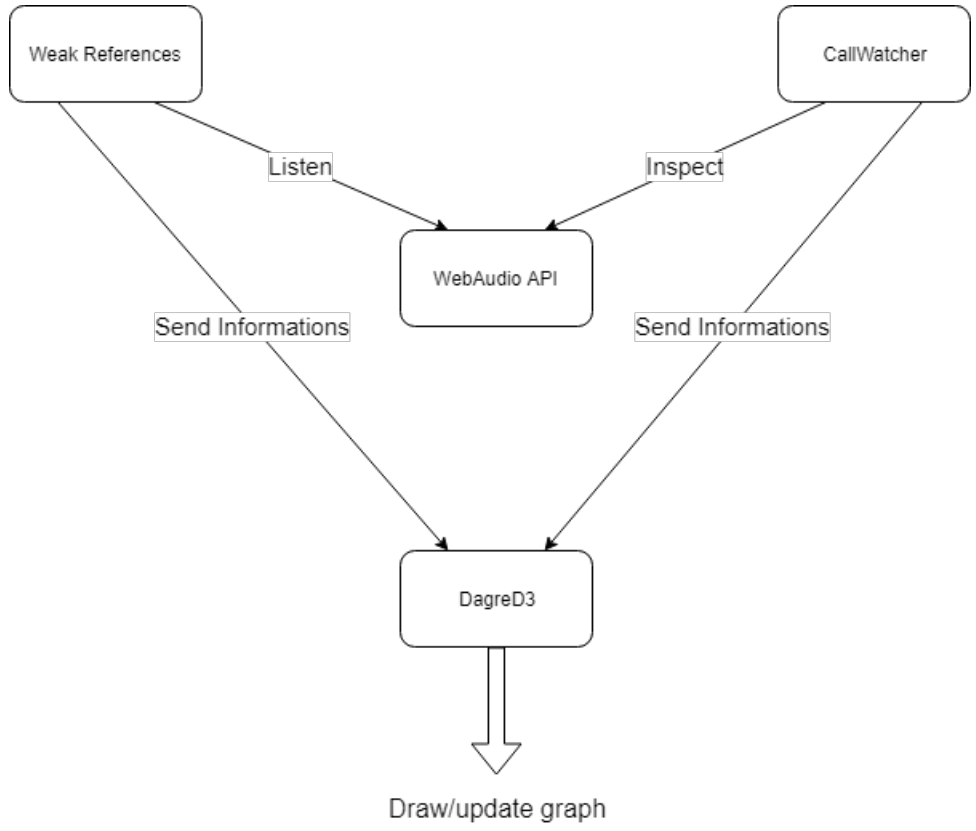

FIGURE 3.1 – Draw WebAudio Graph

3.1.2 Creéation du graphique dans le context du devtools

Voici le graphique résumé explicatif du travail à effectué pour l'extension :

