trackswitch.js: A Versatile Web-Based Audio Player for Presenting Scientific Results

Nils Werner, Stefan Balke, Fabian-Robert Stöter, Meinard Müller, Bernd Edler
International Audio Laboratories Erlangen, Germany
nils.werner@audiolabs-erlangen.de

ABSTRACT

trackswitch.js is a versatile web-based audio player that enables researchers to conveniently present examples and results from scientific audio processing applications. Based on a multitrack architecture, *trackswitch.js* allows a listener to seamlessly switch between multiple audio tracks, while synchronously indicating the playback position within images associated to the audio tracks. These images may correspond to feature representations such as spectrograms or to visualizations of annotations such as structural boundaries or musical note information. The provided switching and playback functionalities are simple yet useful tools for analyzing, navigating, understanding, and evaluating results obtained from audio processing algorithms. Furthermore, *trackswitch.js* is an easily extendible and manageable software tool, designed for non-expert developers and unexperienced users. Offering a small but useful selection of options and buttons, *trackswitch.js* requires only basic knowledge to implement a versatile range of components for web-based audio demonstrators and user interfaces. Besides introducing the underlying techniques and the main functionalities of *trackswitch.js* we provide several use cases that indicate the flexibility and usability of our software for different audio-related research areas.

1. INTRODUCTION

In science, the publication of scientific results is an important goal of research. However, most scientific publications are often only comprehensible to the expert audience active in the corresponding field of research. Additionally, being almost exclusively print-only, even if the result may have a very intuitive message, there is little possibility to present an interactive visualization or auralization.

Recently, the reproducibility of research results has become increasingly relevant [26]. For reproducible research, authors publish not only a manuscript, but also their source code, a dataset, or example files. In the most simple case those additional resources would be offered for download as one big compressed archive. Just like publications, these archives may be useful to other researchers familiar with the topic, but offer no immediate and intuitive insights in the data. Meanwhile, with the proliferation of high-speed internet access and the ever increasing possibilities of modern web browsers, consumers and researchers have grown accustomed to richer web applications and more interactive multimedia experiences. However, to offer such rich multimedia experiences, an increasing amount of technical knowledge is required from the authors. Researchers who may want to leverage these modern web features usually lack the training to work with the technologies required to create such presentations.

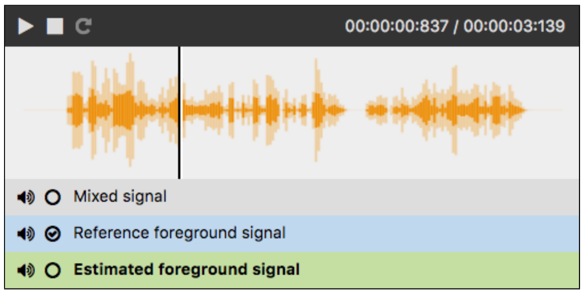


Figure 1: Example of a basic *trackswitch.js* instance with 3 tracks, a visualization of the waveform, and a seekhead indicating the current playback position.

A versatile, but simple to use and set up web audio player is needed to allow researchers to easily present their results and data interactively and in a meaningful way. To allow comparative presentations, the player should be able to play back and mix multiple audio tracks simultaneously. To allow the combination with a wide range of visualizations, a flexible visualization widget should be offered. To be suitable for quick and comprehensible presentations, a simple and intuitive user interface should be leveraged. Lastly, to be usable by developers with limited knowledge in web technologies, a simple programming interface is required.

In this work, we present a tool that fulfills these requirements and has been successfully used many times to present results from different fields in audio research: From comparing audio compression algorithms, to mixing and playing back source separation results, to the visualization and auralization of music annotation data.

The remainder of the article is structured as follows: In Section 2, we outline related implementations and compare their features to our requirements. In Section 3, we discuss the technical aspects of the presented implementation and give usage examples for developers. In Section 4, we present selected published examples from varying research fields before finishing with a conclusion in Section 5.

2. RELATED WORK

There exist many tools that supply audio playback functionalities in browsers. They can be roughly categorized in toolkits, self-hosted special-purpose applications, general video and audio players and lastly cloud-hosted commercial services.

Firstly, there are toolkits that allow the creation of custom-made applications. As an example for such a toolkit, *Peaks.js* [12] is a very robust and flexible software suite that incorporates waveform rendering, waveform file formats, playback control, and means to interact with waveforms. However, the software is missing multitrack playback and is very complex to set up for a simple presentation. A second toolkit is *waveform-playlist* [18]. It is a simpler, but richly featured tool to display waveforms and interact with multiple tracks at once, similar to a digital audio workstation (DAW). However, like many DAWs, the tool offers too many options and buttons to the user to be useful as a simple presentation tool.

