



**HAL**  
open science

# Interoperability/Reusability of high level WebAudio components

Michel Buffa

► **To cite this version:**

Michel Buffa. Interoperability/Reusability of high level WebAudio components. W3C Workshop on Web Games, Jun 2019, Redmond, United States. hal-02366763

**HAL Id: hal-02366763**

**<https://inria.hal.science/hal-02366763>**

Submitted on 16 Nov 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Interoperability/Reusability of high level WebAudio components

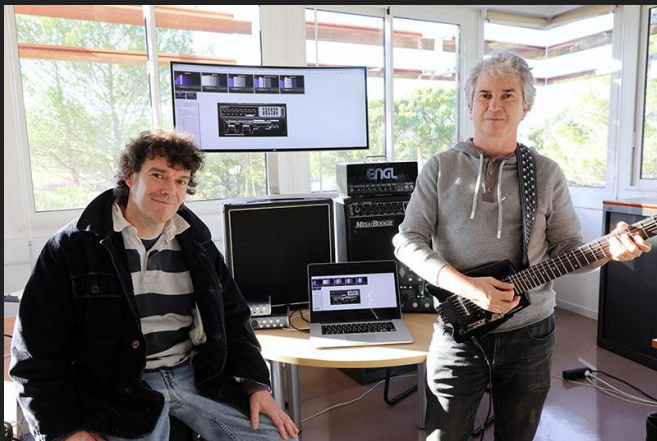
Games on Web W3C Workshop, Seattle 2019

Michel Buffa  
Université Côte d'Azur  
France  
I3S/CNRS/INRIA labs  
buffa@i3s.unice.fr  
@micbuffa



# Who am I?

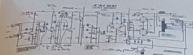
- Professor / researcher at Université Côte d'Azur, in the South of France
  - member of the WIMMICS research group common to INRIA and I3S lab from CNRS
- National coordinator of the WASABI ANR research project, with WebAudio at its heart,
- W3C Advisory Committee Representative for UCA
- I participate to the WebAudio working group



# Real-Time Emulation of a Marshall JCM 800 Guitar Tube Amplifier, Audio FX Pedals, in a Virtual Pedal Board

Michel Buffa, Jérôme Lebrun  
Université Côte d'Azur  
CNRS, INRIA  
(buffa, lebrun)@3s.unice.fr

- WebAudio / WebMidi APIs
- proposal for an open plugin standard (see VST3/STI for the Web)
- Targets: music schools, composers, ...



### Guitar amp schematics

- Plugins can be:
- JavaScript classes
  - WebComponents (GUI)
  - WebAssembly (binary)
  - C/C++ compiled
  - FAUST compiled
- Support for local / remote plugins  
Integrated MIDI support  
Synths by [webaudiomodels.org](http://webaudiomodels.org)  
FAUST DSL: <http://faust.gforge.inria.fr>

Tube guitar amp simulator (designer, final amps packaged as plugins)



Host Webapp for assembling amps/effects and instrument plugins



ANR WASABI  
<http://wasabihome.13s.unice.fr>



Real-Time Emulation of a Marshall JCM 800 Guitar Tube Amplifier, Audio FX Pedals, in a Virtual Pedal Board

Michel Buffa, Jérôme Lebrun  
Université Côte d'Azur  
CNRS, INRIA  
(buffa, lebrun)@3s.unice.fr

- WebAudio / WebMidi APIs
- proposal for an open plugin standard (see VST3/STI for the Web)
- Targets: music schools, composers, ...

Guitar amp schematics

Plugins can be:

- JavaScript classes
- WebComponents (GUI)
- WebAssembly (binary)
- C/C++ compiled
- FAUST compiled

Support for local / remote plugins  
Integrated MIDI support  
Synths by [webaudiomodels.org](http://webaudiomodels.org)  
FAUST DSL: <http://faust.gforge.inria.fr>

Tube guitar amp simulator (designer, final amps packaged as plugins)

Host Webapp for assembling amps/effects and instrument plugins

ANR WASABI  
<http://wasabihome.13s.unice.fr>

# Some ambitious WebAudio examples...

- AudioGraphs of these apps use high-level WebAudio nodes and / or AudioWorklets + WebAssembly.



But... no plugin standard, no “hosts”, no programming model...

