Conservatorio Superior de Música de Murcia
1-2 December 2021

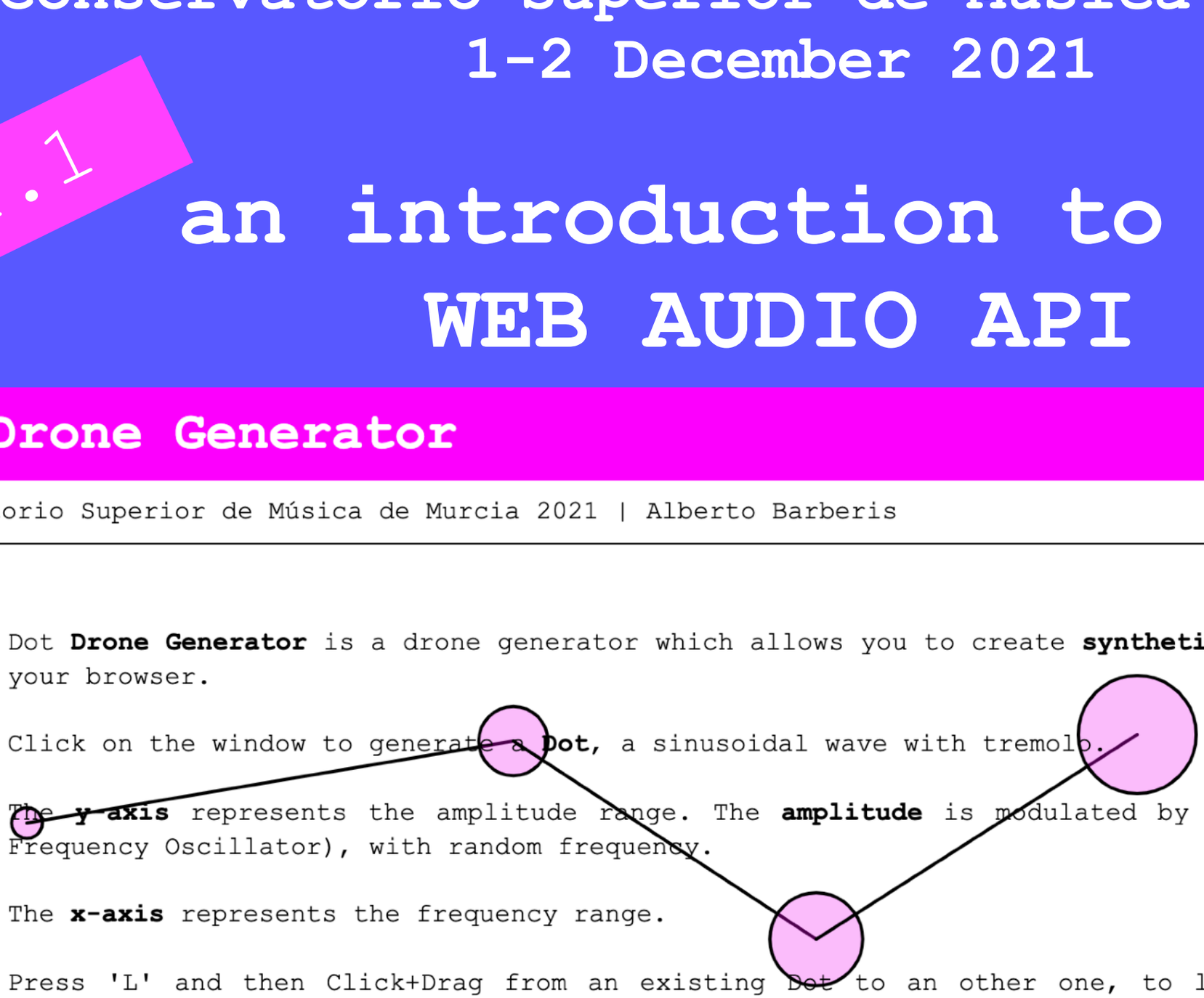
1.1

an introduction to the WEB AUDIO API

Dot Drone Generator

Dot **Drone Generator** is a drone generator which allows you to create **synthetic textures** directly on your browser.

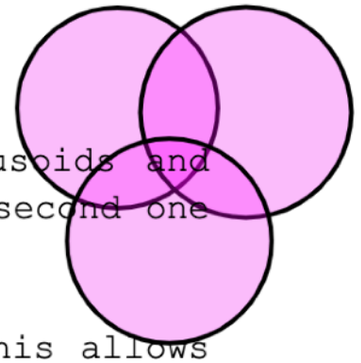
Click on the window to generate a **Dot**, a sinusoidal wave with tremolo.



The **y-axis** represents the amplitude range. The **amplitude** is modulated by a **triangular LFO** (Low Frequency Oscillator), with random frequency.

The **x-axis** represents the frequency range.

Press 'L' and then Click+Drag from an existing Dot to an other one, to link two sinusoids and create a **Frequency Modulation** between them. The first Dot becomes the modulator of the second one (the carrier).



It is possible to create a **chain of modulation**: each carrier can become a modulator. This allows you to create complex spectra, to the point of creating very noisy sounds!

Click on an existing circle to **delete** it or to delete the modulation chain of which it is part.

Alberto Barberis

hello :)

Alberto Barberis -> www.albertobarberis.it

Lecturer in Electronic Music at the Conservatorio della Svizzera Italiana of Lugano (CH)

composer (instrumental and electroacoustic)

engineer (information technology and new media)

guitarist (my origin...)

electronic music performer

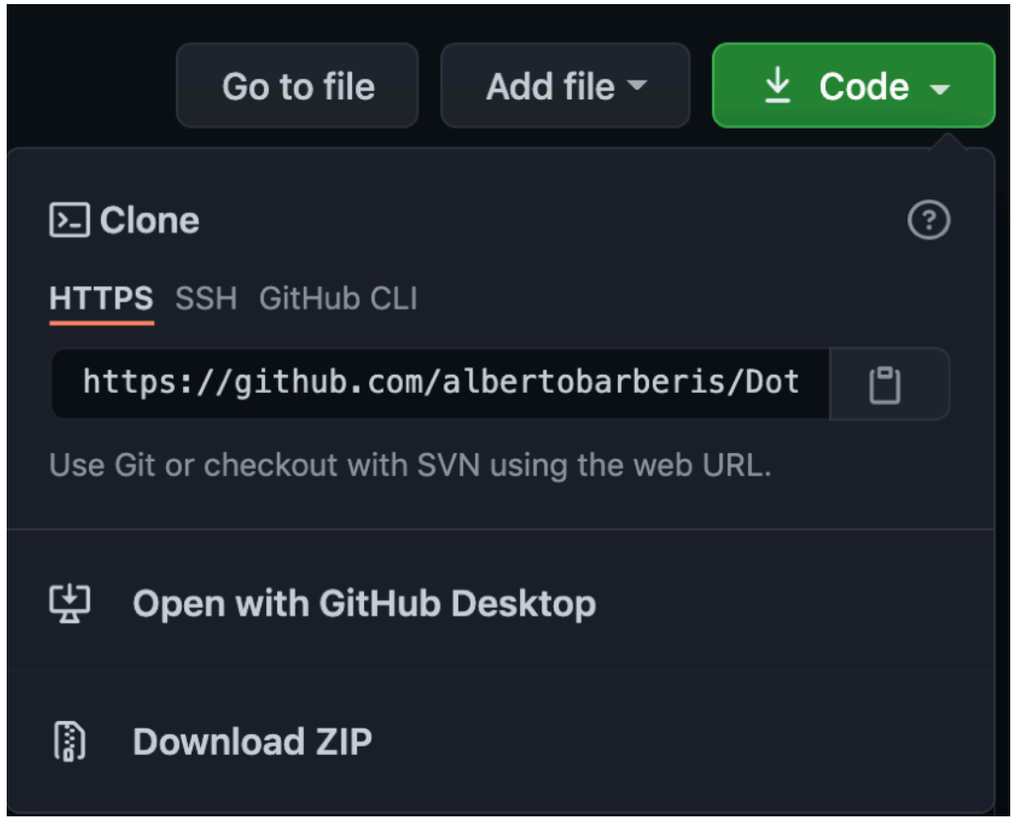
sound designer

code artist and developer

MATERIALS

You find all the **materials** (pdf slides, and code) for the workshop at the following link of a **Github repository**.

https://github.com/albertobarberis/workshop_murcia_2021



Click on the green **Code** button and then on **Download ZIP** to download all the materials from the repository.

