Putting Web Audio API to the test: Introducing WebAudioXML as a pedagogical platform

Hans Lindetorp
Royal College of Music
Stockholm, Sweden
hans.lindetorp@kmh.se

Kjetil Falkenberg
KTH Royal Institute of Technology
Stockholm, Sweden
kjetil@kth.se

ABSTRACT

Web technologies in general and Web Audio API in particular have a great potential as a learning platform for developing interactive sound and music applications. Earlier studies at the Royal College of Music in Stockholm have led to a wide range of student projects but have also indicated that there is a high threshold for novice programmers to understand and use Web Audio API. We developed the WebAudioXML coding environment to solve this problem, and added a statistics module to analyze student works. Three groups of students with technical respectively artistic background participated through online courses by building interactive, sound-based applications. We analysed the projects to understand the impact WebAudioXML has on creativity and the learning process. The results indicate that WebAudioXML can be a useful platform for teaching and learning how to build online audio applications. The platform makes mapping between user interactions and audio parameters accessible for novice programmer and supports artists in successfully realizing their design ideas. We show that templates can be a great help for the students to get started but also a limitation for them to expand ideas beyond the presented scope.

1. INTRODUCTION

Web technologies in general and Web Audio API [1] in particular have a great potential as a learning platform for developing interactive sound and music applications. The combination of open source technologies, standard formats, no need for compilation nor installation and the wide support on a range of platforms makes Web Audio API one of the most attractive technologies for teaching purposes. There is also a growing support for sensory data APIs like DeviceOrientation, Accelerometer, and ForcedTouch that can turn an ordinary smartphone into a powerful, interactive musical instrument.

Through experience of teaching interactive music production we have identified a population of music students who have a deep understanding of music and audio production but little or no programming experience. For this group, the threshold is high for getting started to program the audio signal routing and mapping variables to audio parameters, and that often leads to less interest and to students lowering artistic ambitions in their interactive music projects [15].

In contrast to the above-mentioned music students, we also teach technology students who have a good knowledge of programming, but generally less experience of creative work. This group requires that the platform allows for easy access to artistic realization and experimentation similar to other established frameworks like Pure data and Supercollider, but that better facilitates web applications.

Aiming at meeting those challenges and needs, we developed WebAudioXML [16], an open source framework [1] that offers an accessible XML syntax for configuring audio objects in a similar way as HTML represents visual content in web page. With this declarative approach, Web Audio API becomes more accessible for creators where HTML is the only coding syntax they know. WebAudioXML has since its release in early 2020 been used in six different courses and more than forty student projects at The Royal College of Music (KMH),[2] KTH Royal Institute of Technology, [3] and Södertörn University [4] in Stockholm, Sweden. It is projected to be the main platform for the courses where we work with web-based audio, and also for student projects where data collection in e.g. listening and sound design experiments can be web-based. In addition to the development of WebAudioXML features for programming music interaction, the platform includes a statistical tool for analyzing code and for supporting research efforts.

The aim with this study is to test and evaluate WebAudioXML as a platform for learning how to make interactive web audio applications. Particularly, we identify drivers and barriers for adopting the technology as expressed by the students in order to focus future development efforts to overcome obstacles.

2. BACKGROUND

The current study is a part of the ongoing evaluation of methods for teaching sound and music computing at KMH and KTH. It contributes in a similar way as earlier studies to integrate research and education at our campuses [11]. The following section introduces previous work at KMH and KTH, and also outlines the WebAudioXML development in preparation for this study.

[1] https://github.com/hanslindetorp/WebAudioXML
[2] http://www.kmh.se
[3] http://www.kth.se
[4] http://www.sh.se

Web Audio Conference WAC-2021, Spring, 2021, Barcelona, Spain.

2.1 Earlier experiences

The students at the Music Production bachelor and masters program at KMH often have a background in pop and rock music. In order to expand their perspectives on music production and to face the challenge of producing music for an interactive environment, all of them get to try producing music for computer games, web pages and interactive installations. Yearly since 2013, different student groups have produced multi-channel, interactive music installations for several venues including Kulturhuset, [5] Stockholm City Museum, [6] Nobel Prize Museum, [7] and the Museum of Performing Arts. [8]

These KMH projects have been set up as an iterative process where music production, course syllabus, technical development and research is highly integrated. At KTH, communication, interaction design, and prototyping has been given more focus than music production. In the following sections, we briefly describe key elements of these projects.