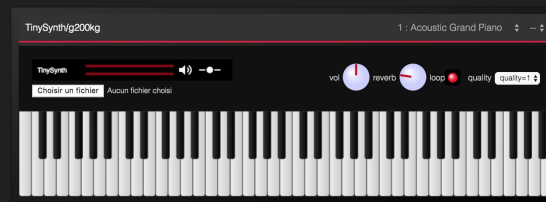
We find some very good JavaScript libraries (i.e. [toneJS](https://github.com/tonedeck))

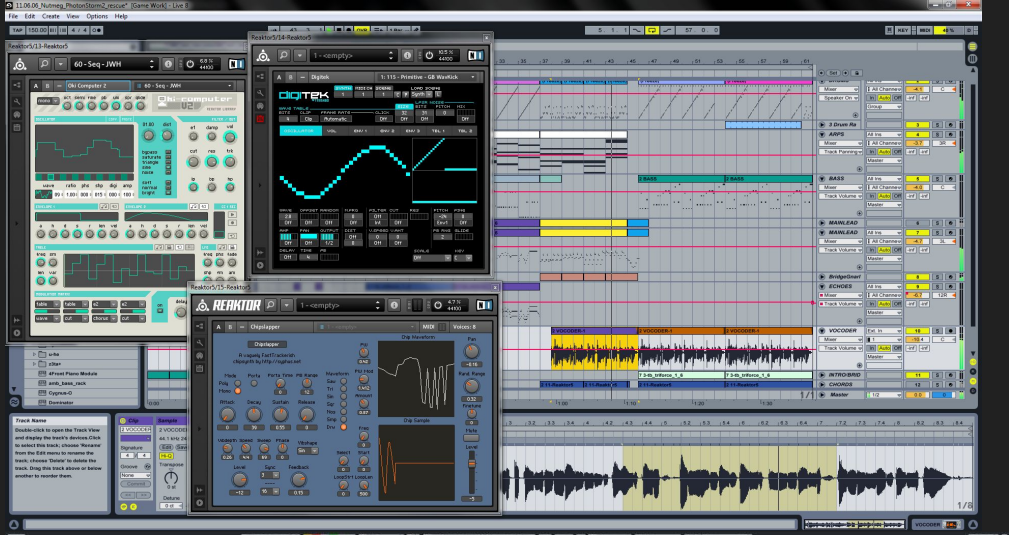
Some open source github repositories (i.e. <https://webaudiodemos.appspot.com/>)

Some online tools for music synthesis ([genish.js](https://genish.js) etc.)

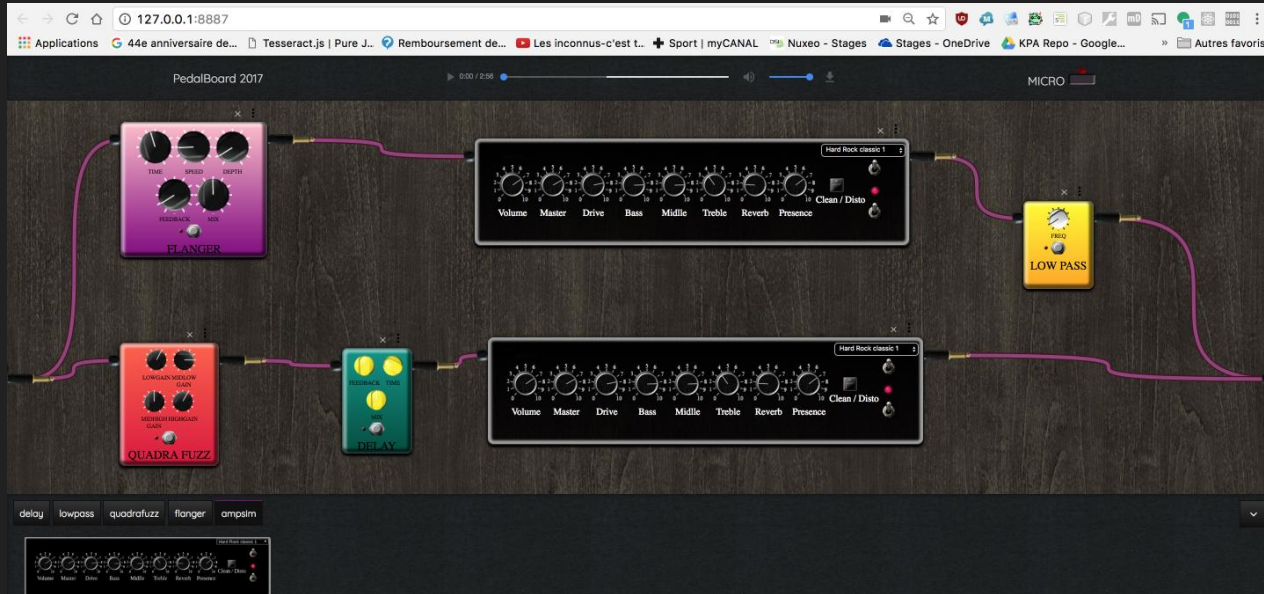
Some DSL for DSP programming ([FAUST](https://faust.cc), etc.)

