Tune.js: A Microtonal Web Audio Library

Benjamin Taylor
Goucher College Digital Arts
1021 Dulaney Valley Rd, Baltimore, MD
benjamin.taylor@goucher.edu

Andrew Bernstein
Goucher College Digital Arts
1021 Dulaney Valley Rd, Baltimore, MD
andrew.bernstein@mail.goucher.edu

ABSTRACT
The authors share Tune.js, a JavaScript library of over 3,000 microtonal tunings and historical temperaments for use with web audio. The current state of tuning in web audio is reviewed, followed by an explication of the library’s creation and an overview of its potential applications. Finally, the authors share several small projects made with Tune.js and ponder future development opportunities.

1. INTRODUCTION
As the web browser becomes a development platform for musical instruments, many JavaScript synthesis toolkits have arrived at a common question: how should we handle the musical scale? While several solutions exist for using standard major, minor, and modal scales, the concerns of contemporary music and musicology often demand more complex and flexible musical tunings.

We introduce Tune.js¹ as a toolkit for composing with microtonal scales and historical temperaments in the browser. Tune.js is a port of the vast Scala tuning archive (see 1.3) into JavaScript, resulting in a microtonal tuning API that can integrate with web audio and which is not confined to any specific synthesis toolkit. We hope Tune.js will assist in a wide variety of projects, from recreating historical instruments in the web (similar to the Harry Partch archive² in Flash), to audiovisual projects which make use of pure tunings and just intonation. Finally, we hope that pairing the Scala tuning archive with the accessibility and audiovisual power of the web can lead to new educational tools for just intonation theory and history.

1.1 Just Intonation
Just intonation (JI), consisting of mathematically pure ratios between note frequencies, formed the basis of much of the world’s music until the 16th century. In Renaissance Italy, the growing popularity of tertian harmony, as well as a rise in instrumental music, led to the gradual temperament of the chromatic scale. Harmonic ratios were slightly adjusted – tempered – in order to strengthen the major third (a dissonant interval in Pythagorean tuning), and to make instruments sound acceptable in all keys. A full explanation of just intonation and microtonality is beyond the scope of the paper; one can be found in Harry Partch’s *Genesis of a Music* [7].

In the 20th century, new technologies, cultural cross-pollination, and modern theoretical concerns all led to a resurgence of JI. Composers such as Harry Partch[7], La Monte Young[2], and James Tenney each composed exciting JI works which are now in the canon of 20th century art music. Meanwhile, digital technologies have arisen which afford more sophisticated control of tuning[4]. As we are no longer limited to how many notes can fit on a keyboard, software instruments have been able to solve Renaissance tuning concerns while keeping pure intervals, causing JI to once again be a practical option.

1.2 The State of Tuning in JavaScript
Most popular web audio toolkits address the need to organize melodies and harmonies around a variety of scales, modes, and tunings, and many do so quite successfully. Yotam Mann’s Tone.js has an API explicitly geared towards musical thinking; for pitch, its synths understand music text notations like "C4". Live coding toolkits such as Gibber[1], Lissajous[9], and Lich.js[6] contain dozens of scales and modes; Charlie Roberts’ Gibber web audio language contains over a dozen scales, including jazz modes and at least two just intonation scales[8], while Lissajous includes over 50 scales including many non-Western and pure tunings. However, there are not yet general purpose toolkits for composers who want to use more eccentric scales outside of these toolkits, for example with the Web Audio API directly.

1.3 Scala and MicroTuner
One of the most complete general purpose pitch organization kits on the web is Teoria³, a comprehensive tool for music theory in JavaScript which is marvelous for composing equal tempered music but has no microtonal capabilities. Meanwhile, other projects such as Mitch Well’s Microtonal Web Synth⁴ use complex microtonality in the web, but are standalone instruments and have not been encapsulated for general purpose use in other projects.