The second class of tools comprises several ready-made applications designed for specific purposes. One such application is *MT5* [20], a multitrack player intended for musicians which offers multitrack playback, waveform and spectrum visualizations, volume control, and track mixing. Similar to *waveform-playlist*, *MT5* offers too many options and buttons to be useful in our scenario. Other examples for special-purpose applications are *webMUSHRA* [24] and *BeaqleJS* [22] which are designed for web-based subjective listening tests. They are specifically designed to not allow mixing multiple tracks and usually contain participant response forms to receive evaluation results.

Thirdly, there are general web multimedia players, e. g. *jPlayer* [8], *video.js* [17], and *plyr* [13]. These players are designed to play back one single audio or video file. They do not offer multitrack playback. Lastly, cloud-based services became more popular, e. g., *Soundtrap* [16], *Audiotool* [3] or *Amped* [1]. They usually offer too many options and require the user to create an account in order to work.

The only simple multitrack HTML audio player the authors know of is *webbasedHTMLplayer* [11]. Unfortunately, it has been unmaintained for years and uses the HTML5 `audio` tag which does not allow exact control over precise synchronous playback or file preloading behavior of multiple tracks at once. Because of all these individual shortcomings, none of the currently available players is suitable to present experimental results without being too hard to set up, too complex to use or too limiting in its technical architecture.

3. TECHNICAL OVERVIEW

trackswitch.js combines several core web technologies to function: HTML, JavaScript and cascaded stylesheets (CSS). For reading audio files, mixing the resulting streams and sending the audio output to the sound card, *trackswitch.js* uses the *Web Audio API* [19].

3.1 JavaScript Stack

In recent years, the JavaScript landscape has changed tremendously, with the rate of change increasing year by year. To accomplish recent development paradigms, many modern JavaScript frameworks, such as *React* [14] or *AngularJS* [2], ship with a large number of dependencies, require intricate compilation pipelines, and generally demand very in-depth expert knowledge from the developer.

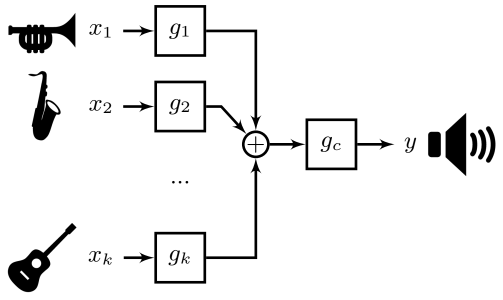


Figure 2: Audio Engine Overview. $x_1, \ldots, x_k$ denote inputs, $g_1, \ldots, g_k$ denote individual gain nodes, g_c denotes common gain node, y denotes output

On the other hand, as most researchers do not intend to develop powerful web applications but require a comprehensible and low-maintenance way to present audio results, most of the advantages of these modern JavaScript frameworks would not be useful to them. Secondly, many websites in education were not designed as a web application but often follow traditional, static web development paradigms. So instead, a solution without dependency to a specific framework and without a complicated development pipeline is required. This is why the only two dependencies for *trackswitch.js* are the very commonly used *jQuery* [9] and *Fontawesome* [5], both of which have no third party dependencies themselves. The two dependencies are either likely to be already in use or can simply be included from a content delivery network and can immediately be used.

3.2 Audio Engine

trackswitch.js makes use of the *Web Audio API*, an advanced programming interface (API) available in all modern web browsers [7]. The Web Audio API allows for very precise synchronous playback of and switching between multiple audio sources.

As shown in Figure 2, each individual audio source is connected to one individual gain node, which is used for mixing or switching between sources. The outputs of these gain nodes are all connected to one common gain node, which is used for temporarily lowering overall volume while seeking to prevent clicking. This common gain node is in turn connected to the audio output sink.

Because in many cases high fidelity audio examples are required, all audio sources are expected to be either uncompressed or compressed at a high bit rate. As browser vendors each only support an individual subset of all available audio formats [10], audio sources can be defined to have multiple input files in multiple different formats. The browser is instructed to pick any of the formats it supports and ignore all other files of a source.