Some effects and instruments





In early 2018, with some researchers and developers we decided to start working on an open plugin standard for WebAudio





We made a team with different researchers / developers, that share same concerns with different approaches

- **1 - Bringing native developers to the Web**

- a. **Jari Kleimola** (Aalto University Espoo, Southern Finland, now at Webaudiomodules.org),
- b. **Oli Larkin** (Developer of VirtualCZ, Endless Series, WDL-OL/iPlug, iPlug2)

- **2 - Bringing low level DSP developers to the Web**

- a. **Stéphane Letz** (senior researcher at GRAME, Lyon, co-author of the FAUST DSL/compiler)

- **3 - Attract Web Developers / JavaScript audio app developers**

- a. **Tatsuya Shinyagaito, aka g200kg**, (Audio and WebAudio developer, huge WebAudio contributor, Yokohama, Kanagawa, Japan)
- b. **Jerôme Lebrun and Michel Buffa** (I3S/INRIA)

# An open standard = API/specification ? Or more... ?

## Be Web-Aware!

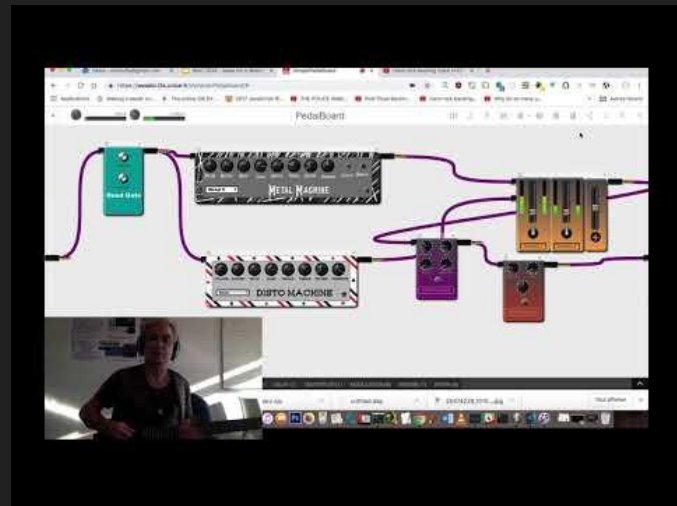
- **Use URIs** : support local or remote plugins, audio and midi routing
- **Asynchronous events in the lifecycle**
- **Plugins can be headless or with a GUI**
- **API for the “audio processor part”, as close as possible to AudioNode**
  - i.e use `plugin.connect(gain)` or `gain.disconnect(plugin)`, etc.
- **Propose an API or at least guidelines on how to package and publish plugins on REST repositories**
- **Avoid naming conflicts** (HTML ids, JS names, CSS rules), metadata...