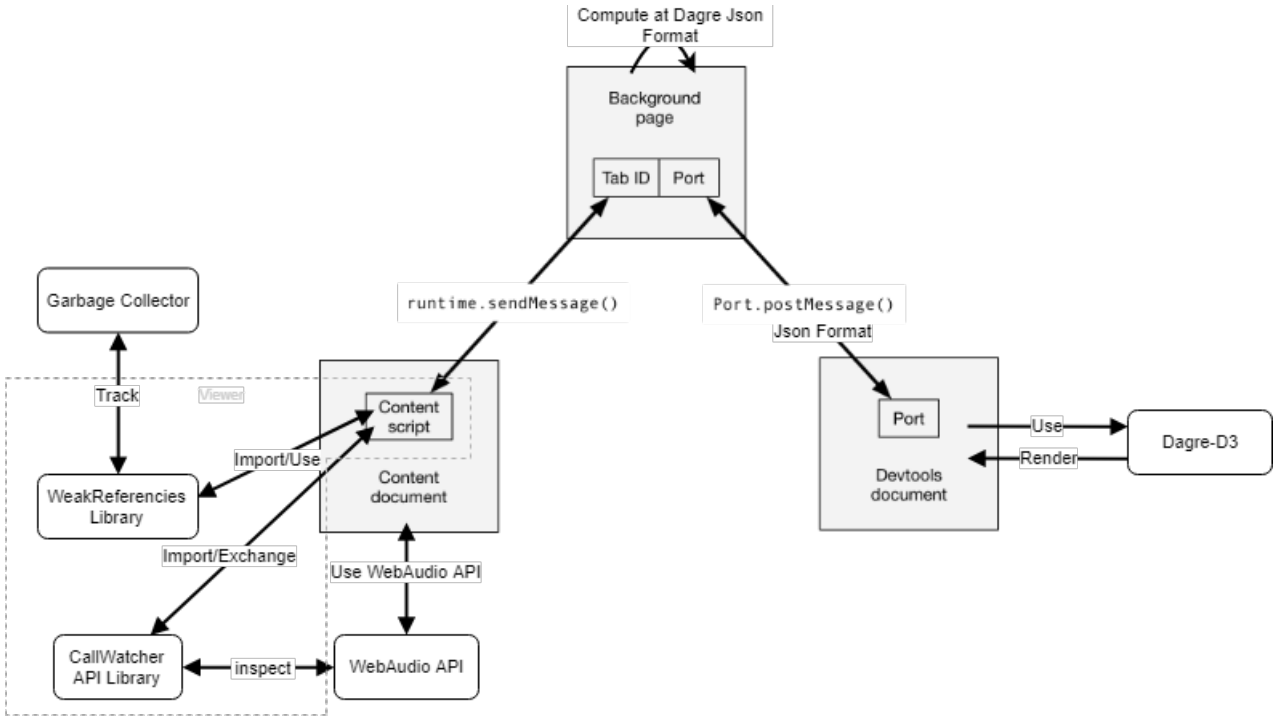

FIGURE 3.2 – Conception de l'extension

3.2 Les problèmes rencontrés

Nous avons eu énormément de rebondissements durant ce projet. En effet, les deux librairies pour permettre de faire le sujet se sont avéré actuellement non exploitable.

- Weakrefs qui devait nous permettre de "log" le garbage collector. Cependant, cette librairie est encore qu'à l'état de prototype qui n'est pas fonctionnel à l'heure actuelle. En effet, lors de la tentative d'utilisation du polyfill, nous nous sommes rapidement heurté à des bugs qui rendaient non utilisable ce dernier. Après discussions avec Mozilla, ils nous ont confirmé qu'il y avait un souci mais nous ont clairement dit qu'il n'avait pas le temps nécessaire pour corriger le problème(ce qui est compréhensible) et que nous devions corriger nous-mêmes les soucis du polyfill ce qui est assez loin de notre domaine de compétence actuelle.
- Callwatcher devait nous permettre de surveiller les appels WebAudio dans notre cas. Cependant, nous n'avons reçu qu'une version non utilisable pour les extensions web. En effet, la version web extension ne nous a jamais été envoyé car Mozilla n'arrivaient pas à contacter le créateur de CallWatcher.

Nous nous sommes donc retrouvés avec un sujet à devoir faire from scratch avec, à devoir faire, les librairies, l'extension web, le code de tracking donc des éléments qui sachant que de base nous avons un très faible niveau de connaissance au niveau du Javascript et encore moins au niveau des fonctionnalités poussées de ce dernier comme les décorateurs par exemple pour faire des éléments qui nous ont été présenté comme fourni noralement.

3.3 Les gists

Une grande partie de la conception a été effectué dans des gists qui sont tous présent dans le repositorie. Nous avons notamment :

- La confection d'un WebAudio pour effectué les différents test de conception dessus
- Une conception basique du Weakref dans lequel nous créons énormément de garbage pour récupéré des références faibles
- Un premier essaie de traque avec les Fonction.prototype
- Un premier essaie de traque avec les proxies
- Un intercepteur de constructeur avec décorateur
- La traque des connections entre les audionodes

— Un premier graph avec Dagre-D3
— Et enfin une première version du viewer

La première version du viewer amène la création du viewer(voir 3.5) dont le traqueur est totalement composé de décorateurs(voir la figure 2.6)

3.4 l'extension web

Au moment du rapport, l'extension web est en version 0.1.4 comprenant une première implementation demo de l'extension. Il y a notamment :

