WEB AUDIO IN ACTION

A Patchable Synthesizer

Who am I

successful tech-lead

bedroom producer

ekozhura at github

kojuro.com

@brutallosapiens at t.me

Web Audio: First Encounter

Tero Parviainen "Reactive Music Apps in Angular and RxJS" Dec 16 @fwdays.

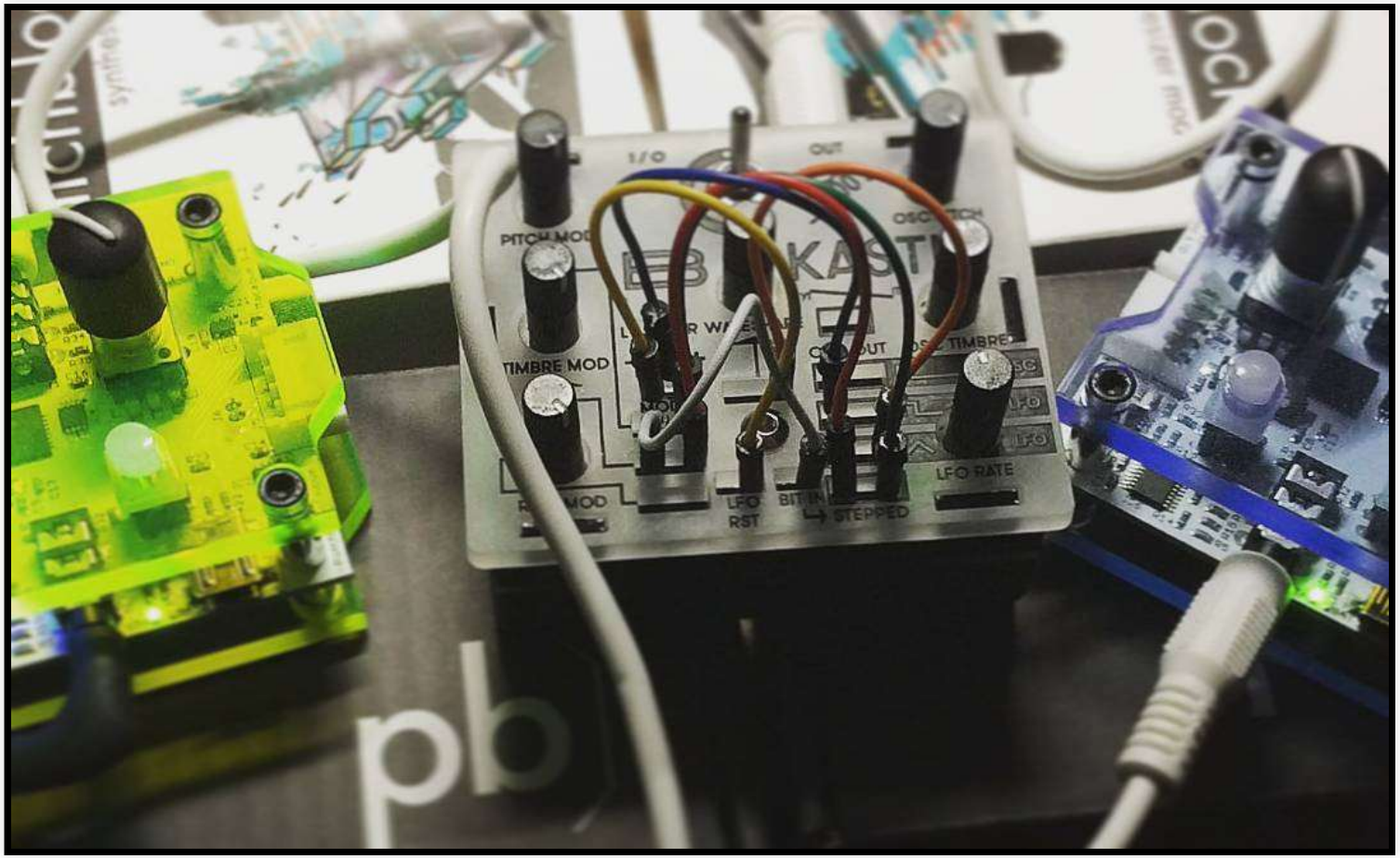
https://youtu.be/EB-CreYq1WY