GOALS

1. understand what is the **web audio** and what are the basic concepts of the **web audio Application Programming Interface** (the **web audio API**);

2. analyze together **some examples** of **web audio applications**;

3. summarize what are the **possibilities** offered by the web audio for sound design and music composition;

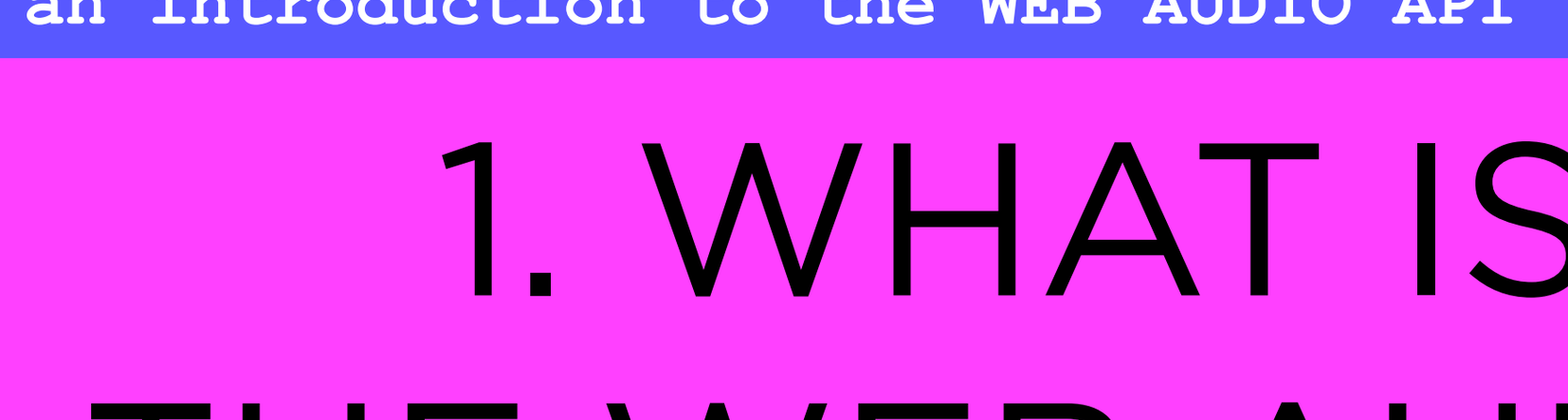
4. understand **what you need to know** to deal with the web audio (web programming: HTML, css, javascript; music theory and sound design; etc.);

5. **develop** a simple **web audio app**: the ***Dot Drone Generator***.

1. WHAT IS THE WEB AUDIO

Dot **Drone Generator** is a drone generator which allows you to create **synthetic textures** directly on your browser.

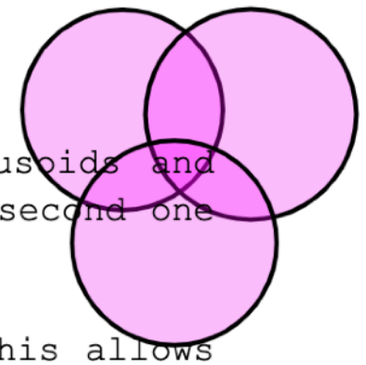
Click on the window to generate a **Dot**, a sinusoidal wave with tremolo.



The **y-axis** represents the amplitude range. The **amplitude** is modulated by a **triangular LFO** (Low Frequency Oscillator), with random frequency.

The **x-axis** represents the frequency range.

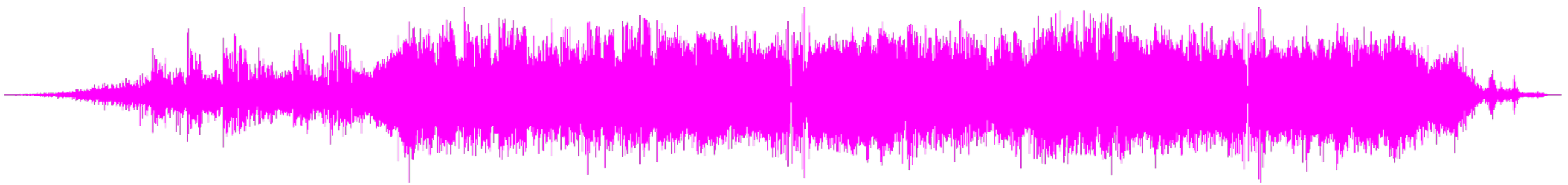
Press 'L' and then Click+Drag from an existing Dot to an other one, to link two sinusoids and create a **Frequency Modulation** between them. The first Dot becomes the modulator of the second one (the carrier).



It is possible to create a **chain of modulation**: each carrier can become a modulator. This allows you to create complex spectra, to the point of creating very noisy sounds!

Click on an existing circle to **delete** it or to delete the modulation chain of which it is part.

1.1 WHAT IS THE WEB AUDIO



With the expression **web audio** we refer to the technologies that allow to **process/synthesize audio signals** and **compose music** in modern **web applications** (running on a browser like Google Chrome, Firefox, etc).

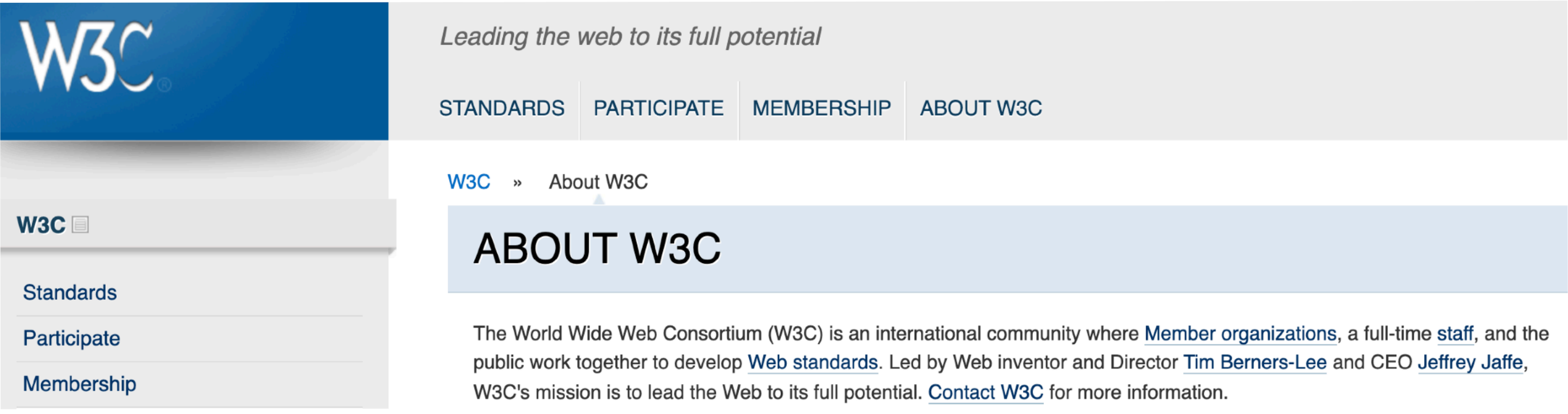
The web audio allows to **create**, **compose** and **manipulate music in real-time** in **different devices** (smartphone, computer, tablet, smart TV, etc.) which use a browser that supports the **web audio API**.

1.1 WHAT IS THE WEB AUDIO

The **web audio** is a rather new technology, made available thanks to the release of the **web audio API** by the **World Wide Web Consortium (W3C)** in 2011.

W3C is the main **international standards organization** for the World Wide Web.

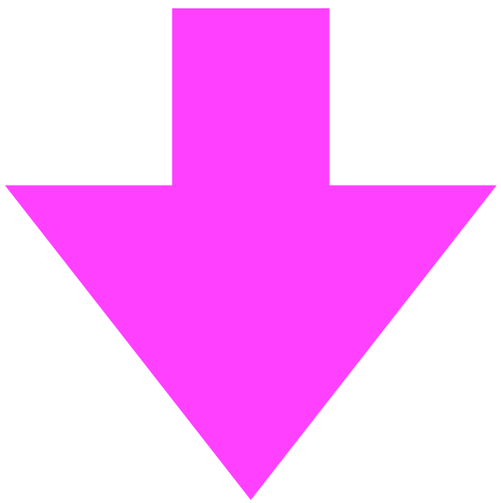
https://www.w3.org/



1.1 WHAT IS THE WEB AUDIO

Before the **web audio API**, audio could be embedded in browsers, but more complex audio effects were only available through the use of plugins.

Dealing with **web audio applications** design, **web audio music**, and **web audio composition** means primarily dealing with **web programming languages**.



What is a programming language?

1.1 WHAT IS THE WEB AUDIO

«**Electronic music is the mainstream**.

[...] Electrical technology has facilitated more than a century of original music, spawning a multitude of new styles, instruments and methods.