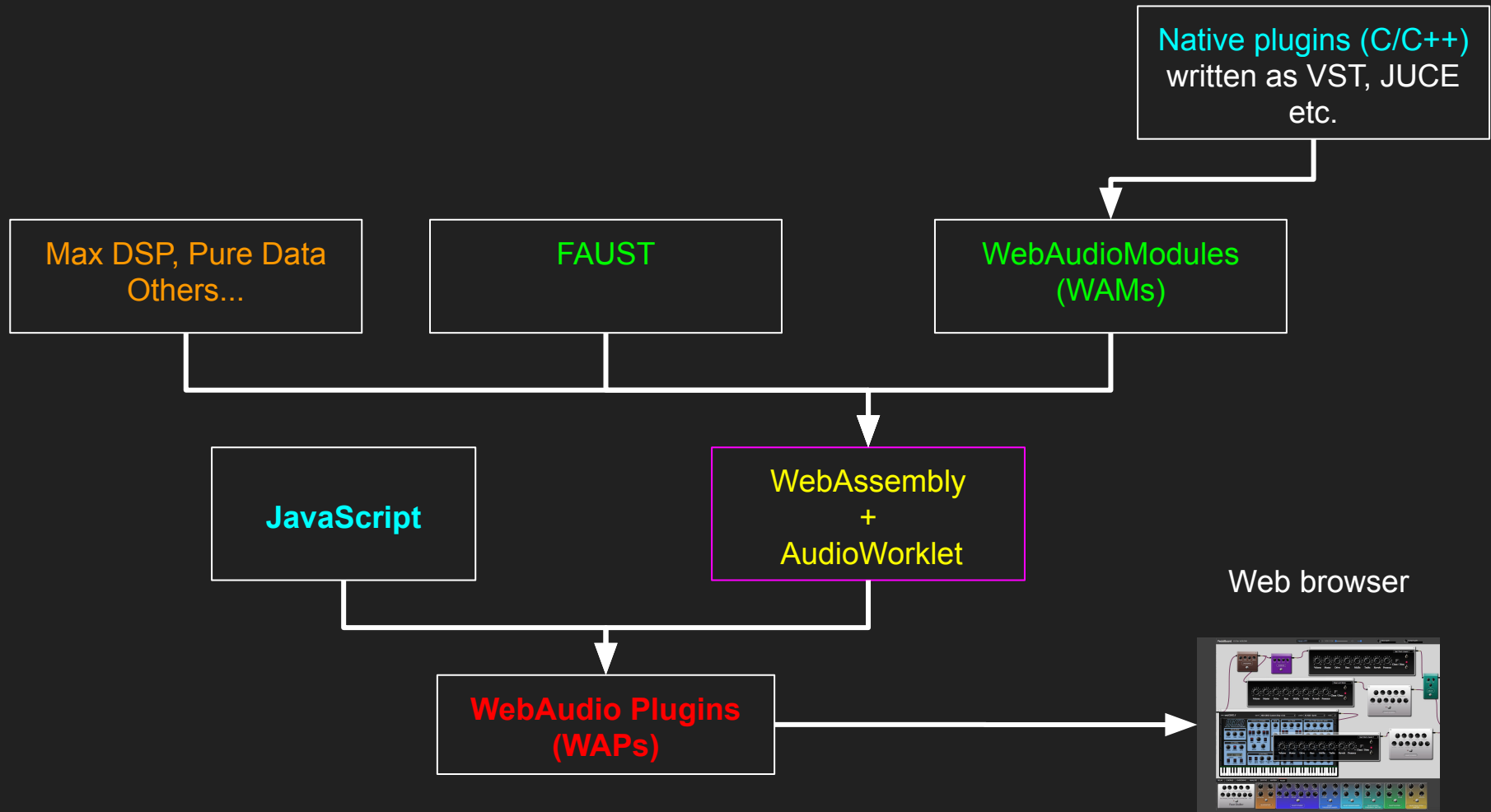
# Let's play with some WAPs (WebAudio Plugins)

The screenshot shows the PedalBoard 2017 software interface. At the top, it displays "Metal riff 1", "MICRO" status, "Input level", and "Output level". The main area is titled "WAM PLUGINS" and contains several virtual pedals connected in a signal chain:

- Two keyboard-based WAPs: "webDX7 //" and "webOBXD //".
- A purple "CHORUS" pedal with knobs for DELAY, SPEED, DEPTH, and MIX.
- A red "overdrive" pedal with knobs for DRIVE and GAIN.
- A silver "FAUST ZitaRev" pedal with multiple frequency sliders (Hi, Lo, Mid, etc.) and a "Distortion" knob.
- A black "Hard Rock classic 1" pedal with knobs for Volume, Master, Drive, Bass, Middle, Treble, Reverb, and Presence, plus a "Clean / Disto" switch.
- A green "DELAY" pedal with knobs for FEEDBACK and TIME.
- A blue "Sound Checker" pedal with a waveform display and a "Clip Frequency Oscillator" control.

