DrSax.js: a JavaScript based Unified Web Audio Library and Framework

Euyshick Hong
MARTE Lab.
Dongguk University
Jung-gu Pil-dong, Seoul, Korea
antaresax@dongguk.edu

Jun Kim
MARTE Lab.
Dongguk University
Jung-gu Pil-dong, Seoul, Korea
music@dongguk.edu

ABSTRACT
Web audio technologies have recently been greatly developed by web developers and software companies, providing web audio libraries and frameworks for audio processing and synthesizing using JavaScript. JavaScript has developed into flexible programming language with front end and server side capabilities, providing dynamic interactions with the web. This paper describes a united sound library and framework system developed using web technologies with JavaScript for musical instruments and voice through a web audio API for artistic expression and various musical applications.

CCS Concepts
• Applied computing → Engineering • Applied computing → Sound and music computing •Computer systems organization.

Keywords
Web Audio library; JavaScript; web audio API.

1. INTRODUCTION
Web technologies have rapidly developed with the advent of JavaScript and HTML5, including a wide range of frame-works, libraries, and multimedia web applications that take advantage of web audio API[1] [2][8][10] and the server side platform [4]. Using JavaScript [7], web audio API can build novel web audio applications with multimedia factors for audio processing, synthesizing, visualization, and media control for web based applications.

Several of interactive networking systems have been developed. For example, Play the light of Monet [5][2] allows collaborative works in real-time, such as musical ensembles where players can discuss each other's playing real-time at the same location, or spatially distant, using instrument interfaces to mobile phones, tablets, laptops, or desktop. Users access the web application through the internet. Such methods for musical collaboration can be rapidly built using various web audio API and libraries thanks to current computer speed and technology. However, most current web audio applications and frameworks struggle to correctly integrate the API and libraries within their frameworks.

Many web audio libraries and frameworks have been created to extend web audio applications, and many new web audio applications have been provided. These have a number of common issues and advantages. Using a library has the advantage of helping to quickly and easily develop new web applications, allowing many people to enjoy it simultaneously. However, it is difficult to properly interface with another libraries due to name and function overlaps within the various libraries. Web audio applications require visualization and data controls to enable multiple user interactions, such as dials; buttons; slides; and pitch and volume meters. However, the audio libraries do not support these extra functionalities. Thus, to correctly incorporate various musical applications, it is necessary to use a unified web audio library and framework that includes visualization and provides a comprehensive base for the applications. Therefore, we developed a totally new web audio library and framework, called DrSax.js, to resolve these issues. DrSax.js is provided as the JavaScript library and realizes a united web audio library and framework, making it possible for users to quickly and easily build web applications to control real-time audio processing, visualization, sound effects, and media installation without requiring further libraries or APIs. It is designed for use by non-specialists, with a particular focus on artistic emotions, and is suitable for not only web audio applications, but also various other interactive work.

2. RELATED WORKS
Web audio libraries and frameworks with web audio APIs have been provided from various sources. Typical examples include: Interface.js [13] enables control of web sound applications through a server interface to mouse, touch screen, and gesture capable PCs or portable devices. Gibberish.js [13] is an optimized sound processing library that provides a useful framework system for audio synthesizers that can be used for PC and mobile phones.

Some other libraries have taken different approaches. WA-AX [12] [9] provides and expanded JavaScript library and framework to enable audio processing based with a web audio API, including useful features such as an audio frame-work for web based applications, and supports development of various web based work faster and with less coding. It is modular and extensible Tuna.js [1] is a library of audio effects and processing for web audio applications. It can be employed to build sound processing effects and can connect other audio units by API nodes.

