Ήχος στο Διαδίκτυο

Εισαγωγή στο Tone.js

Χρυσούλα Αλεξανδράκη
Τμήμα Μουσικής Τεχνολογίας και Ακουστικής
Ελληνικό Μεσογειακό Πανεπιστήμιο

Tone.js Overview

- Tone.js a framework Implemented on top of the Web Audio API offering:
  - Musical abstraction
    - Notes as opposed to audio nodes
  - High Precision Musical Timing
    - *Bars:Beats:Sixteenhs* based on bpm tempo as opposed to audioCtx.currentTIme (sec)
  - Sequencing
  - Simper Synths, Envelopes and Audio FX
  - Fewer lines of code to do the same thing

Example

- Web Audio API

```
const ctx = new AudioContext();
const osc = ctx.createOscillator();
const gain = ctx.createGain();
osc.connect(gain);
gain.connect(ctx.destination);
osc.start()
```

- Tone.js

```
const synth = new Tone.Synth().toDestination();
synth.triggerAttackRelease("C4", "8n");
```

Compare!!

Starting Audio

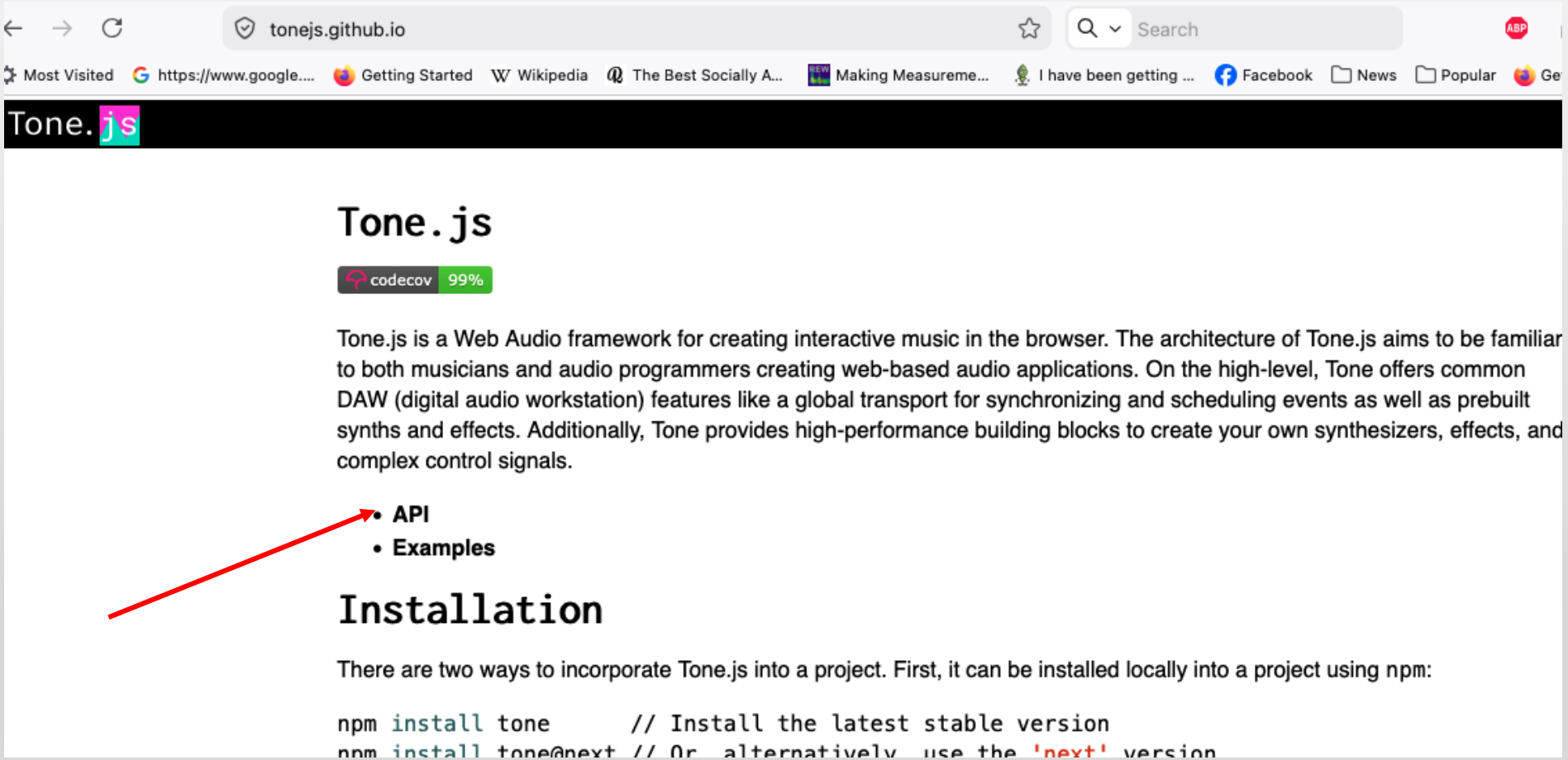
Starting Audio

IMPORTANT: Browsers will not play *any* audio until a user clicks something (like a play button). Run your Tone.js code only after calling `Tone.start()` from a event listener which is triggered by a user action such as “click” or “keydown”.

`Tone.start()` returns a promise, the audio will be ready only after that promise is resolved. Scheduling or playing audio before the AudioContext is running will result in silence or incorrect scheduling.

```
1  //attach a click listener to a play button
2  document.querySelector("button")?.addEventListener("click", async () => {
3      await Tone.start();
4      console.log("audio is ready");
5  });
```

API Reference



Types of Classes

Class Category	Meaning	Representative Examples
Core	infrastructure/audio graph	Gain, Delay
Source	audio generators	Oscilator, Player, FMOscilator
Instrument	playable instruments	FMSynth, MembraneSynth, Pluck Synth
Effect	processors	Chorus, Reverb, PitchShift
Component	reusable DSP building blocks	Envelope, FFT, Panner3D
Signal	control-rate objects	Add, Pow, Scale
Event	timing/scheduling	Sequence, Pattern, Loop