Secondly, due to the very nature of a multitrack player, possibly a large list of files needs to be downloaded before playback. This, together with the growing use of mobile web browsers and mobile internet data plans, dictates that preloading of audio files on page load should be avoided. In-

```
<link href="trackswitch.css" rel="stylesheet" />

<div class="player">
  <!-- Seekable visualization image, with custom margins to set start and end of time axis -->
  <img class="seekable" src="waveforms_input.png" data-seek-margin-left="12.8"
      data-seek-margin-right="9.8" />

  <!-- First track, multiple sources for Browser compatibility -->
  <ts-track title="Mixed signal">
    <ts-source src="mixed.mp3" type="audio/mp3" />
    <ts-source src="mixed.wav" type="audio/wav" />
  </ts-track>

  <!-- Second track, custom style -->
  <ts-track title="Reference foreground signal" style="background-color: #bfd8ee;">
    <ts-source src="reference_f.mp3" type="audio/mp3" />
    <ts-source src="reference_f.wav" type="audio/wav" />
  </ts-track>

  <!-- Third track, bold font face -->
  <ts-track title="Estimated foreground signal" style="background-color: #c5dfa3; font-weight:
      bold;">
    <ts-source src="est_f.mp3" type="audio/mp3" />
    <ts-source src="est_f.wav" type="audio/wav" />
  </ts-track>
</div>

<script type="text/javascript" src="trackswitch.js"></script>
<script type="text/javascript">
  $(document).ready(function() {
    $(".player").trackSwitch();
  });
</script>
```

Figure 3: Example *trackswitch.js* parameterization for Figure 1.

stead, a prominent button to enable each *trackswitch.js* instance and to start preloading is shown to the user. While preloading the audio files, an animation is shown while the interaction elements remain locked. After all tracks have finished loading, the user interface is unlocked and playback can start.

3.3 Visualization Engine

To effectively convey the message of scientific results, many researchers make use of figures in their submitted publications. This means in many cases the researcher already has figures prepared or knows how to create useful visualizations with already familiar tools. Hence, to allow as many visualization types as possible without having to implement and teach a new interface, no actual visualization primitives need to be implemented in *trackswitch.js*. Instead, the researcher is asked to prepare one or multiple image files to be embedded in the player. These images can be created using plotting tools, e. g., *MATLAB*, *matplotlib*, *gnuplot* or any other plotting library that supports saving images.

Optionally, to highlight certain aspects in the visualization, *trackswitch.js* can switch between multiple visualization images, depending on which source is currently selected. If required, an interactive seekhead can be overlaid over the visualization images, to allow the user to intuitively jump to time-positions annotated in the image.

3.4 User Interaction Options

By default, *trackswitch.js* shows a list of tracks, all of which are played back simultaneously. Play, pause, and repeat buttons, as well as a seekhead and a timestamp display are shown to the user. For each audio track, one list item with the source title and toggles to mute or solo each track is shown. Just like most modern DAWs, “solo” can be enabled for multiple tracks at once, while all other tracks are implicitly muted.

While many experiments have common requirements for modes of interaction with the user, some possibilities to enable or disable certain interaction features are required. To this end, *trackswitch.js* supports the following modes:

1. `nomute`: Disable the mute-button for all tracks. Tracks can only be soloed, but not muted individually.

2. `nosolo`: Disable the solo-button for all tracks. Tracks can only be muted, but not soloed individually.

3. `radiosolo`: By default clicking solo on multiple sources will enable them all. After enabling `radiosolo` only one source will be allowed to be active at any given time.

4. `onlyradiosolo`: Combines `nomute` and `radiosolo`.

5. `repeat`: To improve ease of use, especially for very short audio examples, the repeat toggle can be enabled by default.

3.5 Stylesheet

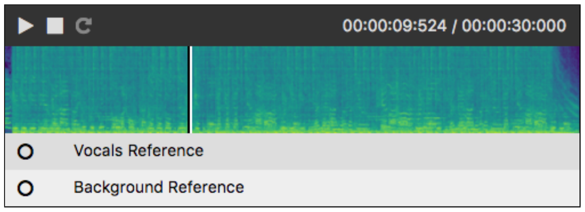
The *trackswitch.js* widget is designed to be responsive. The width of player, visualization, and track list continuously adapt to changes of the screen width. For narrow screens, like mobile touch devices, all interactive elements, like playback control buttons, grow in size to allow comfortable usage. The entire stylesheet contains a reset-stylesheet and is prefixed with a custom widget class to minimize leakage of CSS attributes into *trackswitch.js* and out of *trackswitch.js* into other elements. Each track allows setting custom CSS styles in the parameter markup. This allows highlighting single sources in the player widget. As an example see Figure 3.