[...] It appears ubiquitous, from mobile phones, television and podcasts to the art gallery and other unorthodox performance spaces. In many was, **electronic music is now well accepted and integrated into contemporary practice**.

The **influence of computers in compositional techniques**, **sound analysis and processing**, **performance interfaces** and **concert practice is astounding**».

ELECTRONIC MUSIC, N. Collins, J. d'Escrivàn

1.2 WHAT IS A PROGRAMMING LANGUAGE

A **programming language** is a formal language used to develop *software programs*, *scripts*, or other sets of instructions for computers to execute.

Thousands of different programming languages have been created, and more are being created every year.

Many languages share similarities, but each has its own:

- **syntax** (rules, structure, vocabulary, etc.);
- **semantics** (meaning, logic, characteristics, etc.).

1.2 WHAT IS A PROGRAMMING LANGUAGE

```css
body {
    margin: 0px;
    font-family: 'Courier New', Courier, monospace;
}
h1 {
    margin: 0px;
    padding: 10px;
    background-color: rgb(88, 88, 255);
    color: white;
    -webkit-user-select: none;
}
```

CSS

some examples of web programming languages

```html
<!DOCTYPE html>

<html lang="en">

  <head>
    <title>web audio HS2019</title>
    <meta charset="utf-8">
  </head>

  <body>
    <div>
      this is
    </div>
  </body>

</html>
```

HTML

```javascript
function mouseRealeased() {
    initializeIndex();
}
function keyReleased(){
    initializeIndex();
}

function frequencyModulation(i,j){

    var scalar1=(random(300)+50)*random(
    i.osc.freq(j.freqModulator.mult(scala

    var scalar2=(random(300)+50)*random(
    j.osc.freq(i.freqModulator.mult(scala
}
```

JAVASCRIPT

```php
<?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP

1.2 WHAT IS A PROGRAMMING LANGUAGE

```java
import com.cycling74.max.*;
public class SimpleBiquad extends MSPPerformer
{
    private float freq = 1000.0f;
    private float  c, a0, a1, a2, b1, b2, tin1, tin2,tout1, tout2;
    private float pi = 3.1415926f;
    private float r =  1.4142135f;
    private double sr; // sample rate
    private static final String[] INLET_ASSIST = new String[]{
        "input (sig)"
    };
    private static final String[] OUTLET_ASSIST = new String[]{
        "output (sig)"
    };
    public SimpleBiquad()
    {
        declareInlets(new int[]{SIGNAL, DataTypes.ALL});
        declareOutlets(new int[]{SIGNAL});
        setInletAssist(0, "signal input");
        setInletAssist(1, "Cut-Off Frequency");
        setOutletAssist(OUTLET_ASSIST);
        createInfoOutlet(false);    // suppress info outlet
    }
    public void inlet(float f)
    {

            freq = f;               // this is the cutOff frequency

    }
    public void dspsetup(MSPSignal[] ins, MSPSignal[] outs)
    {

        sr = ins[0].sr; //this is the sample rate

    }
```

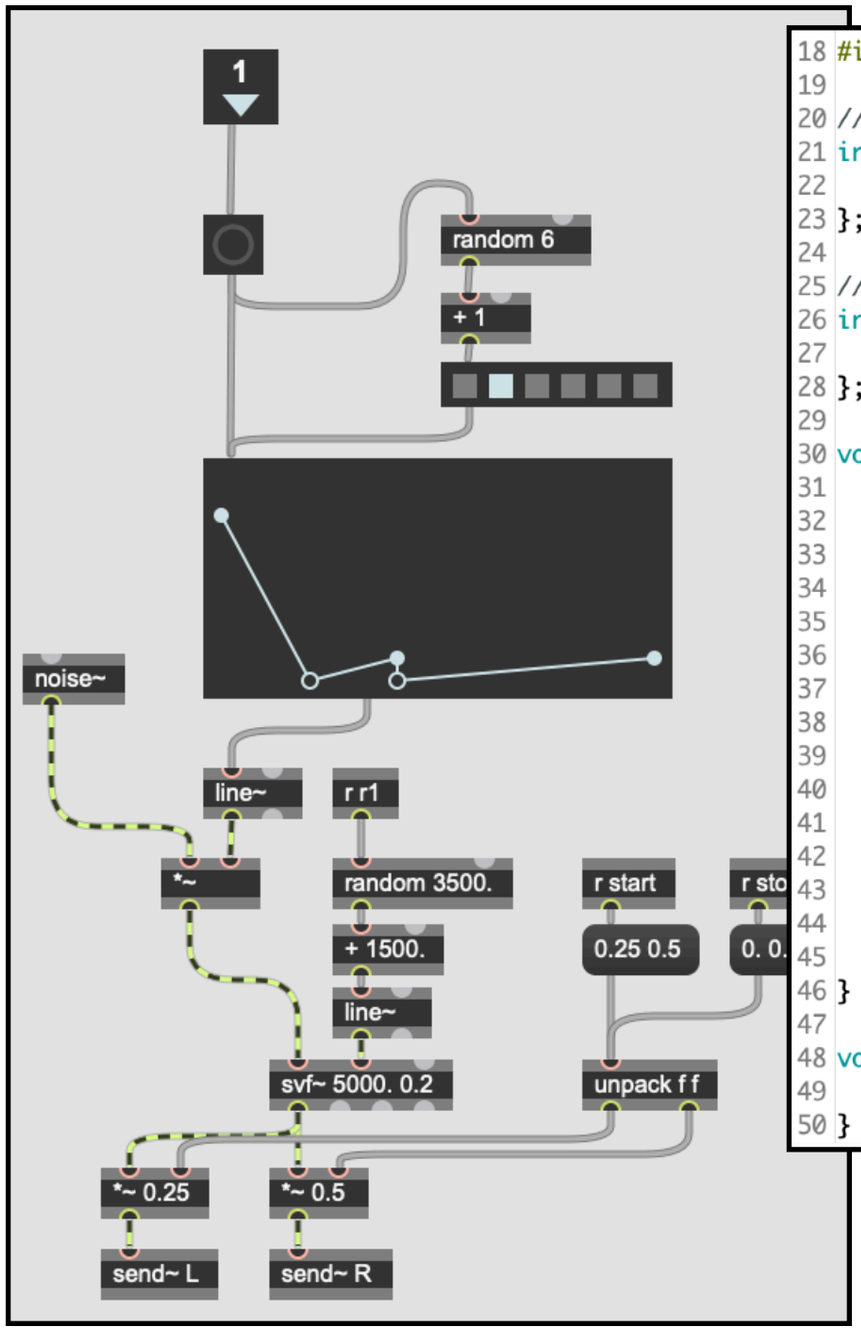
JAVA

C/C++

```c
/**
    @file
    simplemax - a max object shell
    jeremy bernstein - jeremy@bootsquad.com

    @ingroup    examples
*/

#include "ext.h"                            // standard Max include, always required
#include "ext_obex.h"                       // required for new style Max object

/////////////////////// object struct
typedef struct _simplemax
{
    t_object ob;            // the object itself (must be first)
} t_simplemax;

/////////////////////// function prototypes
//// standard set
void *simplemax_new(t_symbol *s, long argc, t_atom *argv);
void simplemax_free(t_simplemax *x);
void simplemax_assist(t_simplemax *x, void *b, long m, char *s);

/////////////////////// global class pointer variable
void *simplemax_class;

```

some examples of complete and powerfull standard programming languages

1.2 WHAT IS A PROGRAMMING LANGUAGE



MAX/MSP

ARDUINO

```
#include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note du
    //e.g. quarter note = 1000
    int noteDuration = 1000 / n
    tone(8, melody[thisNote], n

    // to distinguish the notes
    // the note's duration + 30
    int pauseBetweenNotes = not
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melo
}
```

some examples of programming languages made for artists / designer

PROCESSING

NOC_1_8_motion101_acceleration | Mover

```
class Mover {

  PVector location;
  PVector velocity;
  PVector acceleration;
  float topspeed;

  Mover() {
    location = new PVector(width/2, height/2);
    velocity = new PVector(0, 0);
    acceleration = new PVector(-0.001, 0.01);
    topspeed = 10;
  }
```

1.2 WHAT IS A PROGRAMMING LANGUAGE

Usually a **developer** writes the source code in a **text editor** or **IDE** (Integrated Development Environment) and creates a source file (*.html*, *.js*, *.css*, *.java*, etc.).

Even if a simple text editor can be used, there are **dedicated editors**, which contain many support features: Atom, Visual Studio Code, Notepad++, Sublime Text, etc.

Then the **source code** is *compiled* or *interpreted* to be transformed into a code understandable and executable by the machine itself (the *machine code*).

1.2 WHAT IS A PROGRAMMING LANGUAGE

hs2021.js