2.1.1 Nobel Creations

From 2014–16, fourteen master students from KMH participated in Nobel Creations [10] at the Nobel Prize Museum. Each student produced an interactive composition controlled by sensors and touch screens. The exhibition was running for up to three months with the music responding to the visitors' movements and actions. Web Audio API was controlling the playback of thousands of audio loops in a 16 channel speaker that was set up through a JavaScript framework called iMusic.[9]

2.1.2 Sound Forest

The "Sound Forest" is a permanent, large-scale digital musical instrument installation at the Swedish Museum of Performing Arts that can host a range of different musical material, depending on context or concurrent exhibitions [2, 18]. It consists of five interactive light-emitting strings attached between the ceiling and the floor in a dedicated room covered by mirrors. Visitors can interact with the installation by plucking the strings, resulting in sonic feedback played from a ceiling-mounted loudspeaker located directly above each string, visual feedback as multi-colored light emitted from within each string, and haptic feedback transferred through vibration platforms at the root of each string.

Several projects by master students from KMH have been studied to better understand how a composition is presented to the visitor through the Sound Forest [7] and also how music producers could learn how to compose for a haptic experience [6]. The Sound Forest can host a variety of technical frameworks including Web Audio API. In these studies, iMusic and Web Audio API were the core part of controlling the audio playback, and a predecessor to WebAudioXML was introduced for the students to setup the audio configuration for mixing purposes.

2.1.3 Klangkupolen

"Klangkupolen" is a multi-channel, super-surround speaker setup at KMH [8]. In a study from 2019, bachelor students produced interactive music where the audience controlled the playback using two or more smartphones [15]. The technology used nodejs,[10] express.js, [11] and socket.io [12] to connect the devices to a server that hosted the audio application running iMusic and Web Audio API. This study was the first at KMH that evaluated web technologies and smartphones for building interactive musical instruments. The result was very promising even if many students also pointed out barriers in the technology limited their creativity. The study led to further development of the technology that, in turn, initiated the development of WebAudioXML.

2.1.4 KTH projects

Since early 2000, students of sound and music computing at KTH have developed new musical instruments, novel interfaces, and studied music communication in projects. Most of these have been realized with Pure data, Supercollider, Bela and similar platforms, see for instance [5]. However, many student projects would have made bigger impact if they could have been realized as web applications, such as evaluations of listener's experiences. And, although it is possible to distribute for instance audio-based games through established application stores, a web-based approach would have given students more flexibility and projects more longevity; see for example [12].

2.2 WebAudioXML

WebAudioXML consists of an XML syntax specification and a parser. The framework has been developed together with and tested by audio experts and lecturers from music production and sound and music computing communities. The main feature of WebAudioXML is to let the developer configure an audio graph and map external variables to audio parameters. It also adds objects for routing audio signals, buffering audio files, specifying envelopes, and connecting Web MIDI API to trigger and control audio parameters. The audio configuration can be built using XML syntax only. For more experienced users, there is also a JavaScript API that makes all audio nodes accessible through the standard Web Audio API syntax.

A quick comparison with related frameworks tells that WebAudioXML is an XML standard like MusicXML [9] but is used to describe an audio configuration instead of a musical structure. It simplifies the development of audio applications like tone.js [17] by adding high level functions and objects but does not contain any built-in effects, but rather encourages the community to build and share components in the XML format. Compared to frameworks like Web Audio Modules (WAM) [14] and Web Audio Plug-in (WAP) [3], WebAudioXML is rather different as it operates on a higher level as the audio host rather than a plug-ins and there is an opportunity to use WebAudioXML together with such frameworks by using the AudioWorklet node [4].

As a preparation for this study, WebAudioXML was further developed from its first version [16] to meet the needs of the targeted courses. The new features focused mainly on user interaction, but there were also added mechanics of reporting statistical data about the configuration for research purposes. In the following, these developments are described briefly.