Web Audio: My path

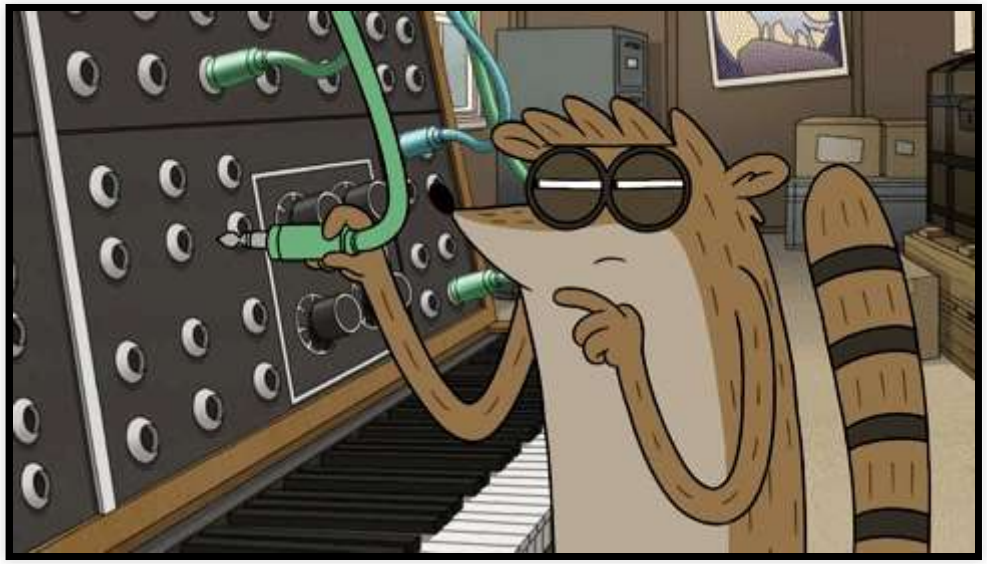
- Fun with bitwise noise (and Web Audio)
- Making beats with Web Audio (and RxJS)

  in progress: a Patchable Synthesizer

A Patchable Synthesizer



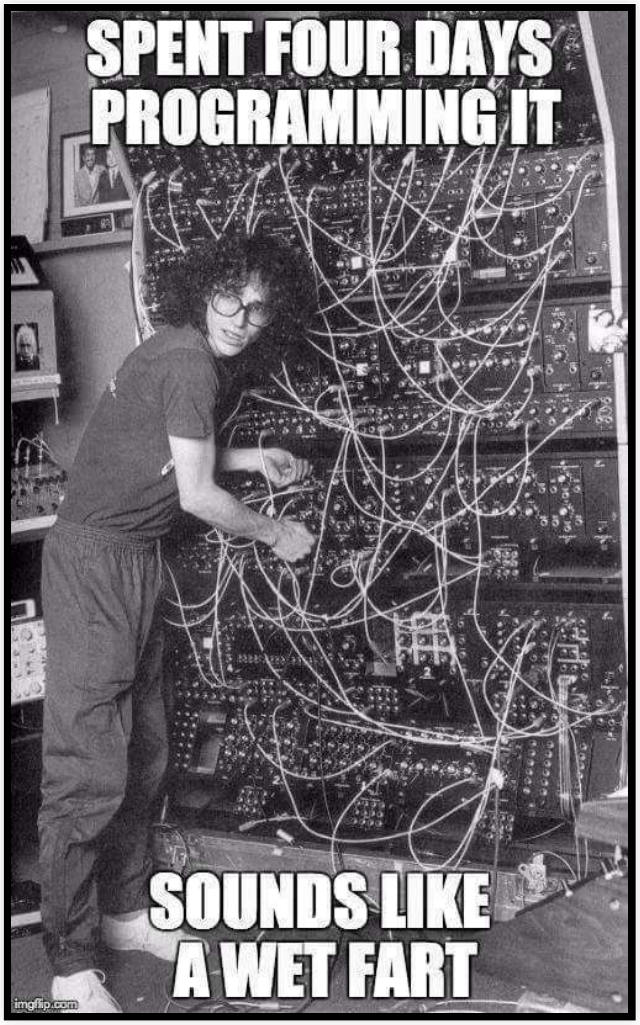
Patching



It is a creative process - each patch can represent a unique timbre or sequence.