— La création de la partie devtools avec l'API devtools
— La création du manifest
— La création d'une partie du background script pour la création du devtools
— La création de la partie paramètres et de la partie graph dans le devtool
— CSS minimal à améliorer dans le futur
— La mise en place de Dagre-D3 pour l'affichage du graph
  — Prise en compte du zoom pourzoomer sur le graph
  — Prise en compte des cliques sur les noeuds

Cependant, cette version ne comprend pas toute la partie viewer(weakref, callwatcher....voir 3.5) qui a été séparé dans son propre module pour des raisons de simplicité de conception. En effet, la décision a été prise de travailler sur l'extension et le viewer de façon séparée ne respectant pas la méthode agile au vu de la difficulté à faire la conception du viewer étant le point le plus important du projet. Les deux seront merge une fois le viewer complètement opérationnel.

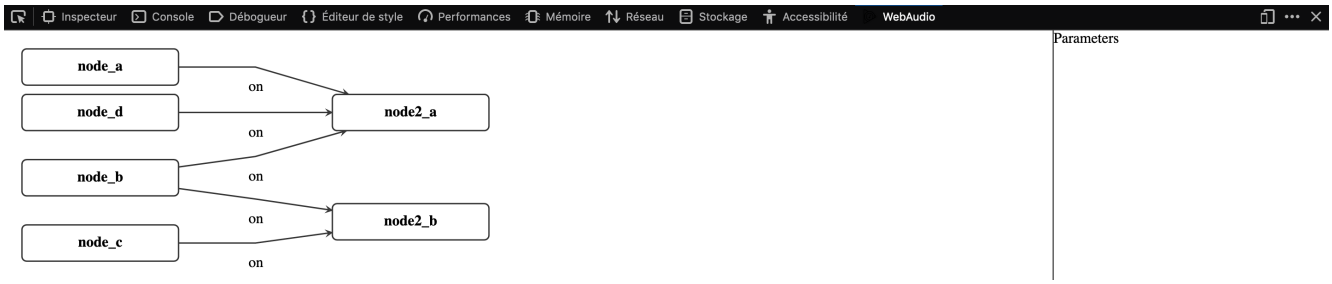

FIGURE 3.3 – Web Extension 0.1.4

3.5 Le viewer

Le viewer est la partie centrale de l'extension et a été fait séparément pour la simplifier sa conception et contient une partie graphique dont voici une première preview à vide.

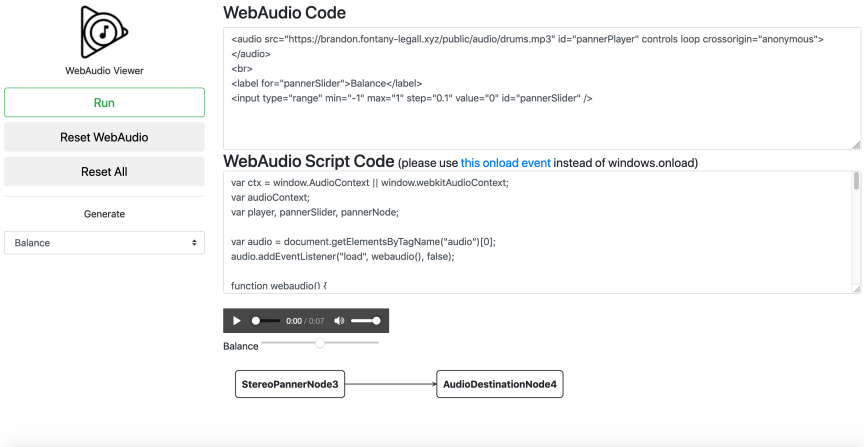

FIGURE 3.4 – Preview Viewer

Il comporte notamment le code permettant de traquer les appels WebAudio et en résoudre un graph. Il comporte donc un petit CallWatcher minimaliste personnalisé qui réalise les décorateurs et redéfinit les appels graces aux méthodes prototypées. Les prototypes sont des constructeurs de fonction. Par défaut, chaque fonction a un prototype vide et nous pouvons ajouter des propriétés et méthodes.