```
/** ////////////////////////////////////////////////////////////////////////
  * GLOBAL CONSTANT AND VARIABLES
    * definition
    * initialization
* //////////////////////////////////////////////////////////////////////// */

const MIN_FREQUENCY_HZ = 5; // min frequency in the x axis
const MAX_FREQUENCY_HZ = 3000; // max frequency in the x axis
const MIN_AMPLITUDE = 0.01; // min amplitude in the y axis
const MAX_AMPLITUDE = 0.5; // max amplitude in the y axis (never higher that 0.5)
const CANVAS_OFFSET = 100; // offset for the canvas
const TITLE_OFFSET = 75; // sum of text + padding of title and paragraph
const MASTER_GAIN = 0.01; // gain adaptation
const MODULATION_INDEX = 1000; // index of modulation for the FM
const MIN_AM_FREQ = 0.05; // min freq for the AM oscillator
const MAX_AM_FREQ = 0.5; // mas freq for the AM oscillator (never higher that 0.5)
const ATTACK_TIME = 1.6; // attack time for sounds
const RELEASE_TIME = 0.3; // release time for sounds
const MIN_DIAMETER = 10; // release time for sounds
const MAX_DIAMETER = 100; // release time for sounds

let audioContext; // variable that will store the audio context
let masterGain; // a Gain Node for the MASTER volume
let arrayOfDots = []; // array of Dots (oscillators)
let modulator = null; // variable that will contain the modulator oscillator (initializec
let carrier = null; // variable that will contain the carrier oscillator (initialized to


/** ////////////////////////////////////////////////////////////////////////
  * SUPPORT FUNCTIONS
    * definition
* //////////////////////////////////////////////////////////////////////// */

function startAudio(){ // audio to start when the first dot is created
   // to be implemented
}

function deleteModulatorCarrier(){
  // to be implemented
}

function deleteDot(currentDot){ // t                                    from th
    // to be implemented
}

function applyFM(modulator, carrier)
    // to be implemented
}

/** ////////////////////////////////////////////////////////////////////////
  * CLASS DOT
    * a Dot is an oscillator
* //////////////////////////////////////////////////////////////////////// */

class Dot{
   // to be implemented
}
```

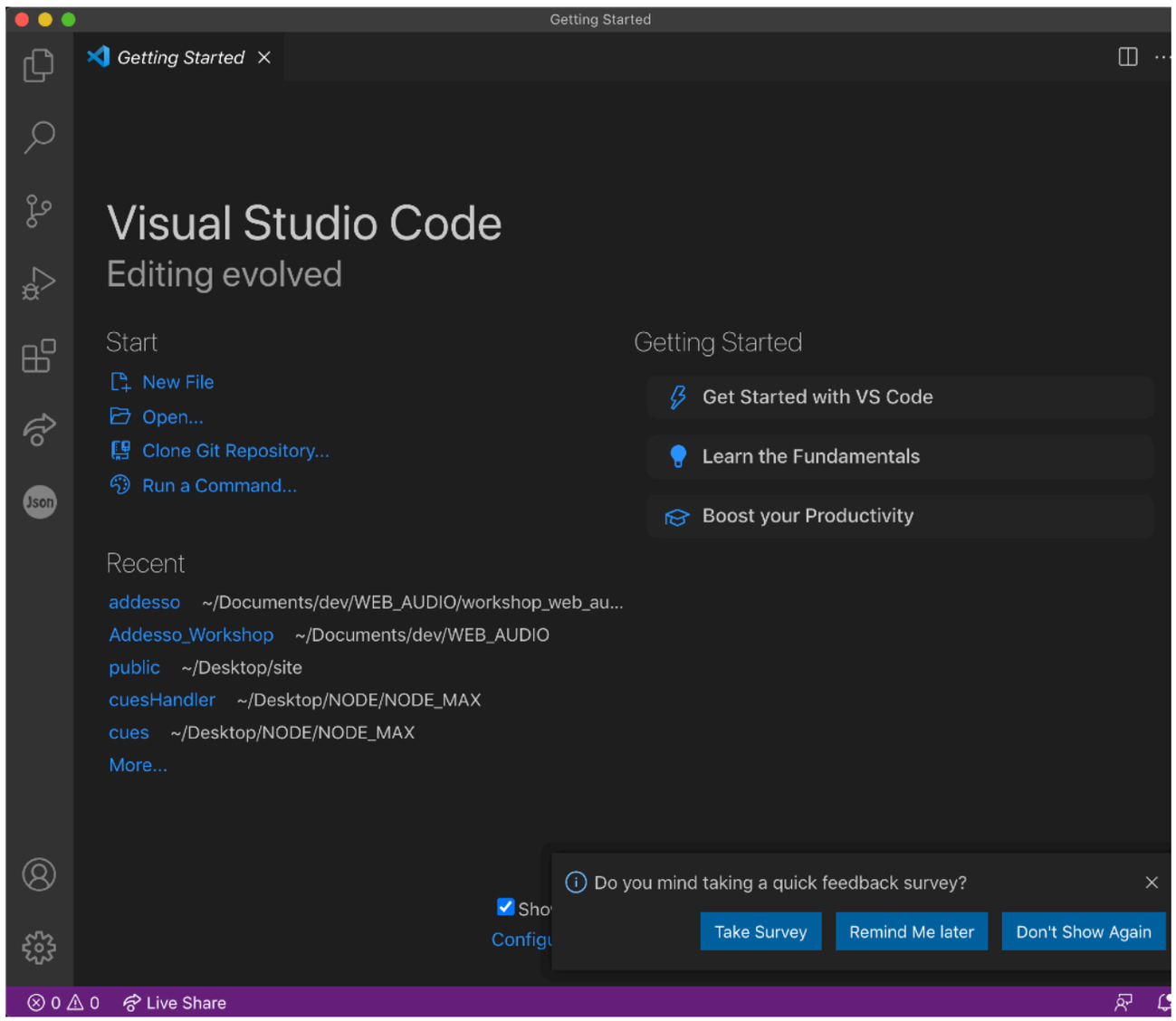
a javascript code written in a simple text editor