And a piece of music in the end...

or just a succession of failures



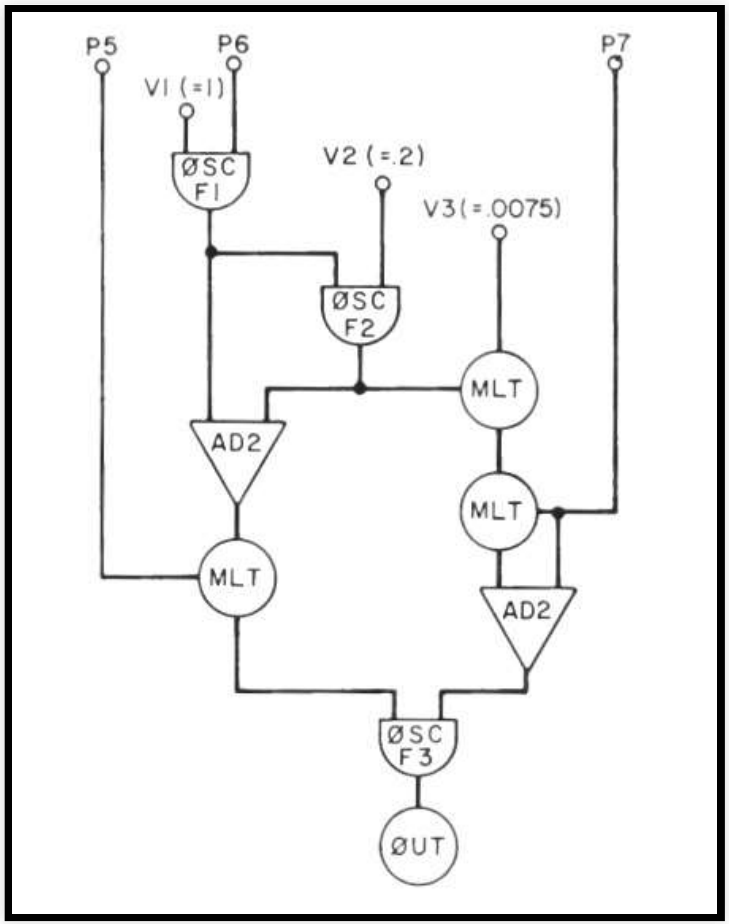
If we consider Web Audio as a modular synthesizer, programming becomes patching.

This approach to audio programming is from 50s and

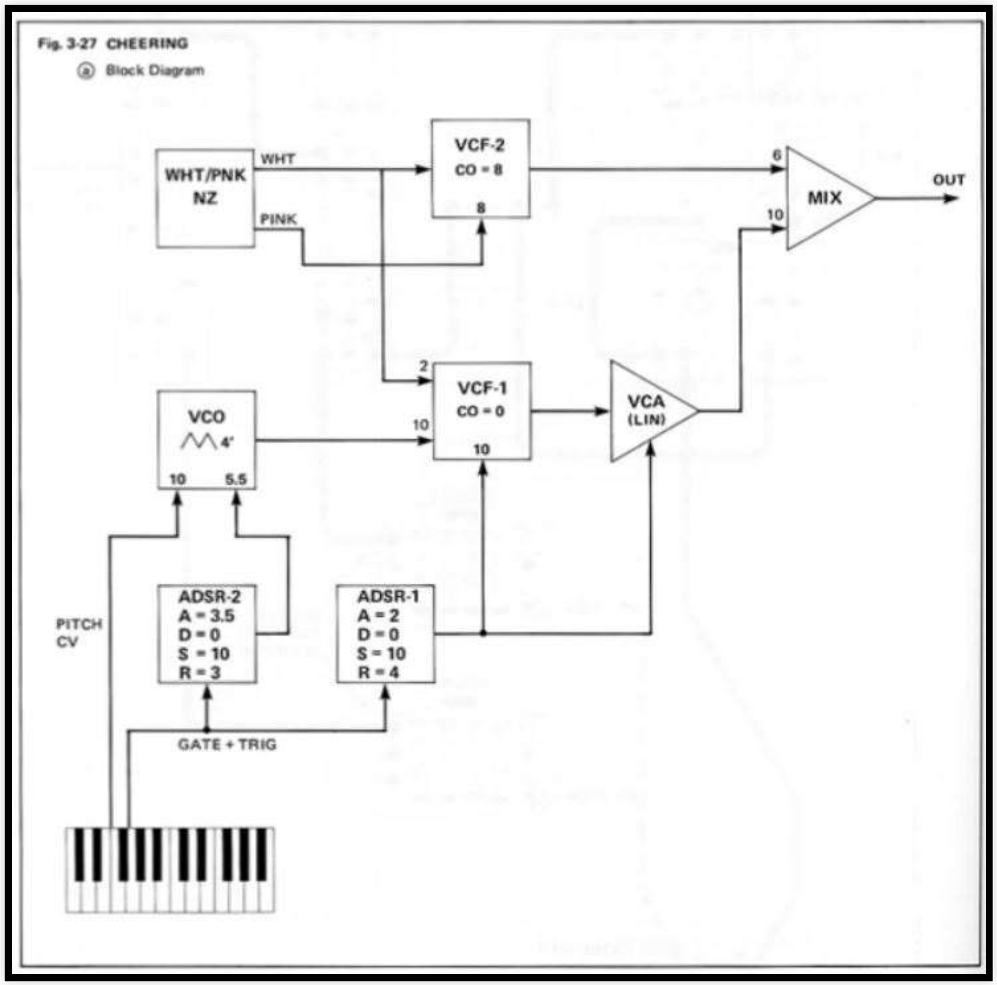
it finds its extension in a number of audioprogramming environments:

- CSound
- SuperCollider
- Max/MSP
- PureData
- ChucK

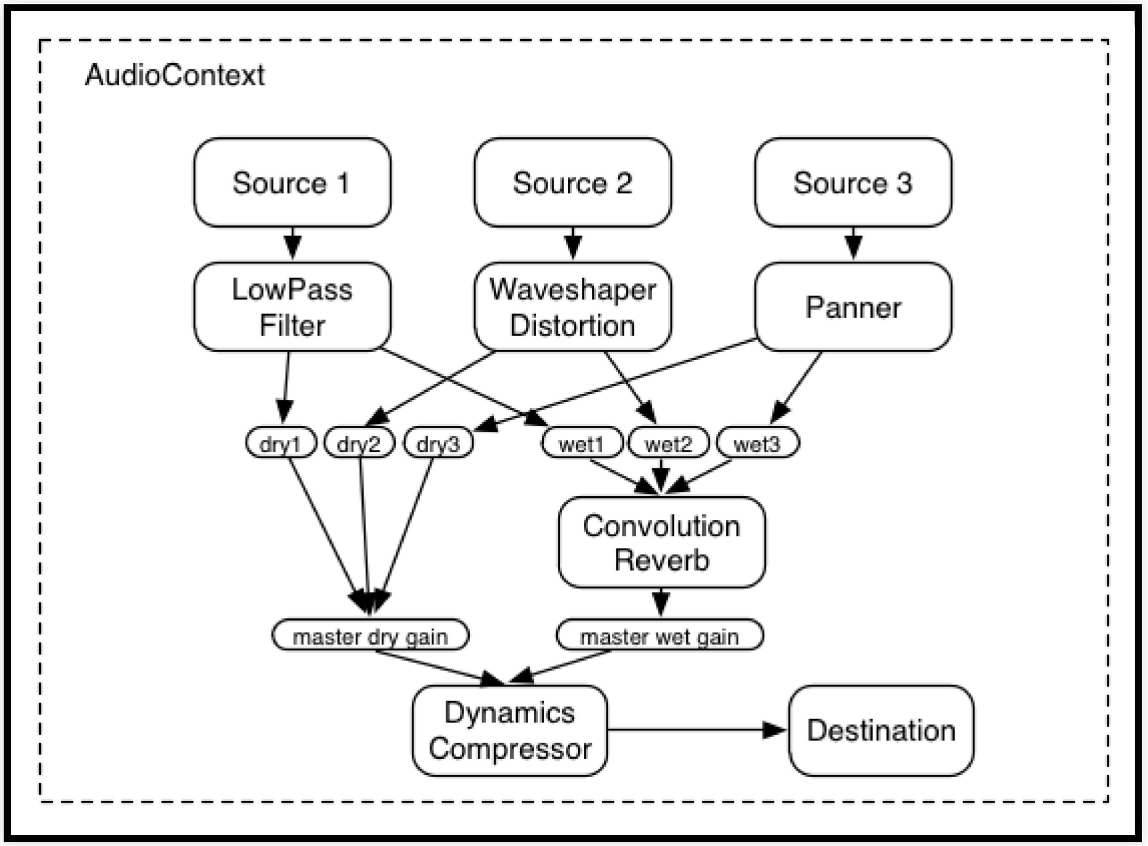
Typical flowchart of a program in MUSIC V



Typical diagram of an analog synth patch



Typical flowchart of a program in Web Audio



Why I am saying all of this?