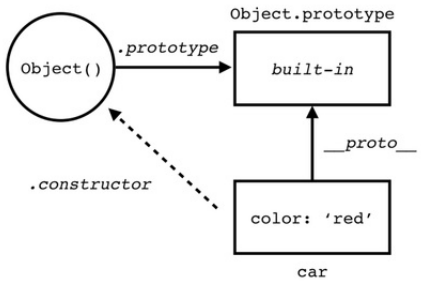

FIGURE 3.5 – Javascript prototypes[15]

De plus, il contient une classe Graph permettant de créer le graph et de le mettre à jour à chaque appel webaudio attrapé.
Pour conclure, nous avons réussi à créer un viewer fonctionnel pour les WebAudio simple pour le moment static uniquement, c'est-à-dire, qu'il ne va pas se mettre à jour automatiquement lors de modifications en temps réel ayant encore des soucis du côté du garbage collector. Nous avons les mêmes résultats que le debugger de Google ce qui est extrêmement encourageant pour la suite du projet.

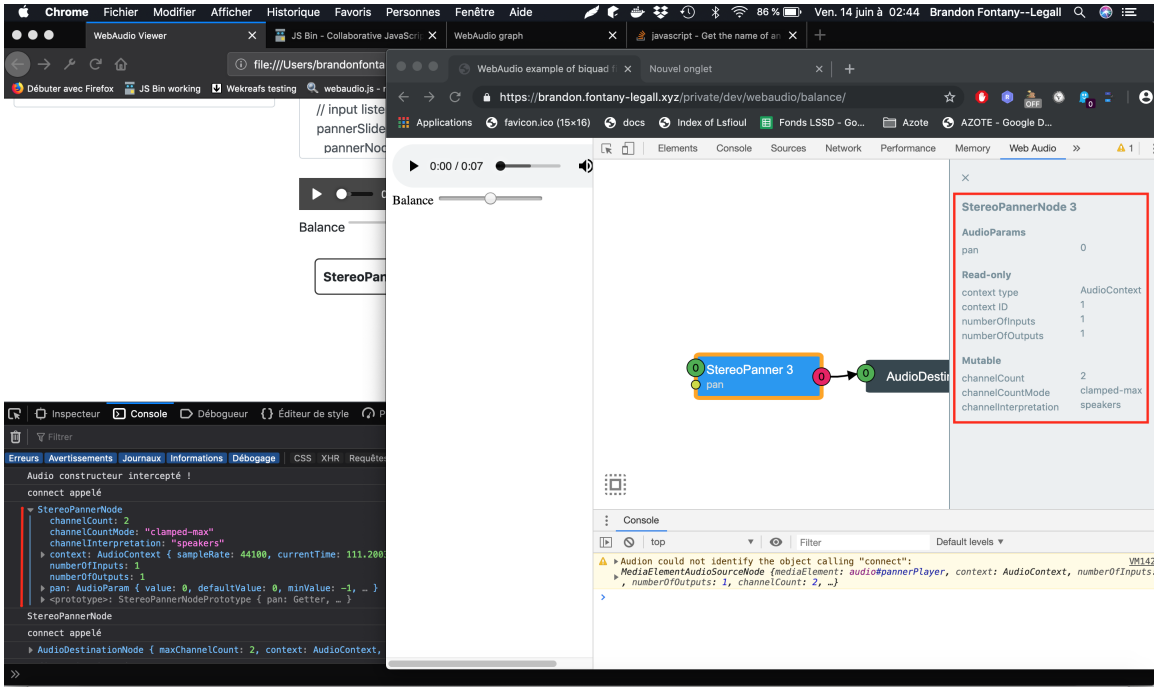

FIGURE 3.6 – Comparaison entre notre Viewer et les résultats de Google
Nous avons les même résultats !

Chapitre 4

Gestion de projet

4.1 Communication

La communication entre les différents acteurs du projet s'est principalement déroulée via e-mail. En effet, le personnel de Mozilla n'étant pas sur Nice, il a été nécessaire de faire passer nos communications principalement par mail avec les contraintes qui s'y impose comme les différents décalages horaire et langue.