```
1  /** ////////////////////////////////////////////////////////////////////////
2    * GLOBAL CONSTANT AND VARIABLES
3      * definition
4      * initialization
5  * //////////////////////////////////////////////////////////////////////// */
6
7  const MIN_FREQUENCY_HZ = 5; // min frequency in the x axis
8  const MAX_FREQUENCY_HZ = 3000; // max frequency in the x axis
9  const MIN_AMPLITUDE = 0.01; // min amplitude in the y axis
10 const MAX_AMPLITUDE = 0.5; // max amplitude in the y axis (never higher that 0.5)
11 const CANVAS_OFFSET = 100; // offset for the canvas
12 const TITLE_OFFSET = 75; // sum of text + padding of title and paragraph
13 const MASTER_GAIN = 0.01; // gain adaptation
14 const MODULATION_INDEX = 1000; // index of modulation for the FM
15 const MIN_AM_FREQ = 0.05; // min freq for the AM oscillator
16 const MAX_AM_FREQ = 0.5; // mas freq for the AM oscillator (never higher that 0.5)
17 const ATTACK_TIME = 1.6; // attack time for sounds
18 const RELEASE_TIME = 0.3; // release time for sounds
19 const MIN_DIAMETER = 10; // release time for sounds
20 const MAX_DIAMETER = 100; // release time for sounds
21
22 let audioContext; // variable that will store the a
23 let masterGain; // a Gain Node for the MASTER volu
24 let arrayOfDots = []; // array of Dots (oscillators
25 let modulator = null; // variable that will contain                    l)
26 let carrier = null; // variable that will contain t
27
28
29 /** ////////////////////////////////////////////////
30   * SUPPORT FUNCTIONS
31     * definition
32 * //////////////////////////////////////////////////////////////////////// */
33
34 function startAudio(){ // audio to start when the first dot is created
35   // to be implemented
36 }
37
```

a javascript code written in a dedicated programming text editor

1.2 WHAT IS A PROGRAMMING LANGUAGE

https://visualstudio.microsoft.com/



If you don't have a text editor you can download and install **Visual Studio Code** (it's free).

For the workshop is better to use **Google Chrome** (better for web audio API compatibility).

1.3 WHAT IS THE WEB PROGRAMMING

The term **web programming** refers to all programming activities, languages, practices, and technologies that enable the creation of **web applications**.

A **web application** is a **dynamic website** running on a browser where is required a **high user interaction** (opposed to static web pages).

> Music has always been a highly **interactive activity**. Musicians interact with their instruments, with other musicians, with dancers or with the audience, so for this perspective, the very idea of 'interactive music' a term often employed for referring to the music resulting from the dialogue between a musician and a computer, seems self-evident.
>
> ELECTRONIC MUSIC, N. Collins, J. d'Escrivan, The Cambridge Companion

1.3 WHAT IS THE WEB PROGRAMMING

A **web browser** is a software for accessing the World Wide Web, an information system where the resources are identified by an URL - Uniform Resource Locator, accessible over the internet.

