Web Audio API & d3.js

Gunharth Randolf
FH Kufstein
Maximilianstraße 31
6020 Innsbruck
+43 681 2050 6446
gunharth@gmail.com

Helene Wechselberger
FH Kufstein
Andreas-Hofer-Straße 17a
6020 Innsbruck
+43 650 560 22 46
wechselbergerhelene@gmail.com

ABSTRACT

Web Audio API is a rather recent W3C API, which was first actively used in beta stage in 2011 and 2012. Since then, the development of Web Audio API has been constantly ongoing, nevertheless the applications using Web Audio API are still outnumbered by commercial and open source tools available on native platforms. Even 7 years after its introduction to the web development community, W3C still lists Web Audio API as Editor's Draft. This paper will therefore look into the matter of in-browser-audio support via HTML5 <audio> tag, describe the current state of the Web Audio API and feature our audio processing experiments with various sources in combination with tone visualization using d3.js svg libraries. [1]–[3]

Categories and Subject Descriptors
[Multimedia Information Systems]: Audio, Video and Hypertext Interactive Systems
[Programming Languages]: Hypertext languages,
[Human-centered computing]: Graph drawings

General Terms
Experimentation, Languages

Keywords
HTML5, JavaScript, Web Audio API, D3.js, SVG

1. INTRODUCTION

Web Audio API handles audio operations within a created audio context. Due to wide-spread browser support no additional installations or plug-ins are necessary to play and control audio in-browser. Web Audio API follows the principle of the audio routing path, which consist of various audio nodes. These nodes are linked into chains, starting by the source node, optionally passing through a processing node and finally arriving at the destination node. This API is built on top of HTML5 <audio> tag and requires the DOM to feature an audio-tag or create one using JavaScript. The audio curve can be represented using built-in features or by transferring the audio data into SVG graphs, provided by D3.js library. [4]

2. TECHNICAL BACKGROUND

The following section of this paper will mention all relevant elements of Web Audio API, describe the main features and show examples taken from our conducted experiment.

2.1 HTML5

HTML5 improved HTML4 markup language by adding new attributes, partly with semantic tag meaning, like <nav>, <section>, <header> - but HTML5 also introduced the media tags like <video> and <audio>. [5]

HTML5 aims at implementing better semantics and dynamics into webpages and thus enhancing web experience by conveying a new level of interactivity for users. [6]

2.2 <audio> tag

This HTML5 element provides native audio playback in common browsers. However, audio control functions are very limited: No real-time effects can be applied, sounds can neither be analyzed nor pre-buffered. [7, S. 1]

2.3 Web Browser API

APIs – short for Application Programming Interfaces – are pre-made constructs that allow to access complex functionality rather easily. Web Audio API counts among Browser APIs, meaning the API is directly implemented in most common browsers and thus making plug-ins and desktop applications redundant. [8]

2.4 Web Audio API

Web Audio API is accessed using JavaScript, allowing developers to easily manipulate audio in the browser, e.g. altering the volume on an audio track, applying effects to it, etc. In the background, this API is based on lower-level code like C++, but the API reduces complexity by providing pre-set features, which can be simply addressed via the use of JavaScript language. [8], [9]

2.5 Audio Routing Graph and Audio Context

With Web Audio API an Audio Routing Graph is set up within an Audio Context. A simple Audio Routing Graph consists of a source node and a destination node, such being the user's speakers for example. In order to apply effects and transformations to the audio from the source node, a processing node is intermediated. For visualizing the audio file in graphs, an analyzer node is required. In the following section we will look at the basic processes within a Web Audio API and describe 3 types of source nodes and give examples for transformations in the process node.

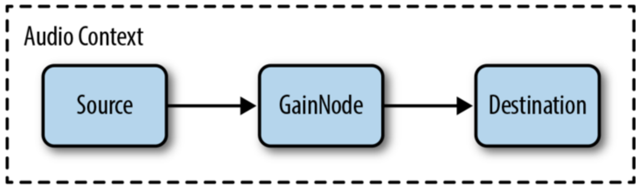

Figure 1. Simple Audio Routing Graph GainNode representing an example for a transformation node

2.6 Handling the Web Audio API

For applying the Web Audio API, an <audio> tag with URL in the HTML file is essential – either it is provided or gets created using JavaScript. This audio URL is loaded into the variable Audio Source.

```
let audioURL; // audio url of the html5 element
let audioSource; // source node data of the
audio context
```

In case no audio element is implemented in the HTML file yet, a new audio element is created using the following code:

```
let audio = document.createElement('audio');
audio.src = audioURL;
$('#' + id).append(audio);
```

Either way, a new audio context is generated and the audio source loaded into this context in order to process it via Web Audio API features. [10]

```
const AudioContext = window.AudioContext ||
window.webkitAudioContext;
const audioCtx = new AudioContext(); // the
defined audio context
```

2.7 Experimenting with source nodes

In this experiment we tried four different ways to receive an audio for the source node.