We have a great source of knowledge, patch ideas and synths implementations from 50s to the present days

Audio Synthesis 101

Start with *AudioContext*

Generate with *AudioScheduledSourceNode*

Process with *AudioNode*

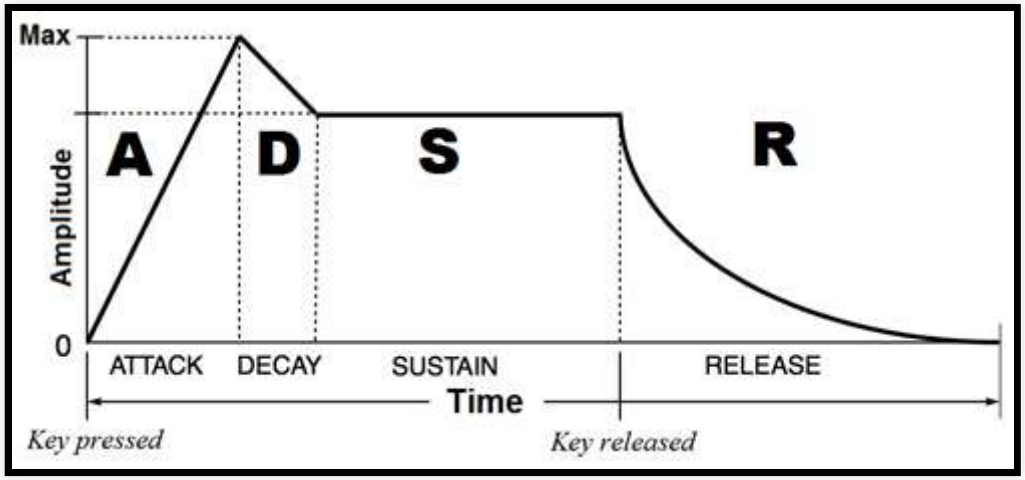
Control with *AudioParam*

Extend with *AudioWorklet*

Output with *AudioDestination*

Envelope Generator

What is an envelope?



Envelope Generator: naive implementation

```
class EnvGenerator {
  connect(param: AudioParam) {
    this.param = param;
  }
  trigger(min = 0, max = 1) {
    let now = this.context.currentTime;
    this.param.setValueAtTime(min, current);
    this.param.linearRampToValueAtTime(max, now + this.attack);
    this.param.linearRampToValueAtTime(this.sustain * max, now +
    this.param.linearRampToValueAtTime(min, now + this.attack + t
  }
}
```

Patch Ideas

Percussion patch

Bassline

Chiptune arp

PATCH 1: BASIC ”ANALOG” KICK DRUM

808 style: sin osc at a low freq and with an envelope with fast attack and decay

```
let atx = new AudioContext();
let osci = new OscillatorNode(atx);
let gainNode = new GainNode(atx);

let volumeEnvelope = new EnvGenerator(atx, 0.01, 0.1, 0.2, 0.35);
volumeEnvelope.connect(gainNode.gain);

osci.frequency.setValueAtTime(80, atx.currentTime);
osci.connect(gainNode).connect(atx.destination);

osci.start();
volumeEnvelope.trigger();
```

Make it punchy

```
let pitchEnvelope = new EnvGenerator(atx, 0.001, 0.001, 80);
pitchEnvelope.connect(osci.frequency);

pitchEnvelope.trigger(80, 10000);
```

PATCH 2: Bassline

basic patch:

```
let atx = new AudioContext();

let pulse1 = new OscillatorNode(atx);
let pulse2 = new OscillatorNode(atx);
pulse1.type = "square";
pulse2.type = "sawtooth";

let lpFilter = new BiquadFilterNode(atx);
lpFilter.type = "lowpass";
lpFilter.frequency.setValueAtTime(400, atx.currentTime);

pulse1.connect(gainNode);
pulse2.connect(gainNode);

gainNode.connect(lpFilter).connect(atx.destination);
```

add envelopes:

```
let pulseEnv = new EnvGenerator(atx, 0.3, 0.4, 1, 0.2);
let cutoffEnv = new EnvGenerator(atx, 0.3, 0.4, 800, 0.2);
pulseEnv.connect(gainNode.gain);
cutoffEnv.connect(lpFilter.frequency)
```