Internet is the global system of interconnected computer networks that uses the protocol TCP/IP to communicate.

1.3 WHAT IS THE WEB PROGRAMMING

Web development is divided into several areas:

- **Web design**;
- **GUI** (graphic user interface) **development**;
- **Client-side** programming and scripting;
- **Server-side** development;
- **Database** management;
- **Network security**;
- ...

1.3 WHAT IS THE WEB PROGRAMMING

The **coding** involved in **Web development** includes different languages:

- **markup languages** (HTML, CSS): define the structure, the organization, and the look and feel of a site.

- **client-side scripting languages** (javascript): transform the website from a static page to an interactive application; client-side code runs in the browser.

- **server-side scripting languages** (PHP / node.js): server-side code lives on a server, and serves as go-between architecture, transferring data to the browser, etc..

Do we need all of this for making music on the browser?

1.3 WHAT IS THE WEB PROGRAMMING

a programmer who deals with all aspects of the web development is called a **full stack developer**

Do we need all of this for **making music on the browser** ?

1.3 WHAT IS THE WEB PROGRAMMING

NO!

Do we need all of this for **making music on the browser** ?

1.3 WHAT IS THE WEB PROGRAMMING

We just need some lines of **client-side code** (html + css + javascript) and a **web browser** (like Google Chrome).

We don't need to manage a database, to organize a large amount of data, or to store user data, to implement security protocols, etc..

Working locally on our machine, **we won't even need a server** where we can save (and request) our client-side code because the browser can load, directly from our file system, the *entry point* of the client-side code: **a simple HTML file with some CSS and Javascript**.

1.4 WHAT IS AN API

In the context of computer programming, an **Application Programming Interface** (API) represents a **set of procedures** for carrying out given tasks.

An API lists a **bunch of operations** that developers can use, along with a description of what they do (the **API documentation**).

The developer doesn't necessarily need to know how they work internally, they just **need to know how to use them**.

The term can also indicate a **software library** of a programming language.

1.4 WHAT IS AN API

An API **reduces the amount of code developers need to create**, and also helps create **more consistency apps** for the same platform.

In last years, the specification and implementation of new APIs in web browsers has allowed for

envisioning the web platform as a fertile play-ground for artists and musicians / composers.

1.5 WHAT IS THE WEB AUDIO API

The **web audio API** is a **high level javascript API** developed by W3C **used for processing and synthesizing audio in web applications**.

The **web audio API** provides a powerful system for:

- controlling audio on the Web;