¹https://www.github.com/abbernie/tunejs/
²http://musicmavericks.publicradio.org/features/feature_partch.html
³https://github.com/saebekassebil/teoria
⁴http://www.websynths.com/
puter music is the Scala program. Scala, a freeware computer music software first released over 15 years ago, lets users create and hear microtonal scales. It is perhaps the most successful hub of just intonation activity in computer music, thanks to the extensive research of its author, Manuel Op de Coul, who collected thousands of historical tunings and made them available within the program. Scala publishes its archive for open use within not-for-profit projects. For example, Victor Cerullo used the archive to create a custom Max object called Microtuner, which lets Max users access Scala scales within the patcher paradigm.

2. TUNE.JS

Tune.js brings the Scala archive of over 3,000 microtonal scales to JavaScript and web audio. Tune’s API takes MIDI pitch values as input, and returns frequency data, adjusted MIDI values, or a pitch ratio to a tonic (see 2.2). The result is a general purpose library of historically interesting scales which may be used in any web audio instrument, composition, or artwork.

2.1 Construction

In the Scala archive, each scale exists as a list of frequencies stored in a unique .scl file. To build Tune, we procedurally accessed each Scala file and parsed its contents, sorting each scale’s frequency list into a property of a large JavaScript object called TuningList.

For example, this is the contents of the Scala file for Ptolemy’s diatonic JI scale:

```plaintext
// ptolemy.scl
//
// Intense Diatonic Syntonon, also Zarlino’s scale
//
@60 :intervals
261.6255653006
294.32876096318
327.03195662575
348.83408706747
392.4383479509
436.04260883433
490.54793493862
523.2511306012
```

By collecting all scales in one JavaScript object and providing an API for accessing it, we are able to offer an encapsulated library that can be included in other JavaScript projects and does not require a Scala file interpreter.

2.2 API

Tune.js offers a compact API with only a few important core methods.

Loading a Scale

First, a user creates an instance of Tune, loads a scale using .loadScale(), and sets a fundamental frequency for the scale using .tonicize()

```plaintext
tune = new Tune()
tune.loadScale("ptolemy")
tune.tonicize(440)
```

A full reference of accepted scale names is at the Tune.js Scale Archive. To use multiple scales at once, create multiple Tune instances.

Playing Notes

Once a scale is loaded, tune.note() is used to convert scale degrees to musically useful numbers such as frequencies, pitch ratios, or adjusted midi values. By default, the function returns a frequency value that can be used to set the frequency of a web audio synth.

```plaintext
synth1.frequency.value = tune.note(0)
synth2.frequency.value = tune.note(2)
synth3.frequency.value = tune.note(4)
```

Given three web audio synths, this code would set the synths' frequencies to the 1st, 3rd, and 5th steps in the scale, respectively, creating a major chord based on the tonic frequency of 440 in Ptolemy’s diatonic tuning.

If the scale degree input to tune.note() is negative or higher than the scale length, the function automatically wraps the scale degree to apply to the next octave. In the following example, assume a 7-note diatonic scale.

```plaintext
tune.note(7) // the first scale degree, one octave up
tune.note(-1) // the seventh scale degree, one octave down
```

An optional second argument for tune.note() specifies an octave transposition of the scale degree. Therefore, the examples above could equivalently be expressed as:

```plaintext
tune.note(0, 1) // the first scale degree, one octave up
tune.note(6, -1) // the seventh scale degree, one octave down
```

---

5http://www.huygens-fokker.org/scala/
6http://cycling74.com
7http://www.venetica.net/Sites/16tone/mtx_file_specs.htm
8http://abbernie.github.io/tune/scales.html
Output Modes
In addition to outputting frequency values, Tune can output tuning information as adjusted MIDI or as a pitch ratio. This way, Tune may be used to set the frequency of an oscillator (frequency mode), a playback speed for an audio file (ratio mode), or to tune a MIDI synth (MIDI mode). For example, in a diatonic JI tuning in which the fifth scale degree is a perfect 3/2 ratio to the tonic of 440, the three output modes would result in the following:

```plaintext
tune.mode.output = "frequency"
tune.note(4) // returns 660

tune.mode.output = "ratio"
tune.note(4) // returns 1.5

tune.mode.output = "midi"
tune.note(4) // returns 76.019547
```