Poor man's chiptune

Square/pulse wave is enough, and the arpeggiator does the trick:

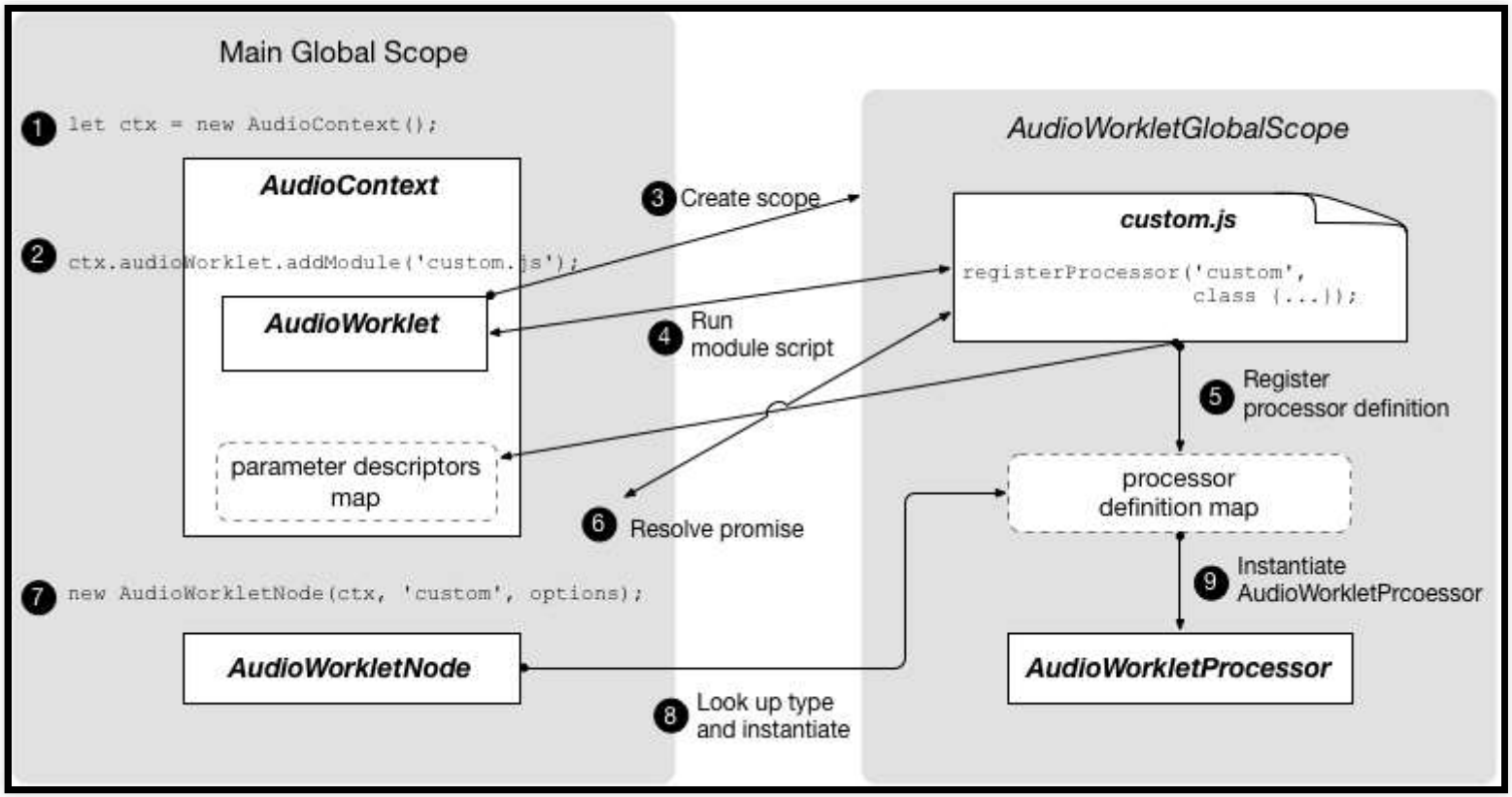
```
let atx = new AudioContext();
let osc = new OscillatorNode(atx);
osc.type = "square";

let arp = (param, interval) => ([first, second, third]) => {
    param.setValueAtTime(first, atx.currentTime + interval * 1);
    param.setValueAtTime(second, atx.currentTime + interval * 2);
    param.setValueAtTime(third, atx.currentTime + interval * 3);
  }
}

let freqs = [138.59, 155.56, 164.81, 207.65]; // [ C#, D#, E, G#]

let seq = arp(osc.frequency, 0.25);
seq([freqs[0], freqs[3], freqs[2]]);
```