Unit	unit conversion utilities	TimeClass, MidiClass

The general Idea

Source → Instrument → Effect → Destination

↑

controlled by

envelopes / LFOs / sequences

Installation of Tone.js

- CDN, simply include the following line in your HTML code
  - `<script src="https://unpkg.com/tone"></script>`
- CDN- Content Delivery Network
  - A network of servers that distributes files
    - .js, .css, media (audio, video image)
  - Each user receives the file from a server that is in nearby location
    - To avoid latency
  - Popular CDNs
    - Couldflare CDN
    - jsDelivr
    - unpkg
    - cdnjs

Musical Time Notation

- Transport: το **μουσικό ρολόι, ο μετρονόμος:**

```
Tone.Transport.bpm.value = 120;
```

- To use scheduled events, e.g. Sequences, Loops, etc., it is necessary to start the metronome:

```
Tone.Transport.start();
```

- Κάθε ηχητικό γεγονός συγχρονίζεται με το τέμπο ως:
  - "4n" → quarter note
  - "8n" → eighth note
  - "16n" → sixteenth note
  - "1m" → one measure

Δημιουργία Sequence()

- Create and start the sequence

```
const synth = new Tone.Synth().toDestination();
const notes = ["C4", "E4", "G4", "B4"];
//call the playNextNote callback at every quarter note
const sequence = new Tone.Sequence(playNextNote, notes, "4n");
sequence.loop = false; //otherwise it loops infinitely
sequence.start(0);
Tone.Transport.start();
```

- The callback every time a new note needs to be played by the sequence

```
//NextNote Callback
function playNextNote(time, note) {
    //the duration of the note is an eight note
    synth.triggerAttackRelease(note, "8n", time);
}
```

PolySynth

- For Polyphonic playback

```
poly = new Tone.PolySynth(Tone.Synth).toDestination();
poly.triggerAttackRelease(["C4", "E4", "G4"], "2n");
```

Envelopes

- Values in seconds

```
synth = new Tone.Synth({
    envelope: {
        attack: 0.1,
        decay: 0.2,
        sustain: 0.5,
        release: 1
    }
    }).toDestination();
    synth.triggerAttackRelease("C4", "8n");
```