Audiolib.js [6] is a novel audio processing library for web frameworks focused on sound effects that builds the audio framework in a manner comfortable for musical expression. WebGL [11] was developed for visual control using JavaScript,

ICACS '17, August 10–13, 2017, Jeju Island, Republic of Korea

ACM ISBN 978-1-4503-5284-0/17/08…$15.00
https://doi.org/10.1145/3127942.3127953

[1] www.w3.org/TR/webaudio

[2] https://youtu.be/uZLOY8onwz4

providing a cross-platform framework for 3D graphics APIs based on OpenGL ES 2.0, exposed through the HTML5 Canvas elements. Extree.js [3] is a JavaScript 3D library that can facilitate easy interaction for visual work through the web. Figure 1 shows a simple web based FM synthesizer application developed with DrSax.js library and framework. It is available for free download at: https://drsax.github.io/DrSAX/examples/demo_fm.html

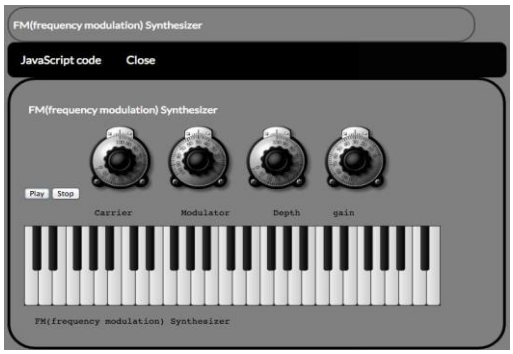

Figure 1. Web FM synthesizer with DrSax.js.

3. DESIGN CONCEPT OF DRSAX.JS

DrSax.js's design concept was to provide a useful web audio application through library elements and frameworks for sound processing and interactive working, that was easy to use without requiring extra libraries or frameworks for various web applications. Thus, it would provide a unified web audio library and framework to build web applications.

3.1 Sound Units

DrSax.js audio units can be separated into four parts. Sound units consist of synthesizers for Frequency modulation, Amplitude modulation, and subtractive modulation; and basic oscillators. These provide not only basic synthesizer roles, but also sound processing to connect multiple input sources, such as microphone, audio file, or various oscillators. Audio units are sound/audio frameworks. An uploaded audio file can be controlled by various control functions.

Table 1. Sound frameworks

Mode	Type
synthesizer	Frequency modulation Amplitude modulation Subtractive modulation basic oscillator
audio file	audio player audio upload audio control
sound input	mic input tunner
effects	equalizer reverb/delay compressor stereo panner

Sound input units include basic microphone input and tuning frameworks. Tuning is a significantly useful function in the musical area, providing sound processing elements, such as equalizers, reverb and delay, compressor, and stereo panning. These are mainly effect elements in sound frame-works. Subsequent sections discuss main sound units and the detailed method to build novel web audio applications.

3.2 Sound Visualization

Sound visualization is a useful method to monitor changes in volume, pitch, and timbre without requiring amplifiers or sound outputs. For example, sound input checks are of-ten required before a stage concert or recording. DrSax.js supports frequency and amplitude domain sound visualization. The HTML5 has Canvas API is incorporated to provide web based visualization canvas control for images and videos, as well as microphone input and visualization. An example application highlighting these features is freely available at: https://drsax.github.io/DrSAX/examples/ demo_mic.html

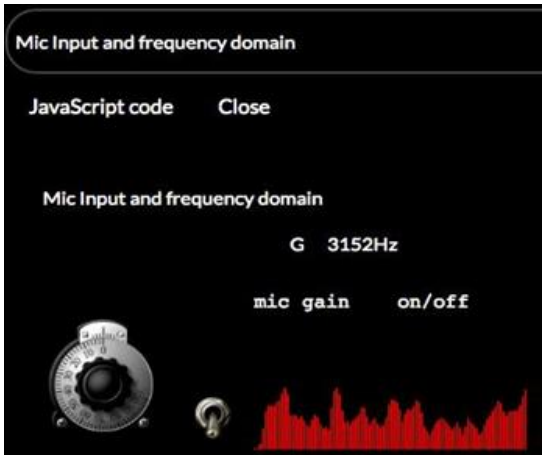

Figure 2. Mic input and visualization.

Table 2. Visualization and data control frameworks

Mode	Type
visualization	frequency domain amplitude domain
value control	data function control data toggle control

3.3 Data Control

Data controllers were designed for real-time control of different playing styles, which is and essential element for interactive frameworks. Two types of data controller are provided: on/off toggle can control frequency, volume, and modulation parameters without requiring extra functions; and volume and pitch control is a toggle API that enables control of on/off frameworks in the application.