3.6 Parameterization

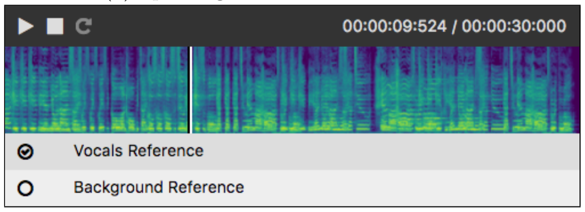
All individual track parameters are set in the document markup. The player is wrapped by any HTML element, inside which each track is defined as a `ts-track` element with common attributes like title, style or a custom visualization image. Inside each track there can be one or more `ts-source` elements, which point to audio files and have an optional media type attribute. Global options like those in Section 3.4 are set in the constructor call of the JavaScript routines. As an example, please refer to Figure 3, which shows the parameterization to reproduce the *trackswitch.js* player from Figure 1.

3.7 Source Code Release

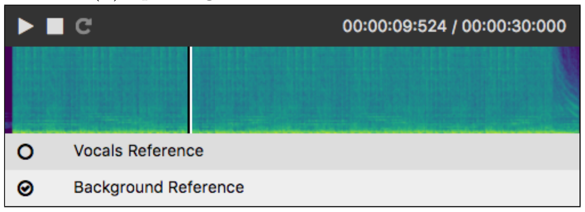
The source code for *trackswitch.js* will be released on `https://github.com/audiolabs/trackswitch.js`. *trackswitch.js* itself comprises a single JavaScript file and a single CSS file, minified versions of both exist for turnkey use. Also, packages for Node.js package manager (*npm*) and *Bower* will be released.



(a) Spectrogram visualization for mix.



(b) Spectrogram visualization for vocals.



(c) Spectrogram visualization for background.

Figure 4: Application of *trackswitch.js* for sound source separation. Toggling the "solo" buttons switches the tracks or mixes them if multiple tracks are selected.

4. APPLICATION EXAMPLES

In the following section we will highlight use cases from several different audio research fields in which *trackswitch.js* has successfully been used to present results.

4.1 Sound Source Separation

Sound source separation describes the task of separating an acoustic mixture into its original components. It is known that humans can easily accomplish this task, however, it is very challenging for machines. Sound source separation is a very active field of research, as the problem is only partly solved until today.

When presenting new methods to address the source separation problem, one way to assess the separation performance is to use objective evaluation measures such as proposed in [27]. Computed scores from these tools, however, do not necessarily correlate with human auditory perception. Therefore, it is still common to present audio separation results along with the objective scores.

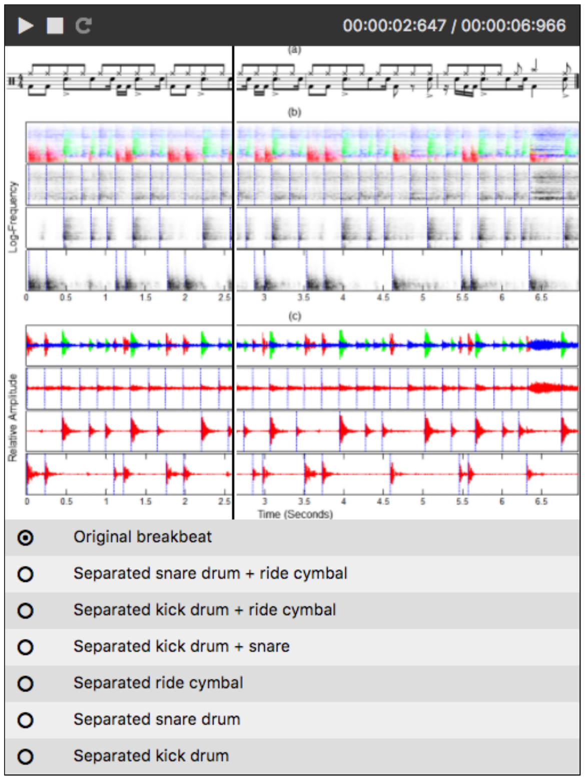
Presenting source separation results typically yields in three classes of audio tracks per example: Sound Mixture, target source (also known as reference), and estimated source. Listeners typically switch between these three tracks to assess the quality of a given source separation result. Furthermore, since many source separation algorithms deal with linear mixtures, *trackswitch.js*'s ability to mix tracks enables the user to reconstruct the mixture by summing up the estimated source tracks. Figure 4 shows how *trackswitch.js* is used to present source separation results generated by a recently developed music source separation approach [4] based on [25]. By using *trackswitch.js*'s `nomute`, users are able to individually mix any combination of tracks for easier comparison. In addition, the linked spectrogram representation switches according to the selected audio track so that users can also visually inspect the separation performance.