Make arpeggiator run faster and pitch a little bit higher

so the sequence is perceived as a chord

```
let seq = arp(osc.frequency, 0.02);
seq([freqs[0] * 2, freqs[3] * 2, freqs[2] * 2]);
```

Bonus: AudioWorklets

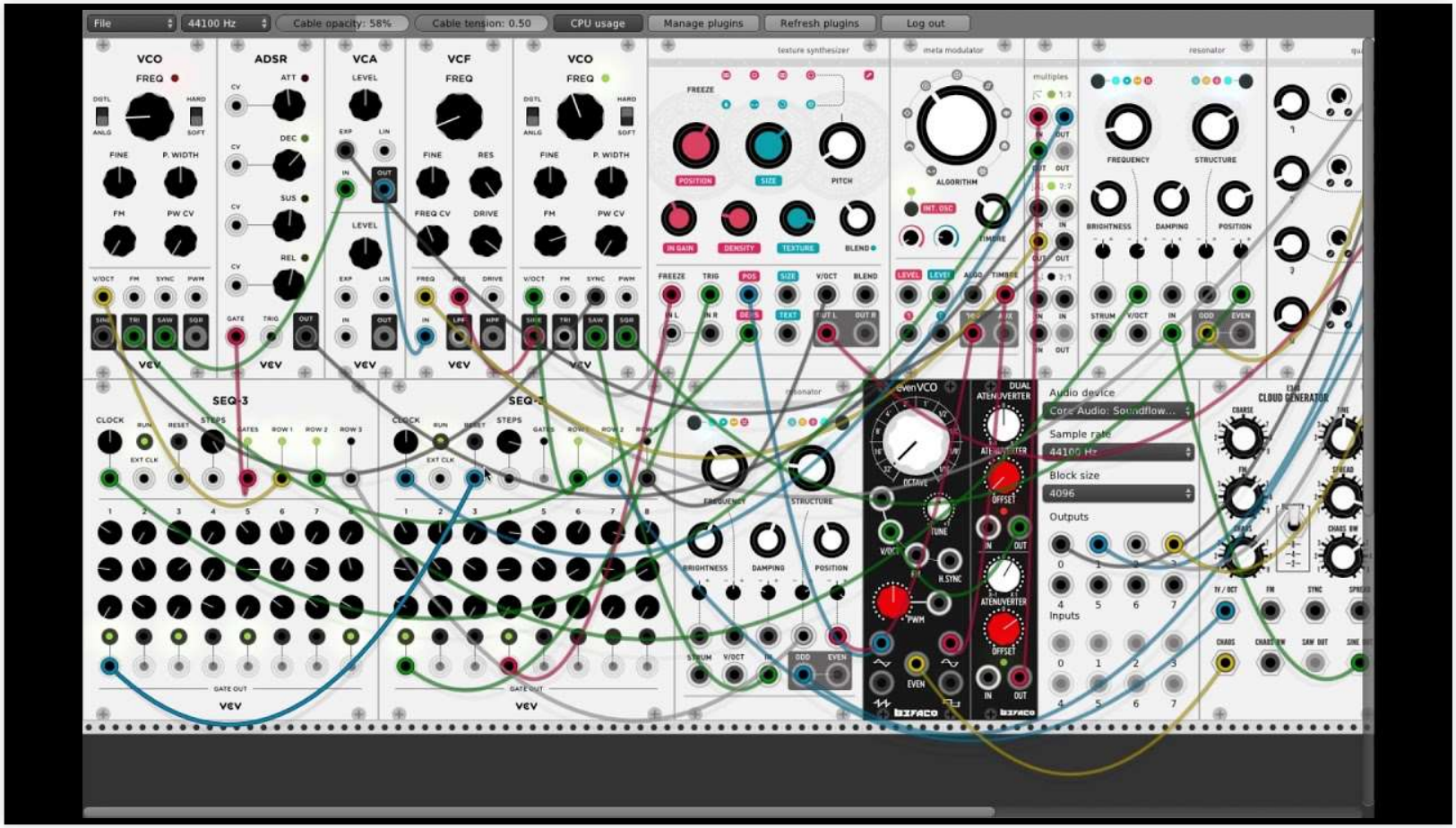


Your next step: Learn

- Sound on Sound articles: Synth Secrets
- Theory and Techniques of Electronic Music (PureData)
- Computer Music with examples in SuperCollider 3