[5] http://kulturhusetstadsteatern.se
[6] http://stadsmuseet.stockholm.se
[7] http://nobelprizemuseum.se
[8] http://scenkonstmuseet.se
[9] http://github.com/hanslindetorp/imusic
[10] https://nodejs.org/en/
[11] https://expressjs.com
[12] https://socket.io

2.2.1 *Multiple touch-events and devices*

Some of the courses in this study requested features for dealing with multiple touch-events letting different fingers control different audio parameters. The previous version of WebAudioXML had variables for mapping touch interaction with a single finger (relX and relY) but the syntax was now extended to support multiple fingers from multiple clients. The touch-events from the connected clients can be referred to using the following syntax:

```
follow="client[clientID].touch[touchID].varName"
```

The following example demonstrates how one client can map the first finger to control the pitch of an OscillatorNode and the second finger to control the cutoff frequency of a BiquadFilterNode. Another client is controlling the gate of a GainNode and the result is a musical instrument that requires co–operation to work. The '#touchArea' refers to an HTML element for capturing the touch-events in the client interface.

```
<?xml version="1.0" encoding="UTF−8"?>
<Audio version="1.0" interactionArea="#touchArea">
  <Chain>

    <OscillatorNode type="sawtooth">
      <frequency follow="client[0].touch[0].relX">
      </frequency>
    </OscillatorNode>

    <BiquadFilterNode>
      <frequency follow="client[0].touch[1].relY">
      </frequency>
    </BiquadFilterNode>

    <GainNode>
      <gain follow="client[1].touch[0].down"></gain>
    </GainNode>

  </Chain>
</Audio>
```

2.2.2 *Mapping delay*

An interesting feature request from the music production master students was a way of delaying the mapping from user interactions to an audio parameter. With this feature, the students built a "gesture echo" where the echo was of the gesture, and not of the sound. This makes it possible to map the interaction with delay to one audio parameter and without delay to another. The syntax enables any mapping to be delayed differently if specified. The following example will make a filter sweep 1 second after the pitch sweep on the oscillator:

```
// e.g.
// relX controlling pitch in real time
<OscillatorNode type="sawtooth">
  <frequency follow="relX"></frequency>
</OscillatorNode>

// relX controlling filter with 1000ms delay
<BiquadFilterNode>
  <frequency follow="relX" delay="1000">
  </frequency>
</BiquadFilterNode>
```

2.2.3 *Event list*

One of the projects focused on comparison between two different gestures. To support this need, we developed a simple event list and sequencer with the possibility to record, playback and compare different gestures. The students used it for recording touch-swipe-gestures only, but the feature is generic and supports any external variables through a small set of JavaScript methods:

```
webAudioXML.addSequence(events, [name])
webAudioXML.getSequence([name])
webAudioXML.play([name])
webAudioXML.copyLastGesture()
```

'events' is an array with time-stamped objects, and 'name' is a string.

2.3 Integrated statistics tool

To easily compare the configurations between the different projects we added a novel statistics tool in WebAudioXML. The feature is used for collecting, sorting and counting all included audio objects in a configuration and also summarizes the number of mappings between external variables and audio parameters. The output can be exported in a spreadsheet format for further calculations.

Table 1: An example of the statistical output

Audio Node	Nr of instances
AudioBufferSourceNode	8
BiquadFilterNode	12
GainNode	12
Mixer	1
Chain	4
Mapped interaction variables	Nr of instances
client[0].touch[0].relX	2
client[1.touch[0].relY	3

3. METHOD

This study is done through collecting data from three ongoing courses at KMH and KTH that have been using various different technologies in previous years. We ran the study in parallel with the courses and acted as both lecturers and researchers, with the students giving their consent to participate and be included. The data were collected through analysis of the configuration files and through online forms. The students also contributed to the study by giving access to their individual project reports and source code.

3.1 Participants

In total, 22 students from three different educational programs participated in the study. Twelve were KMH students of music production at a bachelor level and seven at a masters level. The third group followed a masters level course in sound and music computing at KTH. The students from KMH generally have no prior programming experience but most of them know some basic HTML, while the students from KTH generally have good programming experience but generally less formal music training than KMH students, if any.

3.2 Student Projects

The courses for the three groups were similar but not identical. They all started with 2–3 introductory lectures where the technical and aesthetic foundation was laid out. The lectures were video recorded for the students to revisit. The students were all working from home due to the Covid-19 pandemic restrictions and produced the result in approximately one week and with a few hours of supervision. Below is a brief explanation about the preconditions for the different courses.

3.2.1 Bachelor and master student projects at KMH

The students at KMH created interactive music instruments with a background musical layer and two or more interactive smartphone–instruments with sounds matching the background. The template files contained all necessary HTML, XML, CSS and JavaScript files and an https-server built with nodejs/express.js to set up the communication between the smartphones (clients) and the server (master) using socket.io. The client web page had a configurable touch area that captured touch-events and acted as a reference for all X and Y values for the touch-events. The master web page was configured to buffer, organize and synchronize the playback of audio files using iMusic and to mix, process and generate audio based on the interactions using WebAudioXML. The template file for WebAudioXML contained examples of mixing, mapping and audio configurations.