Red arrows point from the text "WebAudio plugins written in JavaScript" to the CHORUS, overdrive, and DELAY pedals. Another red arrow points from "FAUST PLUGIN" to the FAUST ZitaRev pedal. A bottom panel shows a menu with "delay", "chorus", "overdrive", "analyse", "Synths", "ampsim", and "FAUST", with the FAUST ZitaRev pedal selected.





```
// Once the script has been loaded instantiate the plugin
buildPlugin(className, baseUrl);
}

// will be executed before the onload above...
document.head.appendChild(script);
}

// instantiate the plugin
function buildPlugin(className, baseUrl) {
  var plugin = new window[className](ctx, baseUrl);
  console.log(plugin);

  plugin.load().then((node) => {
    // loads and initialize the audio processor part
    // Then use the factory to create the HTML custom elem that holds the GUI
    // The loadGUI method takes care of inserting the link rel=import part,
    // not doing it twice, and associate the node with its GUI.
    if (checkbox.checked) {
      plugin.loadGui().then((elem) => {
        console.log("ADDING PLUGIN");
        // show the GUI of the plugin, the audio part is ready to be used
        document.querySelector("#WAP").appendChild(elem);
        //mediaSource.connect(node);
        //node.connect(ctx.destination);
        // Add node to the chain of plugins
      });
    }
    document.body.querySelector("#WAP").insertAdjacentHTML('afterbegin', '<h2>' + `${className}`
  try {
    mediaSource.connect(node);
  } catch (error) {
    console.log("this plugin does not use audioworkletnode or compositenode");
    mediaSource.connect(node.getInput(0));
  }

  node.connect(ctx.destination);

  bt_buildIt.addEventListener('click', () => {
    mediaSource.disconnect();
    node.disconnect();
    document.querySelector("#WAP").innerHTML = "";
    document.querySelector("#mocha").innerHTML = "";
  });
});
}
```

Output

Run with JS Auto-run JS

0:11 / 2:56

### Paste here the link to your webaudio plugin

Plugin URL (repository where your main.json file is)

<https://wasabi.i3s.unice.fr/WebAudioPluginBank/jordan-S>

GUI

build it

Try different plugins click on links to fill the form field, then press the "build" button:

- [Metal Machine amp sim](#)
- [Acoustic guitar amp sim](#)
- [PingPongDelay3](#)
- [ZitaRev](#)
- [Blipper](#)
- [Flanger](#)