Table 3. Data value control

Mode	Type
control value	frequency range amplifier range modulation range
toggle control	mic/sound/osc on/off frequency input(piano keyboard)

4. PROGRAMMING

Libraries and frameworks incorporate various type and features. The major focus for DrSAX.js was that its use and programming should be fast, easy, and simple. It must also provide natural coding without requiring further libraries or frameworks to ease

creating novel musical works. The final DrSax.js library is a modern web audio framework that is smaller, faster, and easier to learn and use than current web audio frameworks.

4.1 Implementation with DrSAX.js

```
1
2  <script  src= " DrSax.js " > </script >
3
4  <script >
5
6    var  DSX  =  new  DSX
7    var  osc  =  new    DSX.Osc ({
8                          type: " sine " ,freq:700 }) ;
9    var  am  =  new    DSX.AM ({
10                mod_type  :" sine "
11 ,              modfreq:200 , depth:0.5 , gain:0 . 5 }) ;
12   var  sax_am1  =new  D S X . v a l u e C h a n g e ( "am1 "
         ,osc.freq ) ;
13   var  sax_am2  =new  D S X . v a l u e C h a n g e ( "am2 "
         ,a m . m o d f r e q ) ;
14   var  sax_am3  =new  D S X . v a l u e C h a n g e ( "am3 "
         ,am.depth ) ;
15   var  sax_am4  =new  D S X . v a l u e C h a n g e ( "am4 "
         ,am.gain ) ;
16
17      am . get (osc )
18      a m . c o n n e c t ( gain )
19      g a i n . c o n n e c t (DAC )
20      os c. st ar t () ;
21
22
23      am.stop () ; //    amp  off
24
25  </script >
```

Listing 1: DrSax.js setting and synthesizer

To use DrSax.js in web applications, simply download Dr-Sax.js from the tutorial site and add a script tag to DrSax.js in the web page, e.g. line 2 in Listing 1, then declare "var DSX = new DSX" on the top. Listing 1 shows an example of Amplitude modulation (AM) synthesizer code. Line 7 shows a sound oscillator with type and frequency. The AM synthesizer has 4 data values and declare from 12 to 15 and to control the frequency, modulator, depth, and gain parameter. Line 17 shows how to connect an oscillator and AM with gain. Lines 20 and 23 show how to play and stop.

4.2 United Frameworks

4.2.1 Sound File Controller

```
1 <input id="inputS" type="file" accept="audio/*">
2
3  var DSX = new DSX;
4  var gain = new DSX.Amp({gain: 0.5});
5  var file_upload =new DSX.BGsound('inputS');
6  var st =new DSX.valueChange("gain",gain.gain);
7  var play = document.querySelector('.play');
8  var stop = document.querySelector('.stop');
9  play.onclick = function(){
10              file_upload.connect(gain)
11               gain.connect(DAC)
12   }
13  stop.onclick = function(){
14    file_upload.stop();
15         }
```

Listing 2: Audio file control

DrSax.js supports tree-type sound control units and frame-works. This provides sound file controls much as an audio player: play, stop, pause, and volume control. Listing 2 shows example code to control a sound file. Line 1 is the input tag and line 3 declares "DSX" the code constructs. Lines 5–8 show setting and controlling the uploaded gain of a sound file, play and stop. Other sound file units have similar patterns, but unique features. We can use it by various type.

4.2.2 Sound Input and Visualization

Visualization depend on sound volume and pitch that to get a sound input source. The microphone constructs in line 3 (Listing 3) have initial values set by object literals. Listing 3, lines 2–4 show how to set microphone input and visualization, with line 4 immediately setting a color based on the received data, enabling interactive visualization in real-time using the HTML5 support canvas API. The sound balance can be mixed before playing or recording. Lines 7 and 8 show how to connecting sound inputs to gain and sound out(DAC).

```
1  var DSX = new DSX;
2  var gain = new DSX.Amp({gain: 0.5});
3  var saxInput = new DSX.Mic();
4  var frequrncy_canvas = new DSX.freqDomain('canv', 'red');
5   frequrncy_canvas.getAnalyser(gain);
6
7   saxInput.connect(gain);
8   gain.connect(DAC);
```