3.2.2 Master student projects at KTH

The project challenge was to create a *Simon Says*-type game for smartphones with sound as the gameplay, aimed at helping hearing impaired to train their listening abilities (inspired by previous studies, see e.g., [12]). From the project instruction, the user was to hear a sound originating from a recorded gesture or movement and then try to recreate this sound with a new gesture on a smartphone. The sound is a sonification of the gesture, and this programmed sonification is also the project outcome. The template contained all needed HTML, XML, CSS and JavaScript files for configuring the interaction and audio but required the participants to collectively write the necessary gaming engine such as pattern recognition and matching.

3.3 Data collection

After the students had presented their projects, we collected data to understand how they used the technology to achieve the intended result and what drivers and barriers they experienced in the process. The data made available for analysis thus included comments from oral presentations in addition to the source code of the projects, individual reports and reflections, and questionnaire data from an online form.

3.3.1 Source code analysis

We analyzed the audio configuration file from all projects comparing the number of audio nodes, parameters, and mappings between user interactions and the audio parameters. For this purpose we used the statistics tool that was added to WebAudioXML.

We also compared the code using an online text comparison tool [13] to understand to what degree the work was a result of adding, deleting or changing the template configuration. We compared each WebAudioXML configuration file with the corresponding template file and collected the number of additions, deletions and changes within all changed lines.

[13]https://www.diffnow.com/report

3.3.2 Evaluation form

For each of the three courses, the students were asked to answer a few questions about their experience of working with WebAudioXML through an online form after the end of the course. We asked about the time balance between artistic/creative vs. technical work/coding, their artistic aim, possible barriers that hindered them, what other program(s) they would rather use to solve the challenge and why they would rather use them.

3.3.3 Reports and presentations

The courses had different types of formal assessment, but common to them was that students did oral presentations. In these sessions, there was room for discussion. In two courses the projects were assessed through reports, and in the third, students handed in an assignment including a personal reflection about the task. Collectively, these sources provided us with free-form comments that were analyzed to identify possible barriers and drivers of using WebAudioXML in particular (inspired by the method described in [13]).

4. RESULT

All participating students succeeded with their projects and the technology worked for all students on their different devices and operating systems. The only exception was a few smartphones that did not support DeviceOrientation and ForcedTouch. Below is a summary of the most important findings from the source code analysis, the evaluation form and the reports and presentations.

4.1 Size and complexity of the projects

The analysis of the audio configuration source code is presented in two sections. The first describes the size of the project in regard to the number of audio objects and mapped parameters. The second presents various measurements of the source code which reflects how the students edited the template files. The two projects at KMH used a similar template containing a default configuration with 37/36 audio objects and 79/80 lines of code, respectively. The template for KTH was smaller, with 12 audio objects and 21 lines of code.

4.1.1 Audio Objects

The number of audio objects were between 10 and 88, with an average of 40 audio objects. The master students at KMH implemented the most objects per project (48) and the bachelors the least (33). A comparison with the template file for each group shows that the bachelors reduced their configurations with four objects on average, the masters at KMH increased theirs with eleven and the masters from KTH increased theirs with 32 objects.

Grouping all students according to the complexity of the audio configuration indicates that seven participants created small projects with 20 or fewer audio objects, six participants created a medium size configuration with 21-40 audio objects. Finally, eight students created large projects with

more than 40 audio objects. Participants from all classes were represented in those categories. The number of mappings of user interactions to audio parameters span from three in the simplest project to 26 in the most complex. The average number of mappings were similar in all groups spanning from 7 to 12. One difference between KMH and KTH is that the KMH projects specified the use of multiple devices while the project instruction given to KTH students limited them to one device and only two (X/Y) variables.

4.1.2 Source code

The analysis from comparing the source code using the online text comparison tool shows that the size differs between 26 lines of code in the smallest project to 186 in the most complex. The files from. the master students at KMH were largest (101 lines on average) and the bachelors were smallest (74 lines on average).

A comparison with the template file for each group shows that the KMH bachelors reduced their code with five lines, and the masters instead increased their code with 21 lines. Finally, the masters at KTH increased their code by 57 lines in average.