De plus, un meeting en visioconférences a été réalisé avec notre Mentor, Yulia Startsev, de chez Mozilla ainsi qu'un rendez-vous effectué avec Paul Adenot, aussi employé chez Mozilla.

Nous avions aussi à notre disposition deux serveurs Slacks dont un du nom de WasabiWebAudio qui est l'équipe de recherche WebAudio de Michel Buffa ainsi que le serveur WebAudio avec divers membres du milieu du WebAudio dont certains employés de Google, Mozilla, etc. dans lequel il y a des échanges d'information.
Pour finir, de multiples rendez-vous ont été faits avec le référent pour discuté de l'avancement et de possibles pistes à prendre.

4.2 Gestion des source

Pour la gestion des sources nous avons décider d'utiliser Git pour l'enregistrement et la gestion de l'information. De plus, la possibilité de versionning est aussi grandement appréciée. Nous avons notamment utiliser GitHub[8] comme plateform pour Git.
Github nous permet aussi de mettre en place des issues et kanban pour avoir une idée précise sur l'état actuel du projet en les taches restantes à faire.

4.3 Gestion des Gists

Pour la gestion des gists, nous avons utilisé la plateforme JSBin[9] pour les différentes confections. Une fois la confection fonctionnelle, ils sont publiés sur le git dans le dossier gists.

4.4 Méthode de programmation

Nous n'avons pas utilisé la méthode agile pour ce projet. En effet, nous avons décidé de faire en premier lieu le viewer au vu de la difficulté de la conception rendant plus simple la conception. Cette décision implique donc que les commits ne touchent pas l'intégralité du projet mais se concentre, dans un premier temps, une partie spécifique du projet. Une fois la conception effectué, nous nous recentrerons sur la méthode agile.

Chapitre 5

Conclusion

Pour conclure, nous avons rencontré énormément de soucis lors du projet compliquant énormément un projet qui de base était assez ambitieux. Les librairies que nous avions placées comme fournits se sont retrouvé non fiables devant faire le projet from scratch.
Au niveau du travail effectué, nous avons donc fait une première version de l'extension avec le devtools avec un graphique de base et un Css minimaliste. Nous avons aussi fait le viewer qui implémente une version minimaliste du CallWatcher. Cette version du viewer est statique pour le moment, c'est-à-dire qu'elle prend pas en compte les modifications en temps réel au vu des soucis de weakrefs.

Chapitre 6

Perspectives et réflexions personnelles

Dans l'avenir, nous allons continuer le sujet qui a comme timeline en **mis août**.
Le sujet est extrêmement intéressan et avons appris énormément de chose en Javascript ainsi qu'en web.

Bibliographie

[1] Audio data api. https://wiki.mozilla.org/Audio_Data_API.

[2] Audioworklet. https://webaudio.github.io/web-audio-api/#AudioWorklet-concepts.

[3] Content script. https://developer.mozilla.org/fr/docs/Mozilla/Add-ons/WebExtensions/Content_scripts.

[4] Content script webextension api. https://developer.mozilla.org/fr/docs/Mozilla/Add-ons/WebExtensions/Content_scripts#API%20WebExtensions.

[5] Devtools. https://developer.mozilla.org/fr/docs/Mozilla/Add-ons/WebExtensions/extension_des_outils_de_developpement.

[6] First draft specification of webadio api. https://www.w3.org/TR/2011/WD-webaudio-20111215/.

[7] Garbage collector. https://en.wikipedia.org/wiki/Garbage_collection_(computer_science).

[8] Github. https://github.com/.

[9] Jsbin. https://jsbin.com/.

[10] Slack. https://slack.com/.

[11] Weakrefs proposal. https://github.com/tc39/proposal-weakrefs.

[12] Webaudio proposal. https://lists.w3.org/Archives/Public/public-xg-audio/2010Jun/0018.html.

[13] David Gimelle. Java reference class. https://www.linkedin.com/in/davidgimelle/.

[14] Mozilla. Webextension devtools. https://developer.mozilla.org/fr/docs/Mozilla/Add-ons/WebExtensions/.

[15] Sandeep Ranjan. Prototype in javascript. https://www.codementor.io/sandeepranjan2007/prototype-in-javascipt-knbve0lqo.