Listing 3: Mic input and visualization

4.2.3 Effects Data Control

Listing 4 lines 1–4 show how to configure a 500 Hz saw-tooth oscillator with amplitude gain value 0.5. Line 5 shows how to set delay effects and data values, such as delay time (200ms) and feedback (0.45 s) key value simple that much key data parameters make chaos to control in real-time. Lines 8–10 show how to connecting osc to gain to delay to DAC(output). The example includes various effects: re-verb, delay, panning, equalizer, and compression, and can mix each effect and harmony to expand sound processing.

```
1  var gain = new DSX.Amp({gain: 0.5});
2  var osc = new DSX.Osc({
3            type:"sawtooth",freq: 500
4             });
5  var Delay = new DSX.Delay({
6          delayTime : 200, feedback: 0.45,
7            });
8       osc.connect(gain);
9       gain.connect(Delay)
10      Delay.connect(DAC);
```

Listing 4: Effects value control

5. USES OF NOVEL FEATURES

5.1 Mobile Instruments and Effecter

DrSax.js facilitates creating web based wind instrument using sound input frameworks, such as Ocarina[3] and Webxophone[4]. The

[3] Ocarina : iPhone application into a flute-like instrument by Ge wang

[4] Webxophone: web saxophone instrument by Euy Shick Hong

result is a mobile wind instrument application that accepts sound input from microphone or speaker-phone on a mobile phone. Applications using DrSax.js can achieve harmony web audio technology with a built-in de-vice in mobile that touch screen, speakerphone, mic-phone, built-in sensors. The application can controlled octave frequency by an angle in a smart-phone that up, normal and down, and sound input pitch and volume in mobile or touch portable device. Incorporating this technology in a smart-phone allows development of novel web wind instruments and various works and harmony another multi-media works.

Figure 3. Dr.Saxoman: interactive web audio application with Dr.saxophone II.

5.2 Multi-media Works

Dr.Sax.js is not limited to web audio applications alone, but is also useful for various multimedia applications. The de-signed includes four frameworks, synthesizing the web audio API, controlling sound files, sound visualization, and data control. Thus, Dr.Sax.js can be combined with what to produce novel multimedia platforms for media art installations or interactive performances, such as discussed in the introduction (Play the light of Monet [5]).

6. PERFORMANCE WORKS

6.1 Interactive Performance System

An example interactive web audio network system was con-structed using DrSax.js, including a web audio effector - (Dr.Saxoman) and a hybrid saxophone interface (Dr.Saxophone II) in Figure 3, as shown in The integrated saxophone system can control the sound processing data value in Dr.Saxoman in real-time that wireless networking sys-tem. The example system was used for an interactive live performance “A White Night with Dr.Saxoman and Dr. Saxophone II on stage by Euy Shick Hong. The video of the performance is available at: "Seeing Sound Listening Image -Interactive Performance "2016.11.18 in Lee Hae Rang theater.[5]

6.1.1 Dr.Saxoman

Dr.Saxoman is a web audio application for saxophone or wind instrument and voice that supports sound input, sound tuner, visualization, audio file, recording, effects, equalizer, compressor, AM, pitch shift, reverb, delay, and stereo panning. All data values can be controlled mouse or Dr. Saxophone II and mobile in real time.

- sound input : mic input, tuner.
- effect : EQ, compressor, pitch shift, reverb, delay, panning.
- synthesizer : FM, AM, subtractive.
- interactive system : netwoking with DrSaxophone II or mobile phone.

6.1.2 Dr.Saxophone II

Dr.Saxophone II is hybrid saxophone interface to play various performances in a multi-media performance that expanded Dr.Saxophone that was developed and presented in The 2016 3rd International Conference on Systems and Informatics(ICSAI 2016). The interface device is controlled by a natural gesture that fingering and lift up and down with saxophone. The data of interface sensors are digitized through the note fingering and body gestures during playing. We can be applied to other wind instruments or voice for various media works such as harmony with Dr.Saxoman on web(video :https://youtu.be/do_yuLdVTCo).

6.2 Web application demonstration

The Dr.Sax.js tutorial provides a simple web demonstration, as detailed in Section 8. So, we can find web demo application at each article link in tutorials. DrSax.js presents simple code and demo web application and detail information.