First, an existing HTML5 audio element is used:

```
<audio controls src="piano.wav">
```

Second, an audio file is uploaded through the browser using JavaScript file reader:

```
<input type="file" accept="audio/*"
onchange="loadAudioFile(this.files[0]);">
```

And third, the file is loaded with Ajax, using async await with fetch:

```
async function loadAudioWithAjax() {
// JS Promise
let arrayBuffer = await (await
fetch('guni.ogg')).arrayBuffer();

audioURL = URL.createObjectURL(new
Blob([arrayBuffer]));

audioSource = await
audioCtx.decodeAudioData(arrayBuffer);
}
```

The fourth possibility to gain a source node we experimented with, was by using the MediaStream Recording API – a separate API, generating audio data from the user's microphone.

```
async function recordFromMicrophone() {
   let chunks = [];
   let stream = await
navigator.mediaDevices.getUserMedia({ audio:
true, video: false });
   let mediaRecorder = new
MediaRecorder(stream, { mimeType: "video/webm"
});
   mediaRecorder.start();
   mediaRecorder.ondataavailable = function (e)
{ chunks.push(e.data); };
   mediaRecorder.onstop = async function () {
       let blob = new Blob(chunks, { type:
mediaRecorder.mimeType });
       audioURL = URL.createObjectURL(blob); //
audioSource from audioURL
   }
}
```

When working with source nodes 3 and 4 it is necessary to load the audio element to the arrayBuffer, followed by transforming it to blob data format, which allows us to generate an URL which can then be addressed in the HTML audio tag. [11]–[13]

2.8 Transformation Node

The transformations featured in our example were all provided by the API. More specifically we experimented with Convolver, WaveShaper, BiquadFilter, Gain, DynamicsCopressor, Oscillators and extracts from the Google Jungle library. [14]

2.9 Analyzing Node

This node provides real-time frequency and analyses information of the used audio file. The analyzing node puts the audio stream through from the input (or processing) node to the output node in an unchanged way, but generates data in the meanwhile, processes it and creates audio visualizations from the received data. This is enabled by iterating over the generated arrayBuffer within the Audio Context. [12]

2.10 Visualisation

Standard procedure comprises to pass the audio frequency data to a canvas 2D and paint the graphs in there. Out of experimental interest, we chose to utilise d3.js SVG rendering for visualisation of our test audio files. In real-life production systems however, it is recommended to implement canvas 2D technologies for performance advantages.

2.11 D3.js Library

Based on JavaScript technology, D3.js enables document manipulation by adding interactivity to data by using HTML, SVG and CSS. In our paper we implemented SVG 'frequency group' – a bar chart and in addition to that a line chart, the so called 'wave form group'. [15]

3. CONCLUSIONS

Possibly the most interesting finding from our experiment was to see, that even nowadays beta phases of fairly new technologies and features like the Web Audio API can hold on for longer than 7-8 years. When working with the Web Audio API and the W3C Editor's Draft Documentation it became obvious that the development process is still fully ongoing and documentation sometimes is rather unspecific. However, the improvement of user experience online is enormous by avoiding the installation of external plug-ins. Flexible variations of different audio nodes as well the possibility to combine the Web Audio API with web recording API and SVG visual elements open up uncountable opportunities to enhance interactivity online.

4. REFERENCES

[1] „Web Audio API“, 2018. [Online]. Verfügbar unter: https://www.w3.org/TR/webaudio/. [Zugegriffen: 21-Jän-2019].

[2] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, und S. Letz GRAME, „Towards an open Web Audio plugin standard“, in *WWW ’18 Companion Proceedings of the The Web Conference 2018*, 2018, S. 759–766.

[3] D. Humphrey, C. Brook, und A. MacDonald, „Exposing audio data to the web: an API and prototype“, in *Proceedings of the 19th international conference on World wide web*, 2010, S. 1365–1368.

[4] „Basic concepts behind Web Audio API | MDN“, 2018. [Online]. Verfügbar unter: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Basic_concepts_behind_Web_Audio_API. [Zugegriffen: 21-Jän-2019].

[5] M. Rouse, „What is HTML5?“, 2014. [Online]. Verfügbar unter: https://searchmicroservices.techtarget.com/definition/HTML5. [Zugegriffen: 22-Jän-2019].

[6] P. Shah, „What all you need to know about HTML5“, 2017. [Online]. Verfügbar unter: https://opensourceforu.com/2017/06/introduction-to-html5/. [Zugegriffen: 22-Jän-2019].

[7] B. Smus, *Web Audio API*. Sebastopol, CA 95472: O’Reilly Media, Inc., 2013.

[8] „Introduction to web APIs“. [Online]. Verfügbar unter: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction. [Zugegriffen: 22-Jän-2019].

[9] „HTML5 Audio“. [Online]. Verfügbar unter: https://www.w3schools.com/HTML/html5_audio.asp. [Zugegriffen: 22-Jän-2019].

[10] B. Smus, „Getting Started with Web Audio API - HTML5 Rocks“, *2011*. [Online]. Verfügbar unter: https://www.html5rocks.com/en/tutorials/webaudio/intro/. [Zugegriffen: 21-Jän-2019].

[11] „FileReader | MDN“. [Online]. Verfügbar unter: https://developer.mozilla.org/en-US/docs/Web/API/FileReader. [Zugegriffen: 22-Jän-2019].

[12] „AnalyserNode | MDN“. [Online]. Verfügbar unter: https://developer.mozilla.org/en-US/docs/Web/API/AnalyserNode. [Zugegriffen: 22-Jän-2019].

[13] „Recording audio in HTML5 Archives“. [Online]. Verfügbar unter: https://addpipe.com/blog/category/recording-audio-in-html5/. [Zugegriffen: 22-Jän-2019].

[14] „Web Audio API“, 2018. [Online]. Verfügbar unter: https://webaudio.github.io/web-audio-api/. [Zugegriffen: 21-Jän-2019].

[15] „D3.js - Data-Driven Documents“. [Online]. Verfügbar unter: https://d3js.org/. [Zugegriffen: 22-Jän-2019].

Figure 1.:
B. Smus, *Web Audio API*. Sebastopol, CA 95472: O’Reilly Media, Inc., 2013. Page 6.