## WasabiDistoMachine

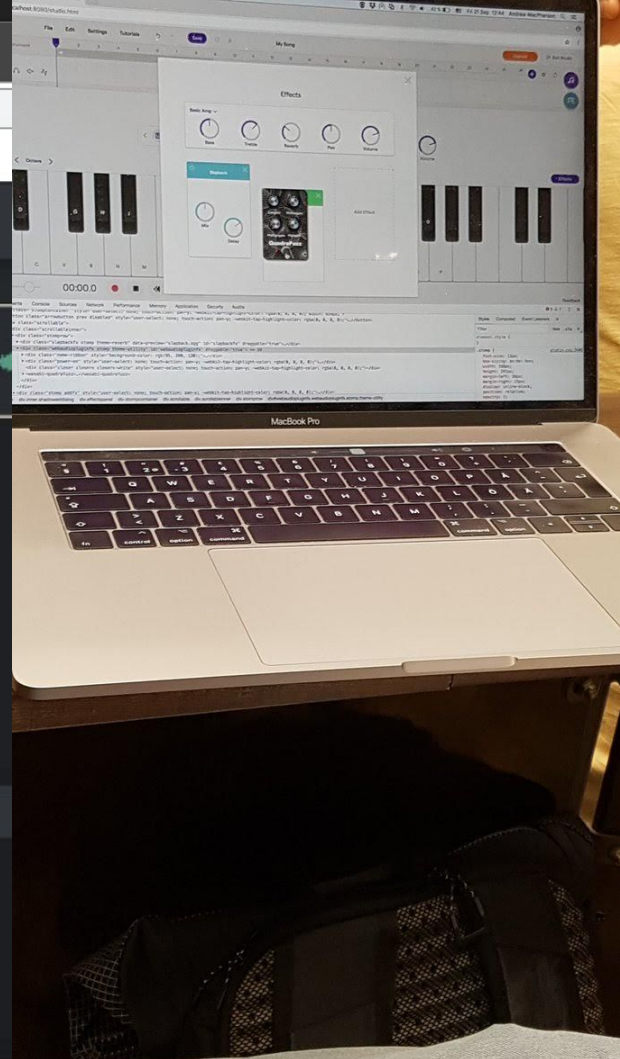
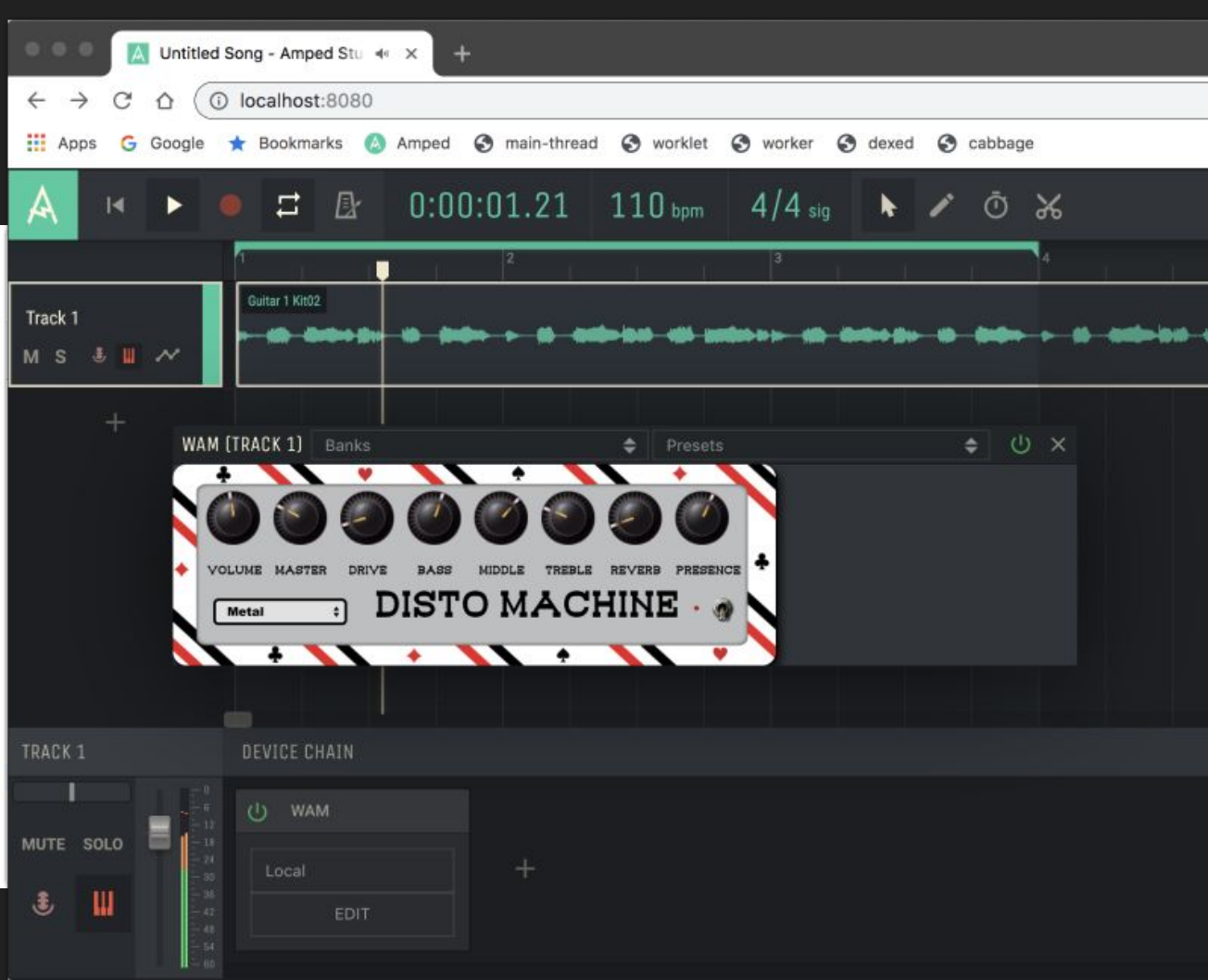


Browser window: **Untitled Song - Amped Stu** | **localhost:8080**

Navigation: Apps, Google, Bookmarks, Amped, main-thread, worklet, worker, dexed, cabbage

DAW Interface:

- Timeline: 0:00:01.21 | 110 bpm | 4/4 sig
- Track 1: Guitar 1 Kit02
- WAM [TRACK 1] Banks Presets
- Distortion Module: **DISTO MACHINE** (Metal preset)
- Parameters: VOLUME, MASTER, DRIVE, BASS, MIDDLE, TREBLE, REVERB, PRESENCE
- Device Chain: TRACK 1 | MUTE SOLO | WAM | Local | EDIT