Input Modes
For compatibility with MIDI devices, Tune has a "midi" input mode in which an input of 60 refers to the tonic, as if it is Middle C on a keyboard. This can be useful in cases of using existing piano rolls, or for connecting Tune to keyboard interfaces which output MIDI values. In "midi" mode, there is no second argument for octave displacement.

```plaintext
tune.mode.input = "midi"
tune.note(60) // returns the tonic frequency
tune.note(62) // returns the third scale degree
tune.note(64) // returns the fifth scale degree
```

The relationship to MIDI pitch notation can be troublesome, however. If a 12-note scale is loaded, the parallel works well: octaves will be heard at 48, 60, 72, and so on. However, if a scale such as "partch43" is loaded, which contains 43 notes, octaves will be heard at 60, 103, 146, and so on, at which point the MIDI pitch analogy becomes less accurate because pitches will exceed the 0-128 MIDI pitch range.

2.3 Scales
Tune.js includes the vast majority of scales in the original Scala archive. Several scales were lost during the porting process due to parsing errors, which the authors are working to remedy. Example scales in Tune.js include:

<table>
<thead>
<tr>
<th>Scale Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ji</td>
<td>Basic just intonation with 7-limit tritone</td>
</tr>
<tr>
<td>harm30</td>
<td>First 30 harmonics and subharmonics</td>
</tr>
<tr>
<td>pyth31</td>
<td>31-tone Pythagorean scale</td>
</tr>
<tr>
<td>ptolemy</td>
<td>Diatonic Syntonon, also Zarlino’s scale</td>
</tr>
<tr>
<td>couperin</td>
<td>Couperin's modified meantone</td>
</tr>
<tr>
<td>partch43</td>
<td>Harry Partch’s 43-tone pure scale</td>
</tr>
<tr>
<td>young-lm</td>
<td>LaMonte Young's Well-Tempered Piano</td>
</tr>
<tr>
<td>slendro</td>
<td>Observed Javanese Slendro, Helmholz</td>
</tr>
<tr>
<td>malkauns</td>
<td>Mode of Indian Raga Malkauns</td>
</tr>
</tbody>
</table>

Table 1: 9 of the 3,000+ scales in Tune.js

Figure 1: Tune.js Demo: Just Intonation Keyboard

Figure 2: Andrew Bernstein's Historicism

3. PROJECTS

Just Intonation Keyboard
The Tune.js demo is a digital piano keyboard with a settable tuning. A dozen different tunings are accessible via a dropdown menu, including scales by Partch, Ptolemy, Pythagoras, Couperin, and Bach. The demo is made using the Qwerty-Hancock HTML5 keyboard and Tone.js. While this keyboard is intended to demonstrate the use of Tune.js, it could also be an educational tool or ear training tool for students learning about just intonation and different historical tunings.

Historicism
Co-author Andrew Bernstein’s artwork Historicism shows the potential of pairing historical tuning, algorithmic composition, and web graphics. For the work, a markovian chain is constructed from a MIDI roll of J.S. Bach’s Prelude in C from The Well-Tempered Clavier, using Node.js. The markov chain is used to construct a generative five-voice composition using piano samples and a mean-tone tempered tuning which is accurate to Bach’s period.

4. CONCLUSIONS AND FUTURE WORK
Tune.js is currently in open-source release as an assistant to tuning JavaScript audio projects. With further development, other potential applications arise. If paired with a web audio analyzer node, Tune.js could act as an acoustic tuner, taking audio input and giving visual feedback. This could encourage the creation of easily accessible sites for piano tuners or other enthusiasts who wish to tune instruments to exotic scales. HTML5 visualizations of these historical scales could also be an educational tool or ear training tool for students learning about just intonation and different historical tunings.
scales could also provide an accessible educational resource for music students. Tune.js has the potential to move just intonation research away from table-based representations\(^\text{11}\) and towards fun and interactive applications accessible on the web.

5. ACKNOWLEDGMENTS

The authors would like to thank the Goucher College Digital Arts community and the web audio community who have helped grow Tune.js.

6. REFERENCES


\(^{11}\)http://www.kylegann.com/Octave.html