- create audio sources;
- add effects to audio;
- create audio visualization;
- compose with algorithmic procedures;
- apply spatial effects;
- be creative!

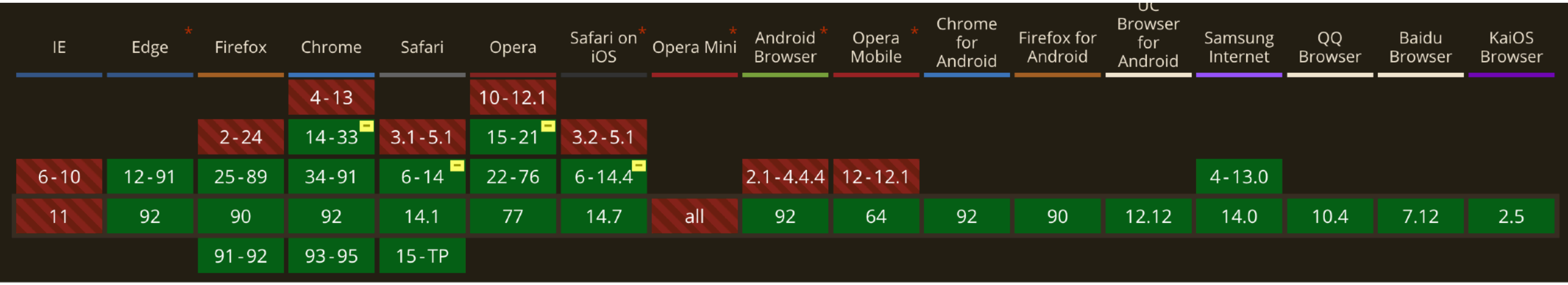
1.5 WHAT IS THE WEB AUDIO API

Web audio API documentation can be found here:

OFFICIAL DOCUMENTATION:
https://www.w3.org/TR/webaudio/

MOZILLA DOCUMENTATION:
https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

1.6 WEB AUDIO API BROWSER SUPPORT



https://caniuse.com/?search=web%20audio

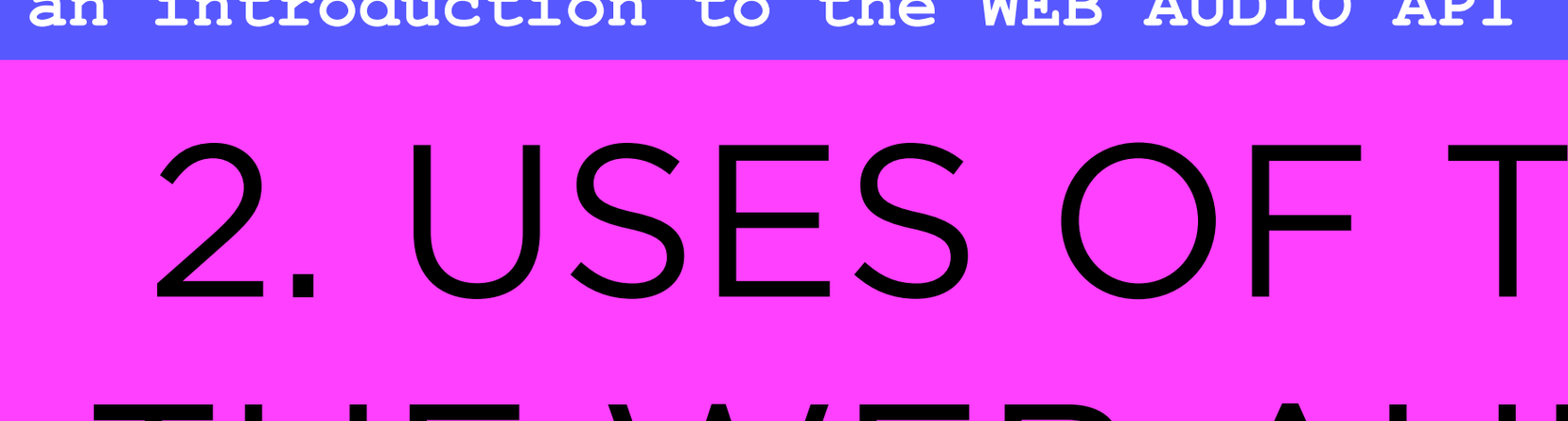
The **web audio API** is supported in most of the modern browsers (desktop or mobile).

1.3 WHAT IS THE WEB PROGRAMMING

Why should we be interested in **creating music on the web as composers** ??

2. USES OF THE THE WEB AUDIO

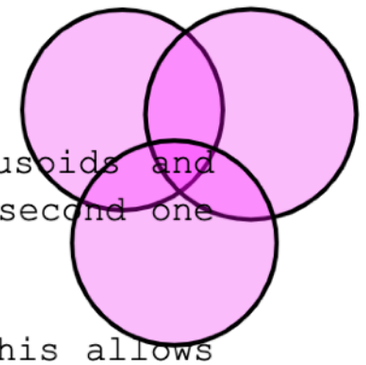
Dot **Drone Generator** is a drone generator which allows you to create **synthetic textures** directly on your browser.



Click on the window to generate a **Dot**, a sinusoidal wave with tremolo.

The **y-axis** represents the amplitude range. The **amplitude** is modulated by a **triangular LFO** (Low Frequency Oscillator), with random frequency.

The **x-axis** represents the frequency range.



Press 'L' and then Click+Drag from an existing Dot to an other one, to link two sinusoids and create a **Frequency Modulation** between them. The first Dot becomes the modulator of the second one (the carrier).

It is possible to create a **chain of modulation**: each carrier can become a modulator. This allows you to create complex spectra, to the point of creating very noisy sounds!

Click on an existing circle to **delete** it or to delete the modulation chain of which it is part.

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

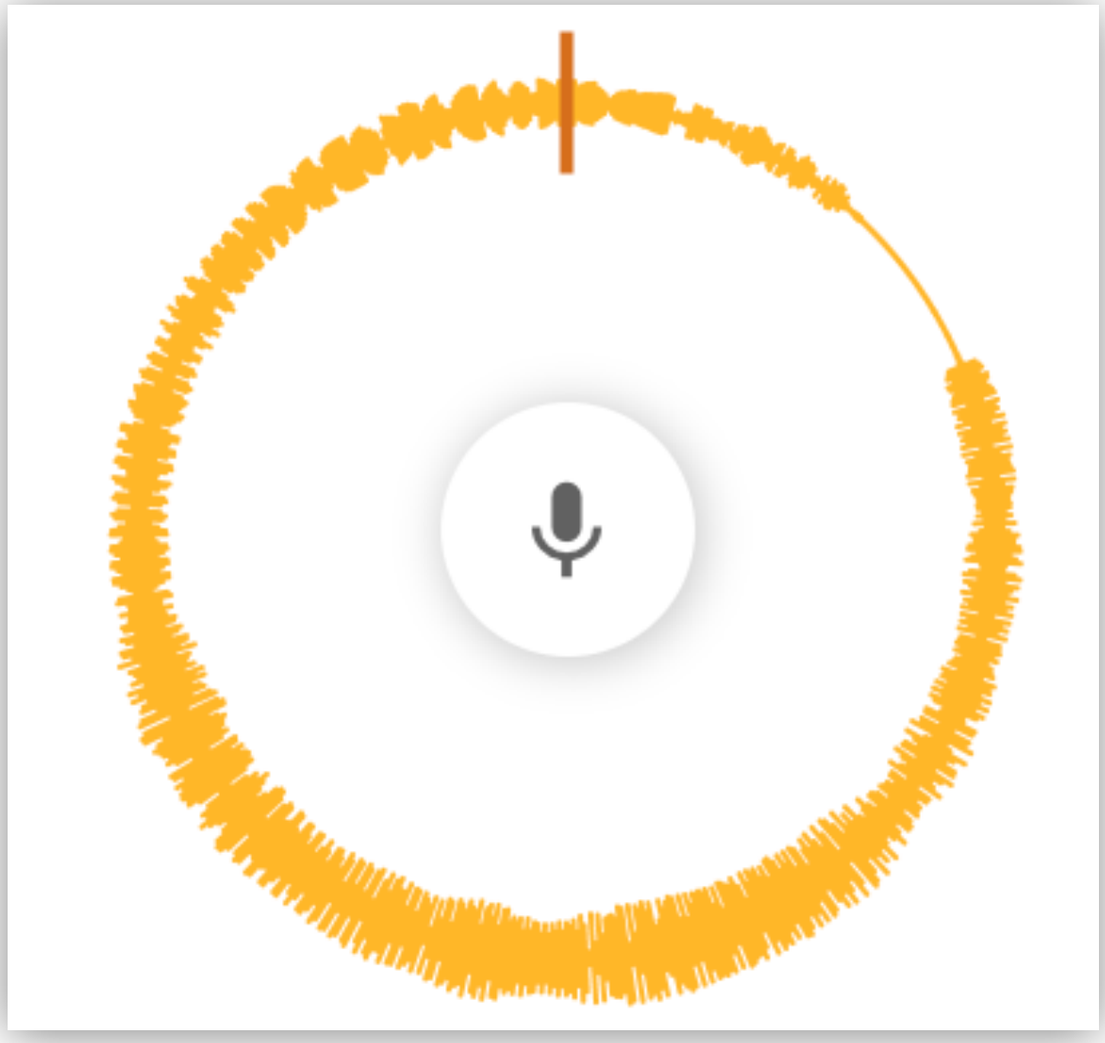
1. **play audio tracks** in the browser with a control over panning and volumes, and process them using DSP algorithms;

2. performing **sound analysis** in the time domain and frequency domain (waveform and spectrum);

3. develop **sound visualization applications**; integrate the web audio API with web graphics libraries (webGL, three.js);

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

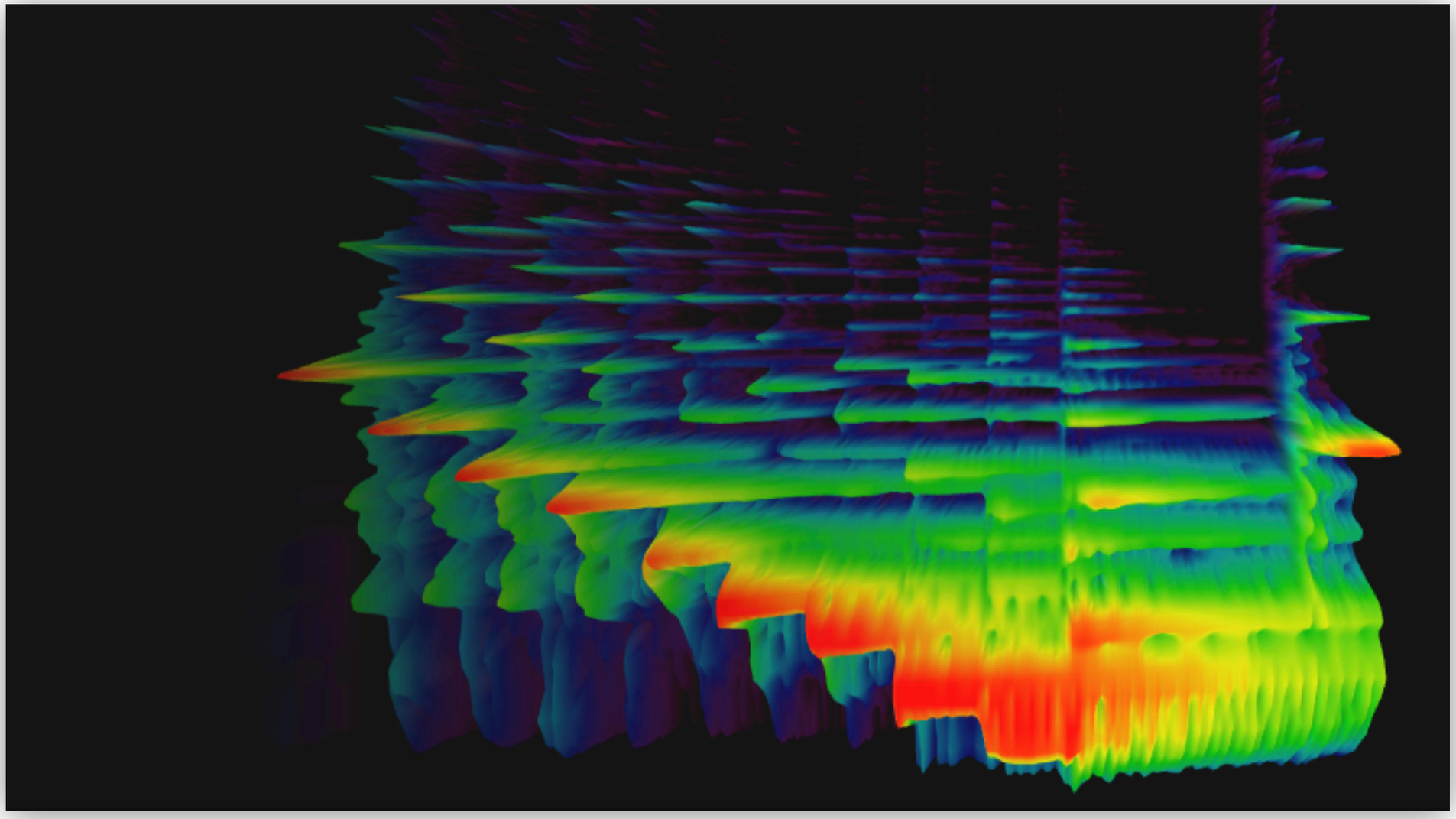
example 1 | VOICE - SPINNER



https://musiclab.chromeexperiments.com/Voice-Spinner/

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 2 | AUDIO ANALYZER



https://musiclab.chromeexperiments.com/Spectrogram/

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

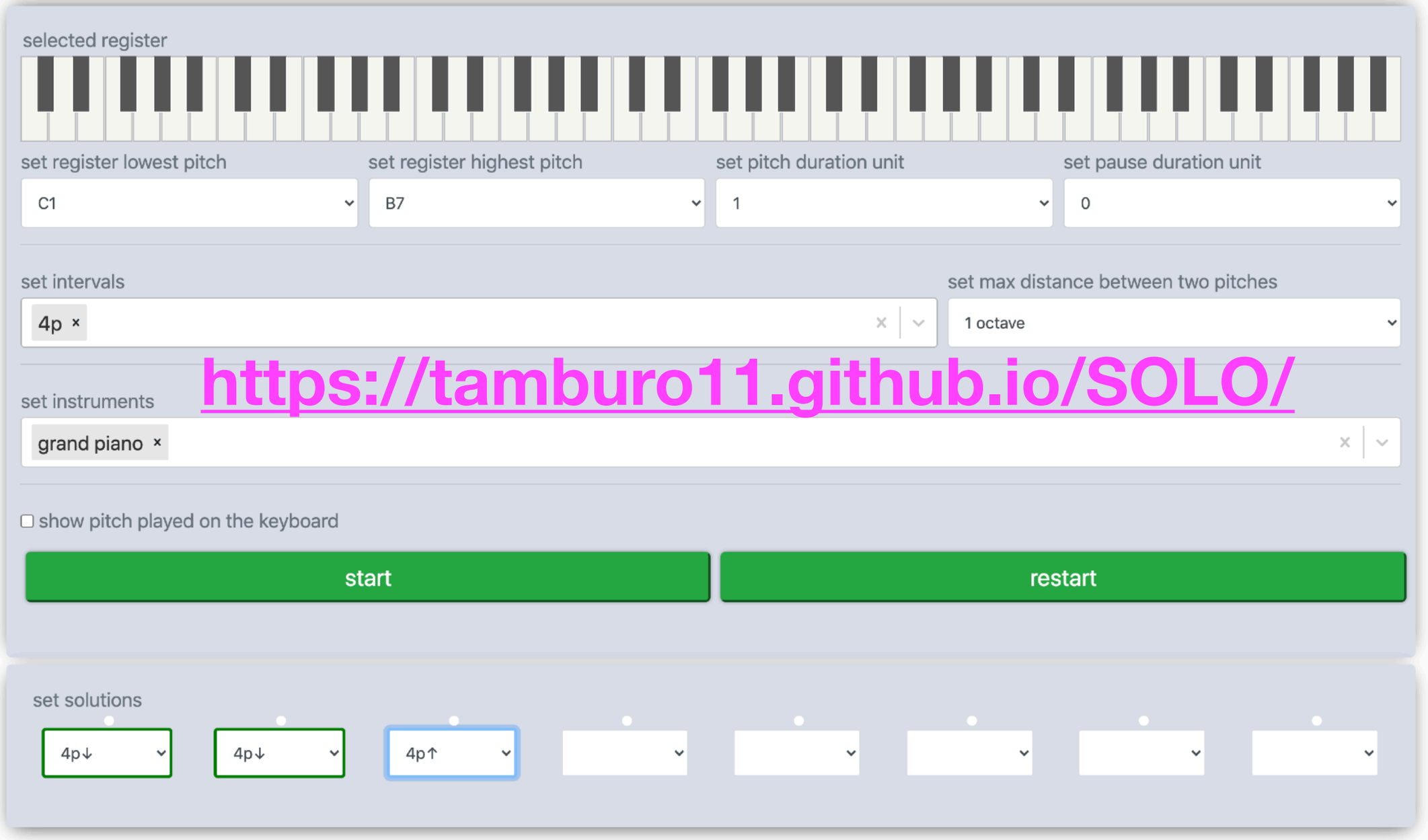
4. creating **web applications for music education** (music theory, ear-training, etc.);

5. creating **generative compositions** living in the browser (exploiting the algorithmic logic offered by the object oriented programming language javascript and the whole ecosystem of its libraries and extensions);

6. create **interactive compositions**, in which the users play an important role in the creative process;

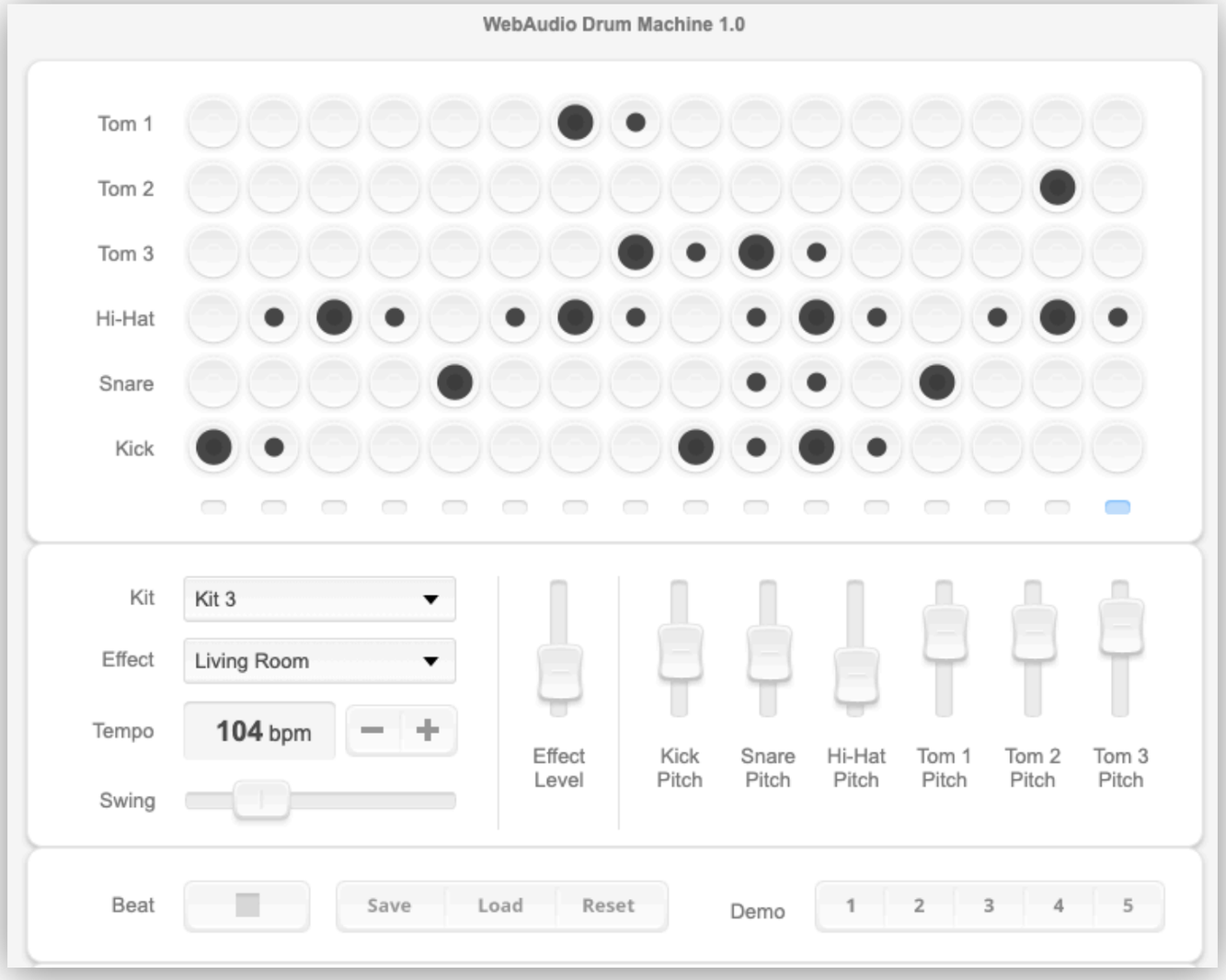