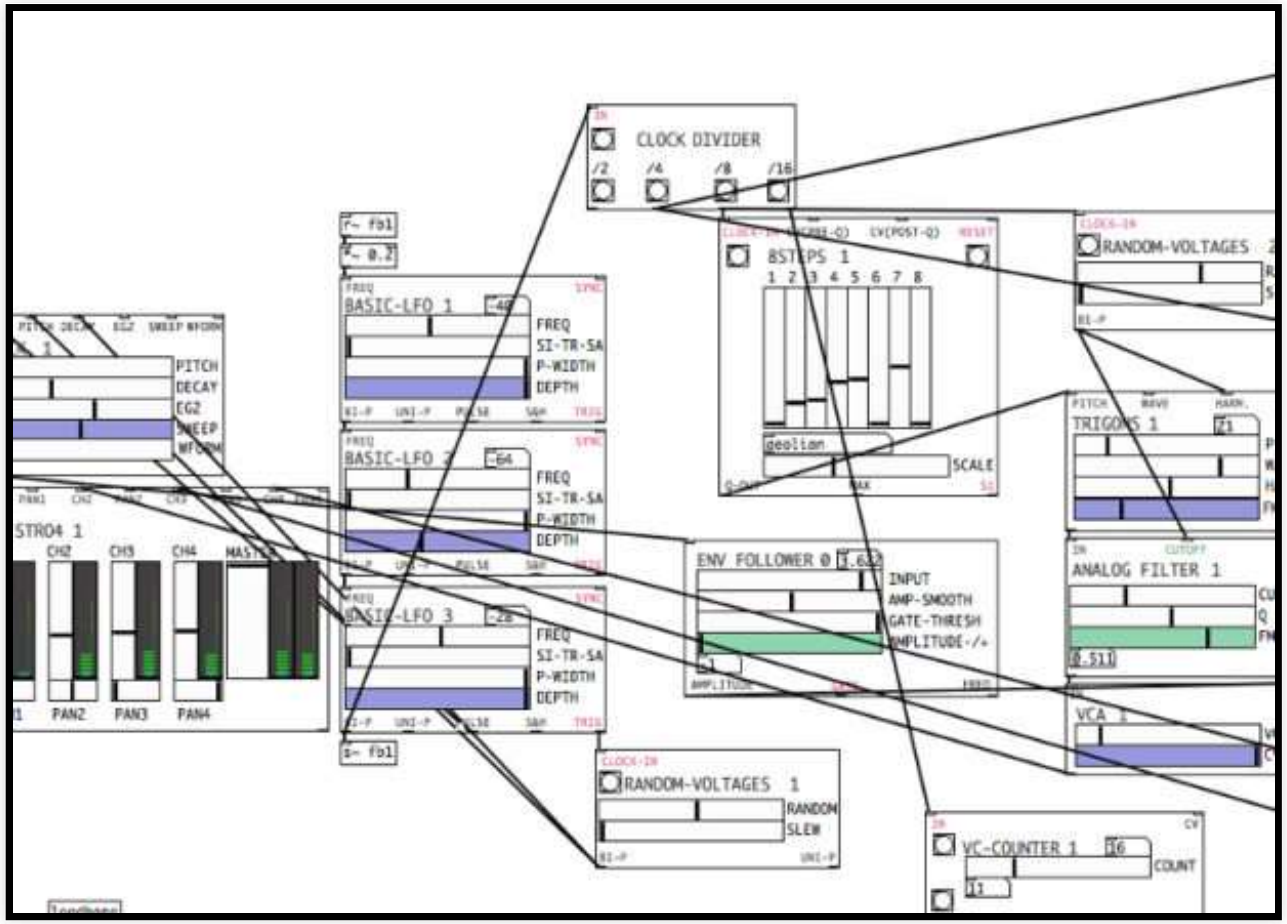
Your next step: Experiment

VCV RACK



Your next step: Experiment

AUTOMATONISM



Thank You!