7. CONCLUSIONS

Web developers and sound artists have used web based audio technologies to developed many audio libraries and frameworks for audio processing and synthesizing using JavaScript. This paper described a united web audio library and framework system developed in JavaScript for interactive real-time integration of saxophone and voice through the web audio API. The application is embodied in DrSax.js and is suitable to build web audio applications without requiring extra libraries or frameworks.

The web based audio applications can easily control real-time audio processing, visualization, sound effects, and media installation, and work well for expressing artistic emotions and interactive musical works.

8. TUTORIAL AND WEB AUDIO APPLI CATION WITH DRSAX.JS

DrSAX.js tutorial site and application in Github site.

- DrSax.js tutorial site:
  https://drsax.github.io/DrSAX/lib.1.8.html.
- DrSax.js link in Github:
  https://drsax.github.io/DrSAX/DrSax.1.7.0.1.2.js.
- DrSaxoman application site:
  https://drsax.github.io/DSX/drsaxjs.1.7.0.0.html

Web audio application with DrSAX.js.

- simple sound oscilattor:
  https://drsax.github.io/DrSAX/examples/demo_osc.1.0.html.
- Delay and Attack and release:
  https://drsax.github.io/DrSAX/examples/demo_attack.html.
- Stereo panning:
  https://drsax.github.io/DrSAX/examples/demo_pan.html.
- Reverb and Attack and Release:

[5] https://youtu.be/THJLy0GvAso

https://drsax.github.io/DrSAX/examples/demo_reverb.html.

- FM(frequency modulation) Synthesizer:
  https://drsax.github.io/DrSAX/examples/demo_fm.html.
- AM(amplitude moduration) Synthesizer:
  https://drsax.github.io/DrSAX/examples/demo_am.html.
- Simple Subtractive Synthesizer:
  https://drsax.github.io/DrSAX/examples/demo_sub.html.
- Audio File upload and 5 band EQ:
  https://drsax.github.io/DrSAX/examples/demo_eq. html.
- Sound file upload player:
  https://drsax.github.io/DrSAX/examples/demo_play.html.
- Audio File upload:
  https://drsax.github.io/DrSAX/examples/demo_play2.html.
- Audio Player and Speed control:
  https://drsax.github.io/DrSAX/examples/demo_playspeed.html.
- Mic Input and frequency domain:
  https://drsax.github.io/DrSAX/examples/demo_mic.html.
- Mic Input and pitch tuner:
  https://drsax.github.io/DrSAX/examples/demo_tune.html.
- Dynamic Compressor:
  https://drsax.github.io/DrSAX/examples/demo_comp.html.
- Mic Input and amplitude domain:
  https://drsax.github.io/DrSAX/examples/demo_ampdomain.html.

9. REFERENCES

[1] An audio effects library for web audio. *https://github.com/Theodeus/tuna*.

[2] Rogers, C. web audio api-w3c working draft. *https://www.w3.org/TR/webaudio*.

[3] Javascript 3d library. *https://github.com/mrdoob/three.js/*.

[4] Node.js. *https://nodejs.org*.

[5] Play the light of monet : interactive web audio networking system. *https://github.com/drsax/monet*.

[6] A powerful audio tools library for javascript. *https://github.com/jussi-kalliokoski/audiolib.js*.

[7] tutorial of javascript,html, css. *https://www.developphp.com/*.

[8] Web audio api extension. *https://github.com/hoch/waax.*

[9] Web audio api extension. *https://github.com/hoch/waax.*

[10] Web audio tutorials -middle ear. *http://middleearmedia.com/category/web-audio- tutorials/.*

[11] Webgl - opengl es 2.0 for the web. *https://www.khronos.org/webgl/*.

[12] Choi, H. and Berger, J. Waax: Web audio api extension. In *Proceedings of the 13th Conference on New Interfaces for Musical Expression(NIME-13), Daejeon, Korea,, 2013.*

[13] Roberts, C., Wakefield, G., and Wright, M. The web browser as synthesizer and interface. In *Proceedings of the 13th Conference on New Interfaces for Musical Expression(NIME-13), Daejeon, Korea,, 2013.*