# Low level DSP automatized -> WASM

The screenshot displays a web-based DSP editor interface. The top part shows a code editor with the following code:

```
13 block_on(fx) = par(i, inputs(fx), _f[!-bypass]);
14 block_off(fx) = par(i, inputs(fx), _bypass);
15
16 bypass_fx(fx) = par(i, inputs(fx), _) <: ((block_on(fx):fx), block_off(fx));> par(i, outputs(fx), _);
17
18 basepitch = hslider("BasePitch [unit:semitones] [OWL:PARAMETER_A] [style:knob]", 60, 24, 96, 0.1) : si.smooth(ba.tau2pole(0.01));
19 pitchmod = hslider("PitchMod [unit:semitones] [OWL:PARAMETER_B] [style:knob]", 24, -64, 64, 1) : si.smooth(ba.tau2pole(0.005));
20 //attack = hslider("Attack [unit:ms] [OWL:PARAMETER_C]", 2, 2, 1000, 1) : *(0.001) : max(1.0/float(ma.SR));
21 release = hslider("Release [unit:ms] [OWL:PARAMETER_C] [style:knob]", 20, 2, 100, 1) : *(0.001) : max(1.0/float(ma.SR));
22 attack = 0.005;
23 mix = hslider("Mix [OWL:PARAMETER_D] [style:knob]", 0.5, 0, 1, 0.01) : si.smooth(ba.tau2pole(0.005));
24
25 blipper(l, r) = l, r <: *(1-mix), *(1-mix), mono2stereo -> _;
26
27 with {
28 mono2stereo = + : pc2 * mix <: _;
29 pc2 = an.amp_follower_ud(attack, release) <: (ba.midikey2hz(basepitch + (pitchmod * _)) : os.triangle, _ : *);
30 };
31 process = bypass_fx(blipper);
```

The left sidebar contains various controls:

- Run** button
- Share** button
- Poly Voices** dropdown (set to Mono)
- Buffer Size** dropdown (set to 1024)
- Use AudioWorklet
- Save DSP Code
- Save Params State
- Save DSP Cache
- Real-time Compile
- Popup UI
- Plot** section with **Mode** dropdown (set to Offline), **Samples** dropdown (set to 1024), **Sample Rate** dropdown (set to 48000), **FFT Size** dropdown (set to 256), and  Draw Spectrogram.
- A Plot First Samples button

The bottom part of the interface is split into two panels:

- Diagram** panel: Shows a GUI Builder with fields for Name, Background color (purple), Image opacity (1), Width (241 px), Height (277 px), Radius (10 px), and Background Images.
- GUI Preview** panel: Displays a preview of the GUI with four knobs labeled BasePitch, Mix, PitchMod, and bypassRelease. A message above the preview says "Welcome on the WAP GUI2 editor ! click on MENU to start !".


# Online testers (individual plugin, repository)

100% passed: 12 failures: 5 duration: 0.05s

## Plugin Tester

Paste here the link to your webaudio plugin

URL:



Metadata

- ✓ plugin should have a JSON `getMetadata()` method
- ✓ the `getMetadata()` function should return a json object

Descriptor

- ✓ plugin should have a JSON `getDescriptor()` method
- ✓ `getDescriptor()` function should return a json object

Param getter

- ✗ plugin should have a `getParam(key)` method

```
AssertionError: expected undefined to exist at Context.<anonymous> (test.html:135:54)
```

- ✗ the `getParam()` function should not be empty


```
AssertionError: .empty was passed non-string primitive undefined at Context.<anonymous> (test.html:138:61)
```

100% passed: 14 failures: 0 duration: 0.02s

## Plugin Tester

Enter the URL of a WebAudio plugin repository (path to the json file which describe it) and press the button to start discovering/loading plugin thumbnails.

Click on a thumbnail to create an instance of the plugin Test it's compatibility with the API Here the input default value is <https://wasabi.i3s.unice.fr/WebAudioPluginBank/repository.json> You can try the WAM repository <https://codeandmobile.com/repository.json>



Metadata

- ✗ plugin should have a JSON `getMetadata()` function

Descriptor

- ✗ plugin should have a JSON `getDescriptor()` function

Param getter

- ✗ plugin should have a `getParam(key)` method

Param setter

- ✗ plugin should have a `setParam(key, value)` method

100% passed: 14 failures: 0 duration: 0.02s

## Plugin Tester

Click to fill with the URL of :

- WasabiPingPongDelay - FaustZitaRev - LarkinBlipper - LarkinFlanger
- WasabiMinilogue - CleanMachine - QuadraFuzz

-Paste here the link to your webaudio plugin

**PLUGIN API TESTER**

Shows declared params, inputs, etc.