4.2 Drum Decomposition

This application demonstrates the usage of *trackswitch.js* in the context of score-informed separation and restoration of drum sound recordings. The goal of this scenario is to decompose drum solo recordings (especially breakbeats) into their constituent drum sound events corresponding to different instruments such as snare drum, kick drum, and cymbals. Figure 5 shows the algorithmic output of separated drum sounds applied to the famous "Amen Break" from the song "Amen, Brother" by "The Winstons" [21]. This example consists of seven tracks with the original song as the first track, followed by different decomposition results. The corresponding visualization of the results consists of sheet music, spectral, and temporal representations of the decomposed sources. By allowing only one track being active at a time, the user is able to compare the different separated drum tracks to the original drum sound mixture, while the seekhead indicates the current playback position. Additional examples can be found on the accompanying website [15].

4.3 Ethnomusicology

The analysis of recorded audio sources has become increasingly important in ethnomusicological research. Such audio material may contain important cues on performance practice, information that is often lost in manually generated symbolic music transcriptions. In this application, *trackswitch.js* is used to make audio material as well as various annotations publicly available for a musically relevant audio



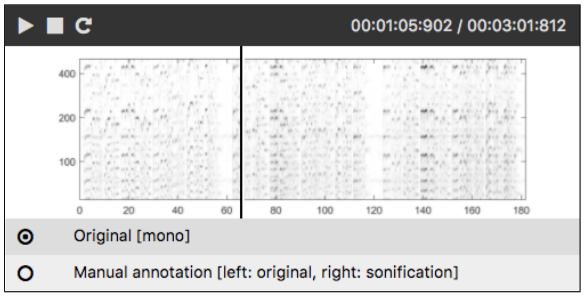
[✓ Original breakbeat]
Separated snare drum + ride cymbal
Separated kick drum + ride cymbal
Separated kick drum + snare
Separated ride cymbal
Separated snare drum
Separated kick drum

Figure 5: Presenting results obtained from applying audio decomposition techniques to the "Amen Break". The player comprises 7 tracks with several visualizations including sheet music, spectral, and temporal representations of the decomposed sources. Furthermore, *trackswitch.js*'s seekhead functionality is used to indicate the current playback position.

collection that consists of more than 100 three-voice polyphonic Georgian chants [23]. The corpus was enriched with manual annotations of the voices' fundamental frequencies (F0) and subsequently used in the experiments. On the accompanying website [6], segment boundary annotations as well as the F0-annotations for each of the songs have been made available in a simple CSV format. Furthermore, *trackswitch.js* is used provide a direct access to visualizations and sonifications of the F0-trajectories. As an example, Figure 6a shows the music recording "The Angels in the Heaven" (ID 002) with its corresponding spectrogram. When switching to the second track, the spectrogram is overlaid with the manual boundary segmentation and F0-annotations, see Figure 6b. Furthermore in this track, the left channel contains the original music recording and the right channel contains a sonification of the manual F0-annotations. This allows researchers to conveniently explore and interact with the audio material and the annotations.

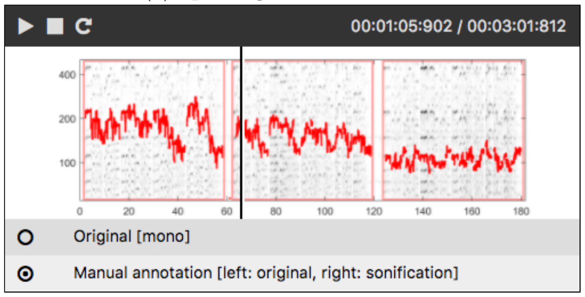
5. CONCLUSION

In this work, a simple but flexible multi track player for presenting scientific audio results on the web has been introduced. We compared existing implementations to our requirements and concluded that none of them fits our profile. After discussing the technical implementation details of our solution we presented a few examples from a wide range of audio research where *trackswitch.js* has successfully been used. *trackswitch.js* will be released as open source software to the public.



[✓ Original [mono]]
Manual annotation [left: original, right: sonification]

(a) Spectrogram visualization.



Original [mono]
[✓ Manual annotation [left: original, right: sonification]]

(b) Spectrogram and annotation visualization.