2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 3 | MUSIC EDUCATION



2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 4 | DRUM MACHINE



https://webaudiodemos.appspot.com/MIDIDrums/index.html

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

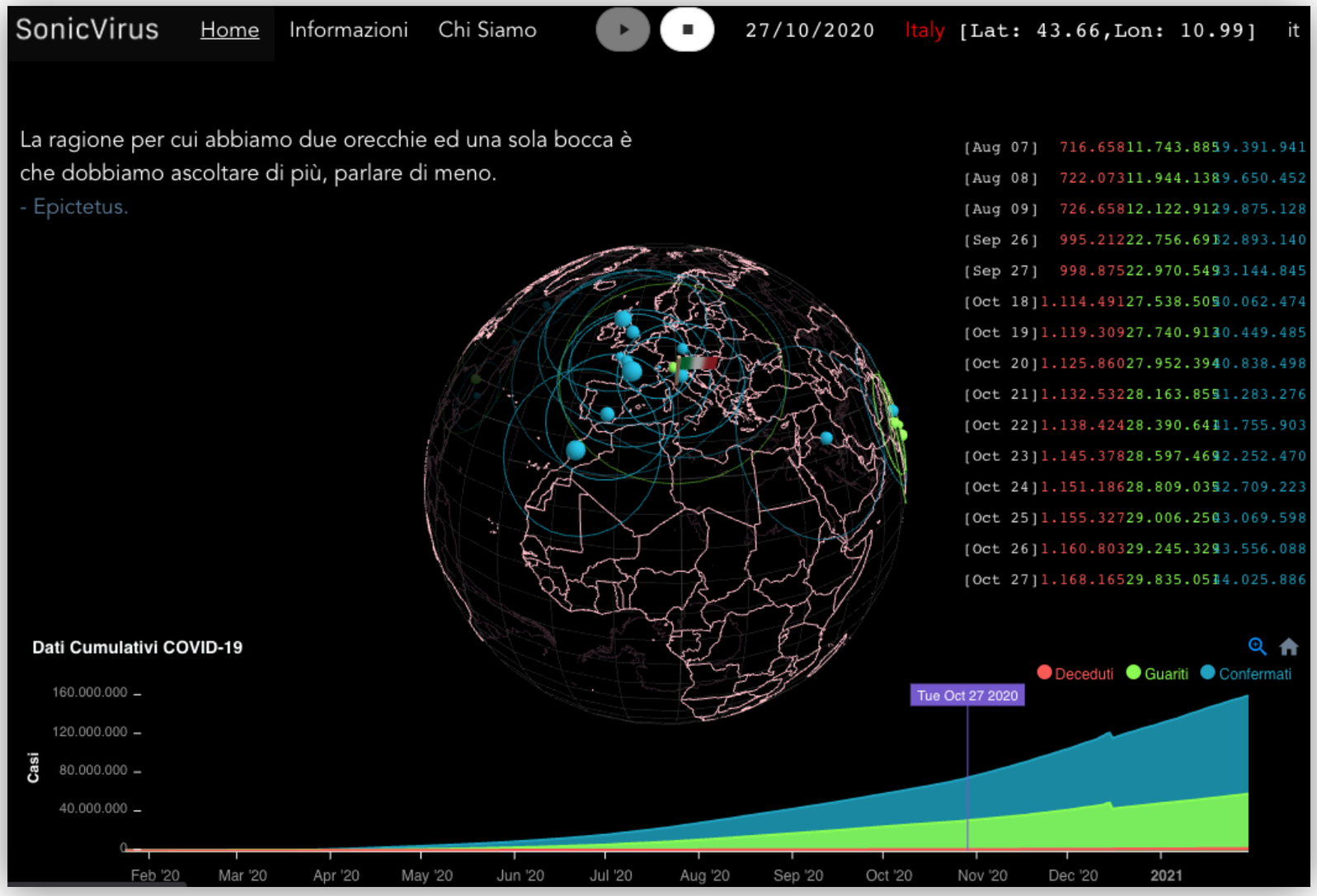
7. real-time **data analysis and data sonification**; exploiting data analysis for sound synthesis and composition, taking advantage of the many APIs to collect and receive data in real time;

8. create **not-fixed media music**; creating compositions that can be accessed anywhere and from different devices;

9. develop **synths and web musical instruments**;

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 5 | DATA SONIFICATION



https://sonicvirus.si.usi.ch/

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 6 | WEB AUDIO SYNTHESIZER

https://noisehack.com/scissor/

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 7 | WEB PEDAL BOARD

https://pedalboard.netlify.app/

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

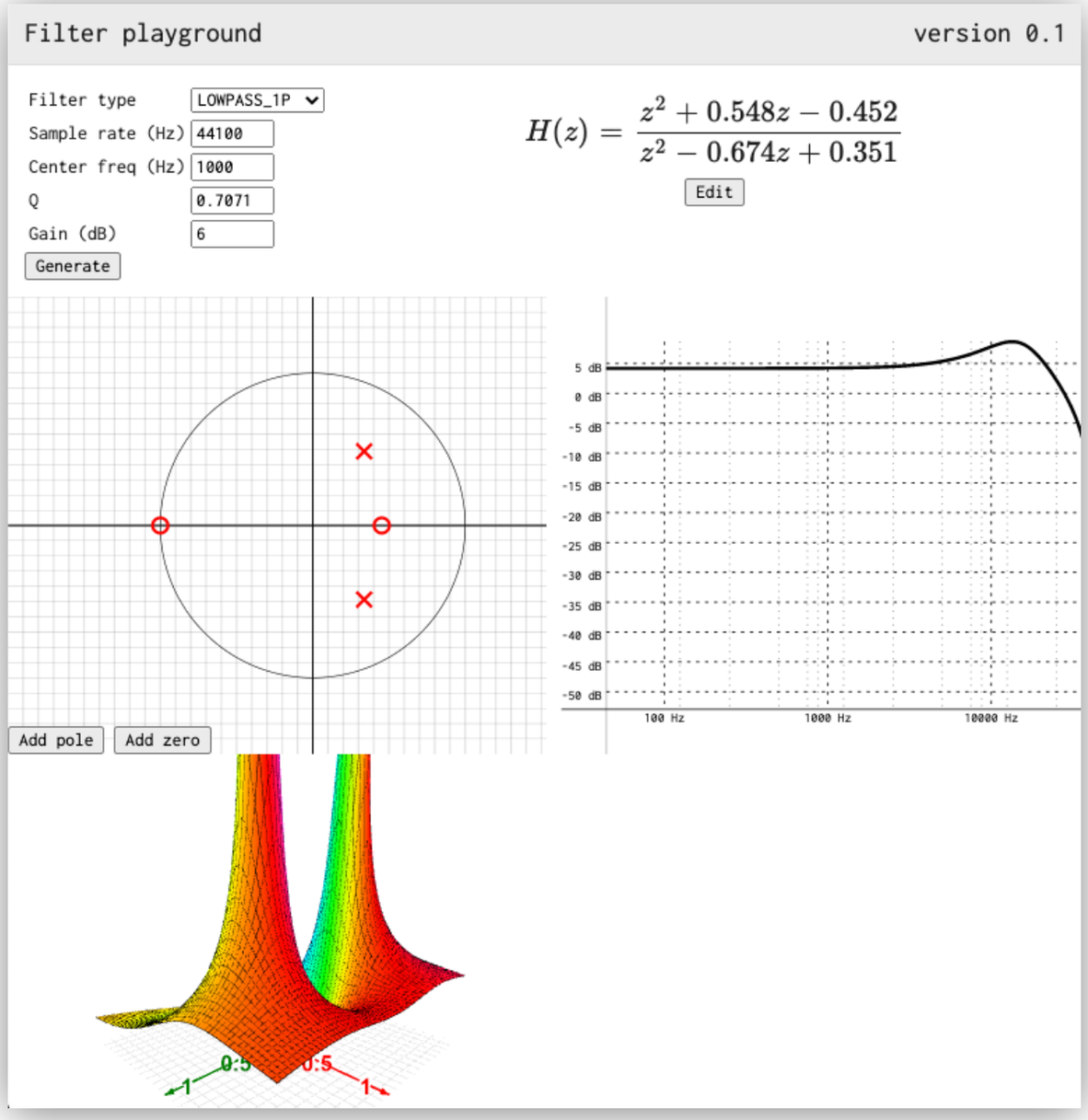
10. use the web audio along with **web MIDI** to create integrated browser / desktop software / MIDI interface environments;

11. develop **web audio effects for real-time** DSP; with the AudioWorklet it is possible to create high-performance DSP algorithms that work at the sample level (there are some free ones, such as *genish.js*);

12. create **environments for live-coding** and **online music performances** that take advantage of the web technologies and web support;

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 8 | FILTER DESIGN



https://borismus.github.io/filter-playground/?equation=%28z%5E2%2B0.5475285649299622z-0.45247143507003784%29%2F%28z%5E2-0.6741842031478882z%2B0.35059242778065 16%29

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 9 | LIVE CODING ENVIRONMENT

https://mercury-sketch.glitch.me/

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

13. exploit the possibilities of **integration with the entire web ecosystem** (extensions and libraries) to do almost anything!

14. exploit the web audio to create **web IDEs** that interpret other audio languages (e.g. Csound: https://ide.csound.com/);

15. develop **web audio effects for real-time** DSP; with the AudioWorklet it is possible to create high-performance DSP algorithms that work at the sample level (there are some free ones, such as *genish.js*);

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

16. create **environments for live-coding** and **online music performances** that take advantage of the web technologies and web support;

17. exploit the possibilities of **integration with the entire web ecosystem** (extensions and libraries) to do almost anything!

18. exploit the web audio to create **web IDEs** that interpret other audio languages (e.g. Csound: https://ide.csound.com/);

2.1 (NON-EXHAUSTIVE) LIST OF WEB AUDIO USES

example 10 | GENERATIVE MUSIC

https://tamburo11.github.io/thetricliniumdesireWeb/