**Node Part**

URL: node.URL

```
"https://wasabi.i3s.unice.fr/WebAudioPluginBank/WASABIPingPongDelay3/3/"
```

Descriptor: `node.getDescriptor()`

```
{ "mix": { "minValue": 0, "maxValue": 1, "defaultValue": 0.5, "time": { "minValue": 0, "maxValue": 1, "defaultValue": 0.5 } }, "feedback": { "minValue": 0, "maxValue": 1, "defaultValue": 0.5 } }
```

Node input number `node.numberOfInputs` : 1

Params at this time `node.getState()` (promise)

```
{ "mix": 0.5, "time": 0.5, "feedback": 0.5, "status": "disable" }
```

Metadata `node.getMetadata()` (promise)

```
{ "documentation": "...", "name": "PingPongDelay", "thumbnail": "WasabiPing..." }
```



# Conclusion : Where are we today?

SDK for JS developers

FAUST scripts and IDE that compile .dsp files to WAPs, embedded GUIs builder, publish to remote servers

WebAudioModules C/C++ toolchain for native audio developers (and WAMs are WAPs)

Multiple examples of hosts that load plugins dynamically using URIs

Tools: plugin validator, repository validator, GUI editor

[Check the GitHub!](#)

[Check the pedalboard demo!](#)





LEARN A SONG  
LESSON MODE  
PAUSE/STOP/PLAY  
LESSONS  
GUIT ARCADE  
MULTIPLAYER  
TONE DESIGNER  
SHOP  
UPLAY

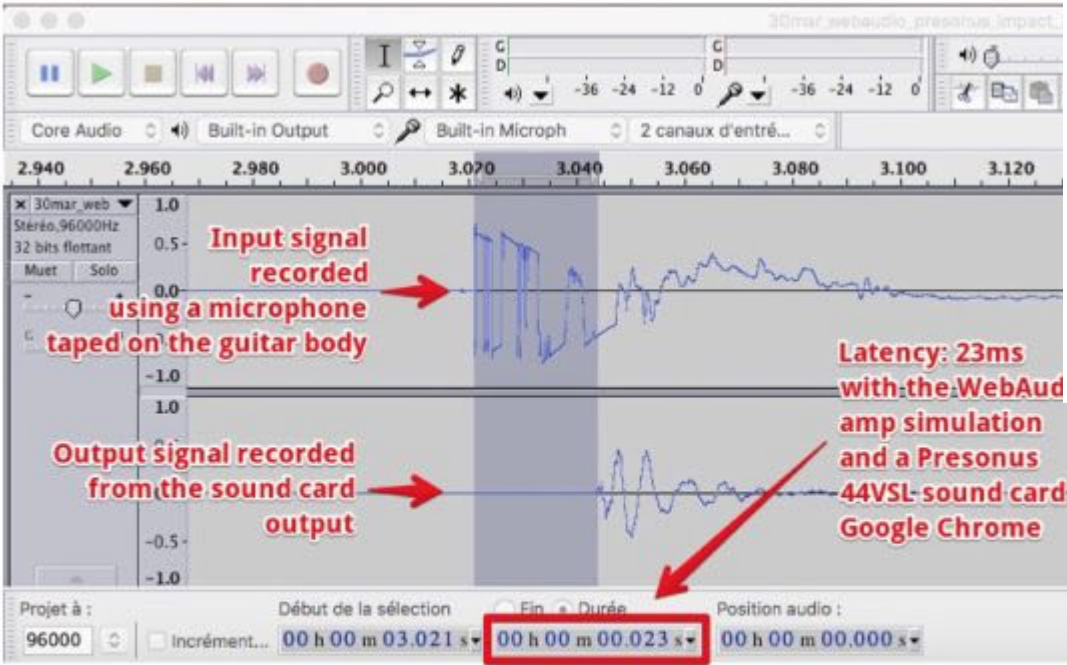
Rocksmith  
Available on PS4, Xbox One, and PC

# Guitar Hero / Rocksmith for the Web?

Yes, this is possible

- Good latency
- Pitch detection

Still the latency is not addressed



**Figure 7: Example of measure in Audacity, here the WebAudio amp sim with Google Chrome and a Presonus 44VSL sound card.**