Figure 6: Example player instance, 2 tracks, with spectrogram visualization. Fundamental fundamental frequency annotation in second figure is only shown if second track is selected.

6. ACKNOWLEDGMENT

The authors would like to thank Bastien Liutkus for his work on *webbasedHTMLplayer* [11], which was the inspiration for this project. Also, we thank Jack Hallybone for his extensive work on integrating the WebAudio API, general playback smoothness and seekable images.

The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS.

7. REFERENCES

[1] Amped Studio. https://ampedstudio.com/. Accessed: 2017-04-06.
[2] AngularJS. https://angularjs.org/. Accessed: 2017-04-06.
[3] Audiotool, produce music online. https://www.audiotool.com/. Accessed: 2017-04-06.
[4] Common fate transform for supervised source separation. https://www.audiolabs-erlangen.de/resources/2017-SISEC-COMMONFATE. Accessed: 2017-04-06.
[5] Font Awesome, the iconic font and CSS toolkit. https://jquery.com/. Accessed: 2017-04-06.
[6] Interactive fundamental frequency estimation with applications to ethnomusicological research. https://www.audiolabs-erlangen.de/resources/MIR/2017-GeorgianMusic-Erkomaishvili. Accessed: 2017-04-06.

[7] Javascript API features browser support chart. http://caniuse.com/#feat=audio-api. Accessed: 2017-04-06.

[8] jPlayer, HTML5 audio & video for jQuery. http://www.jplayer.org/. Accessed: 2017-04-06.

[9] jQuery, write less, do more. https://jquery.com/. Accessed: 2017-04-06.

[10] Media formats supported by the html audio and video elements. https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats. Accessed: 2017-04-06.

[11] multitrackHTMLPlayer, automatic various multitrack audio player. https://github.com/binarymind/multitrackHTMLPlayer. Accessed: 2017-04-06.

[12] Peaks.js, browser-based audio waveform visualization. http://waveform.prototyping.bbc.co.uk/. Accessed: 2017-04-06.

[13] plyr, a simple, accessible HTML5 media player. https://plyr.io/. Accessed: 2017-04-06.

[14] React, a javascript library for building user interfaces. https://facebook.github.io/react/. Accessed: 2017-04-06.

[15] Reverse engineering the amen break – score-informed separation and restoration applied to drum recordings. https://www.audiolabs-erlangen.de/resources/MIR/2016-IEEE-TASLP-DrumSeparation. Accessed: 2017-04-06.

[16] Soundtrap, make music online. https://www.soundtrap.com/. Accessed: 2017-04-06.

[17] Video.js, the player framework. http://videojs.com/. Accessed: 2017-04-06.

[18] waveform-playlist, multitrack web audio editor and player with canvas waveform preview. https://github.com/naomiaro/waveform-playlist. Accessed: 2017-04-06.

[19] Web Audio API, w3c working draft 08 december 2015. https://www.w3.org/TR/webaudio/. Accessed: 2017-04-06.

[20] M. Buffa, A. Hallili, and P. Renevier. MT5: a HTML5 multitrack player for musicians. In *WAC 1st Web Audio Conference January 26-28, 2015 - IRCAM & Mozilla Paris, France*, Paris, France, May 2015. IRCAM & Mozilla Paris.

[21] C. Dittmar and M. Müller. Reverse engineering the amen break – score-informed separation and restoration applied to drum recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9):1531–1543, 2016.

[22] S. Kraft and U. Zölzer. Beaqlejs: Html5 and javascript based framework for the subjective evaluation of audio quality. In *Linux Audio Conference, Karlsruhe, DE*, 2014.

[23] M. Müller, S. Rosenzweig, J. Driedger, and F. Scherbaum. Interactive fundamental frequency estimation with applications to ethnomusicological research. In *Proceedings of the AES International Conference on Semantic Audio*, Erlangen, Germany, 2017.

[24] M. Schoeffler, F.-R. Stöter, B. Edler, and J. Herre. Towards the next generation of web-based experiments: a case study assessing basic audio quality following the ITU-R recommendation BS. 1534 (MUSHRA). In *1st web audio conference, Paris, France*, 2015.

[25] F. Stöter, A. Liutkus, R. Badeau, B. Edler, and P. Magron. Common fate model for unison source separation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, 2016.

[26] P. Vandewalle, J. Kovacevic, and M. Vetterli. Reproducible research in signal processing. *IEEE Signal Processing Magazine*, 26(3):37–47, May 2009.

[27] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, July 2006.