Another measure that indicates how the source code was changed from the template is to count the number of added and deleted lines. This test shows that there was a bias towards deleting lines among the bachelor students (1 added and 17 deleted lines) while the master students at KTH did not delete any lines but added 32.

A last comparison shows the number of changes within changed lines (in-line additions, deletions and changes). We found that the master students at KTH had in average 23 lines with changes, the KMH bachelors had 54, and the KMH masters had 91 changes.

4.2 Drivers and barriers

From the collected material, we identified around one third more comments to signify barriers than drivers. A typical example of a barrier from the material was "*I had a bit of a hard time understanding how everything was working*", while a typical driver was "*when I learned all the parameters in WebAudioXML the process went on being much easier than I thought it would be*". Many comments were however less definite, such as "*I think it requires a whole lot of thinking in both composition and programming*", which could be interpreted as being both a barrier (too much to take on) and a driver (inspiration to combine competences).

In a first approximation, drivers and barriers were tagged and sorted into four main, arbitrary categories: ambition, attitude, competence, and planning. Each category will necessarily have slightly different meaning for barriers and drivers, respectively. Ambition typically involved underachievement for both barriers and drivers; for instance, "*[I did not create a custom waveform] because I thought that four different waveforms were enough*". Attitude typically involved having a preconception or experience that the platform is too limited (barrier) or advantageous (driver). Competence was expressed as either lack of or sufficient technical competence; there were no comments indicating lack of artistic skills. Planning in the material was related to tasks and time management.

For barriers towards using WebAudioXML, most comments were associated with (lack of technical) competence, (disproportionate) ambition, and to (bad) planning. Most drivers for using the platform were associated with (a positive) attitude, but also of (sufficient technical) competence. Driver comments were generally not related to issues concerned with planning.

5. DISCUSSION

Even if it is not feasible to make a one-to-one comparison with our earlier projects as they did not share objectives, technical framework, or students, these three projects stand out in various ways. Music producers at KMH with little or no programming experience managed to successfully develop fully functional and interesting multi-sensory-controlled applications for multiple devices. It is the first year the course was given remotely through video-conferencing software, and still all students managed to set up the configuration with a limited amount of supervision. With a similar challenge, the master students at KTH managed to build synthesized audio models that could be tested and used online. This is a great contribution in general and specifically for our future research plans that explore ways of helping hearing impaired with new tools; it has been a challenge in the past to distribute and run tests with this particular focus, and even of engaging test participants.

Some of the students could take the advantage to use the open structure in WebAudioXML to build instruments and tools in a very creative and playful style. It confirms a proof-of-concept of WebAudioXML and indicates that the XML-based syntax is the key for some students to transfer their musical creativity into web audio development. A few students, on the other hand, had to struggle more to make even the simplest configurations work. They would still get stuck in situations where small errors in the code (momentarily) break the whole application.

We tried different approaches regarding the complexity of the template files for the different student groups and the result indicates that the students from KMH, who got a more complex file, rather picked and chose from the contents, while the students from KTH with the simpler template created more code by themselves. From this study, it is not possible to say whether the template or the students' programming skills were the most important factors, but that would be an important question in future studies.

A part of the result that was not in focus before the course was the importance of the musical context in which the students from KMH built their interactive instruments. The musical qualities of the background tracks in their applications framed the design of the instruments in a beneficial way, bringing meaning and setting up rules for the interaction to unite with.

The drivers and barriers experienced by the students towards using the technology offered by WebAudioXML are only briefly analyzed. In subsequent studies, these issues need to be given more attention. However, the results presented show that if we can tackle the technical issues with appropriate supervision and preparation, and provide students with a realistic time management appraisal, we can lower the threshold for getting something to work even further.

In addition to the mentioned drivers and barriers, students commented on their learning outcome, as defined in their course descriptions. While this is not covered in the current study, it is apparent from the material that most students have developed from working with their projects,

and not only in solving technical difficulties: "*what I could learn during this time opened up for a new way of thinking and to 'get outside the box' [musically]*". If given more time, the comments suggest that students would spend more time in experimenting.

Another finding was that during the tough times of Covid-19 social restrictions, where students were not allowed to visit the school premises nor collaborate face-to-face, the possibility offered by the system to have projects running on web pages brought students closer together. Also, it was much easier for external persons (to the course) to take part, test, and experience the works.

One of the most important results for the future development of WebAudioXML brought forward by this study is the statistics tool. It is very beneficial that a software platform used in a study also has an integrated tool that can give detailed information about what objects the audio configuration contains and how they are connected. We see a potential for an interface where different audio-related activities can be monitored, stored and analysed including developing work, testing and user interactions.

6. IMPLICATIONS

The result indicates that WebAudioXML works well as a learning platform for web audio application development. It serves as a convenient door opener for novice programmers to create relatively complex and artistically interesting results within a limited time frame. We would gladly welcome further discussions and development by a community of researchers, educators and developers that potentially could lead to useful standards and tools for building Web Audio applications.

Acknowledgments

The authors want to thank all participating students at KMH and KTH for contributing with valuable data.

7. REFERENCES

[1] P. Adenot and R. Toy. *Web Audio API: W3C Candidate Recommendation, 18 September 2018*, 2018 (accessed February 18, 2020).

[2] R. Bresin, L. Elblaus, E. Frid, F. Favero, L. Annersten, D. Berner, and F. Morreale. Sound Forest/Ljudskogen: A Large-Scale String-Based Interactive Musical Instrument. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 79–84, 2016.

[3] M. Buffa, J. Lebrun, J. Kleimola, O. Larkin, and S. Letz. Towards an open Web Audio plugin standard. In *Companion Proceedings of the The Web Conference 2018*, pages 759–766, 2018.

[4] H. Choi. Audioworklet: the future of web audio. In *Proceedings of ICMC*, 2018.

[5] K. Falkenberg, H. Lindetorp, A. Latupeirissa, and E. Frid. Creating digital musical instruments with and for children: Including vocal sketching as a method for engaging in codesign. *Human Technology*, 16(3):348–371, 2020.

[6] E. Frid and H. Lindetorp. Haptic music - exploring whole-body vibrations and tactile sound for a multisensory music installation. In *Sound and Music Computing Conference*. Zenodo, 2020.

[7] E. Frid, H. Lindetorp, K. F. Hansen, L. Elblaus, and R. Bresin. Sound Forest: Evaluation of an Accessible Multisensory Music Installation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA, 2019. Association for Computing Machinery.

[8] H. Frisk and W. Brunson. Building for the Future - research and innovation in KMH's new facilities. In R. Bresin, editor, *SMC Sweden 2014: Sound and Music Computing: Bridging science, art, and industry*, pages 10–11, 2014.

[9] M. Good et al. MusicXML: An internet-friendly format for sheet music. In *Xml conference and expo*, pages 03–04, 2001.

[10] J.-O. Gullö, I. Höglund, J. Jonas, H. Lindetorp, A. Näslund, J. Persson, and P. Schyborger. Nobel creations : Producing infinite music for an exhibition. *Dansk Musikforskning Online*, pages 63–80, 2015.

[11] K. F. Hansen, R. Bresin, A. Holzapfel, S. Pauletto, T. Gulz, H. Lindetorp, O. Misgeld, and M. Sköld. Student involvement in sound and music research: Current practices at KTH and KMH. In *Proceedings of the Nordic Sound and Music Computing Conference*, pages 36–41, 2019.

[12] K. F. Hansen, R. Hiraga, Z. Li, and H. Wang. Music Puzzle: an audio-based computer game that inspires to train listening abilities. In D. Reidsma, H. Katayose, and A. Nijholt, editors, *Advances in Computer Entertainment*, volume 8253 of *Lecture Notes in Computer Science*, pages 540–543. Springer International Publishing, 2013.

[13] P. Josefsson, A. Baltatzis, O. Bälter, F. Enoksson, B. Hedin, and E. Riese. Drivers and barriers for promoting technology enhanced learning in higher education. In *INTED 2018 Proceedings*. IATED, mar 2018.

[14] J. Kleimola and O. Larkin. Web audio modules. In *Proc. 12th Sound and Music Computing Conference*, 2015.

[15] H. Lindetorp. Immersive and interactive music for everyone. In *Proceedings of the Nordic Sound and Music Computing Conference*, pages 16–20, 2019.

[16] H. Lindetorp and K. Falkenberg. WebAudioXML: Proposing a new standard for structuring web audio. In *Sound and Music Computing Conference*, pages 25–31. Zenodo, 2020. QC 20200722.

[17] Y. Mann. Interactive music with tone.js. In *Proceedings of the 1st annual Web Audio Conference*, 2015.

[18] J. Paloranta, A. Lundstrom, L. Elblaus, R. Bresin, and E. Frid. Interaction with a Large Sized Augmented String Instrument Intended for a Public Setting. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 388–395, 2016.