



Web Audio Modules 2.0: Audio Plugins for the Web Platform!

Michel Buffa

► To cite this version:

Michel Buffa. Web Audio Modules 2.0: Audio Plugins for the Web Platform!. ADC 2022 - Audio Developer Conference, Nov 2022, Londres, United Kingdom. hal-03871660

HAL Id: hal-03871660

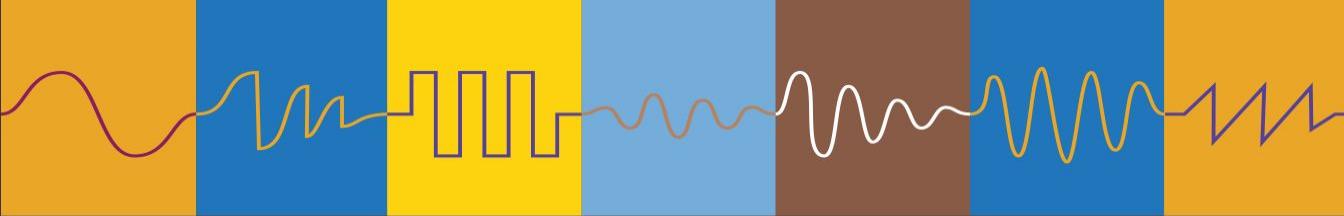
<https://inria.hal.science/hal-03871660>

Submitted on 25 Nov 2022

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.

ADC22



WEBAUDIO MODULES 2.0: *AUDIO PLUGINS FOR THE WEB PLATFORM!*

MICHEL BUFFA



Who am I? What is our group?

- Professor / researcher at Université Côte d'Azur (UCA), France
 - Member of the WIMMICS research group common to INRIA and I3S lab from CNRS
 - W3C Advisory Committee Representative for UCA
 - Member of the W3C WebAudio Working Group since 2014
 - michel.buffa@univ-cotedazur.fr, @micbuffa
- Other members of the WAM group:
 - WAM original creators: Jari Kleimola And Oliver Larkin,
 - Developers, academic researchers, PhD Students : Shihong Ren, Owen Campbell, Tom Burns, Steven Yi, Stéphane Letz, Hugo Mallet...
 - Thanks to: Jordan Sintes, Guillaume Etevenard, GRAME friends...



WAMs



Web Audio Modules: VSTs for the Web!



Wams in hosts... here with a host from the WAM distrib https://mainline.i3s.unice.fr/wam2/packages/_/

Example WAM Host

NEW: See WAM 2.0 API docs

NEW: See WAM 2.0 Parameter Manager docs

0:00 / 2:56

Select MIDI input device Select...

Select live input device Par défaut - MacBook Pro Microphone (Built-in)

Live input: NOT ACTIVATED, click to toggle on/off.

Enter any WAM Plugin URL ./obxd/index.js

LOAD PLUGIN

Midi

- Simple MIDI Keyboard
- Midi Virtual keyboard No Sound (just emits events)
- Random note generator
- Simple Transport
- MIDI Sequencer

Pure JavaScript

- Quadratuzz
- Quadratuzz without builder
- Disto Machine without builder
- PingPongDelay
- Graphic Equaliser (pureJS)
- WAM Example (pureJS)
- TinySynth, a GM synth without GUI. Accepts program changes, volume_change, notes...
- Wam Event Viewer
- WebMIDI Output

Faust

- faustPingPongDelay
- faustPingPongDelayDefaultUI
- Guitar AmpSim 60s (generated by faust IDE)
- Stone Phaser (generated by faust IDE)
- TS9 Overdrive (generated by faust IDE)
- BijouFuzz Fuzz (generated by faust IDE)
- faust Flute MIDI

TypeScript

- LivesGain
- Oscilloscope
- Spectroscopic
- Spectrogram

CSound

mainline.i3s.unice.fr/wam2/packages/_/ Applications MidJourney Showc... An evil monkey in... MHKG program an... (8) Comment utili... Formalités admini... maquette MIAGE ... sco.p Mettre à jour Autres favoris

Program Change: 0:00 / 2:56

Volume: 100%

MidiOut

flute

freq band gain envelope sustain

otherParams

modulationDepth vibratoDepth outGain

gate

LOAD PLUGIN

SAVE STATE RESTORE STATE

```
instance.descriptor : {"name":"Faust Flute MIDI","vendor":"Graeme","description":"Faust MIDI Instrument","version":"1.0.0","apiVersion":2.0.0,"thumbnail":"screenshots/flute","faust":true,"isInstrument":true,"website":"","hasAudioInput":true,"hasAudioOutput":true,"gui.properties.dataWidth.value": undefined, (you should define get properties in Gui.js)}  
gui.properties.dataHeight.value : undefined, (you should define get properties in Gui.js)  
instance.audioNode.getParameterInfo() :
```

- ✓ /flute/otherParams/vibratoFreq : ("id":"/flute/otherParams/vibrateFreq","label":"/flute/otherParams/vibratoFreq","type":float,["units":""})
- ✓ /flute/midi/freq : ("id":"/flute/midi/freq","label":"/flute/midi/freq","type":float,["units":""})
- ✓ /flute/midi/main : ("id":"/flute/midi/main","label":"/flute/midi/main","type":float,["units":""])

Add Automation... Duration (sec): 3 Apply

0,0,9
0,24,0,1
0,89,0,79
1,49,0,14
2,49,0,68

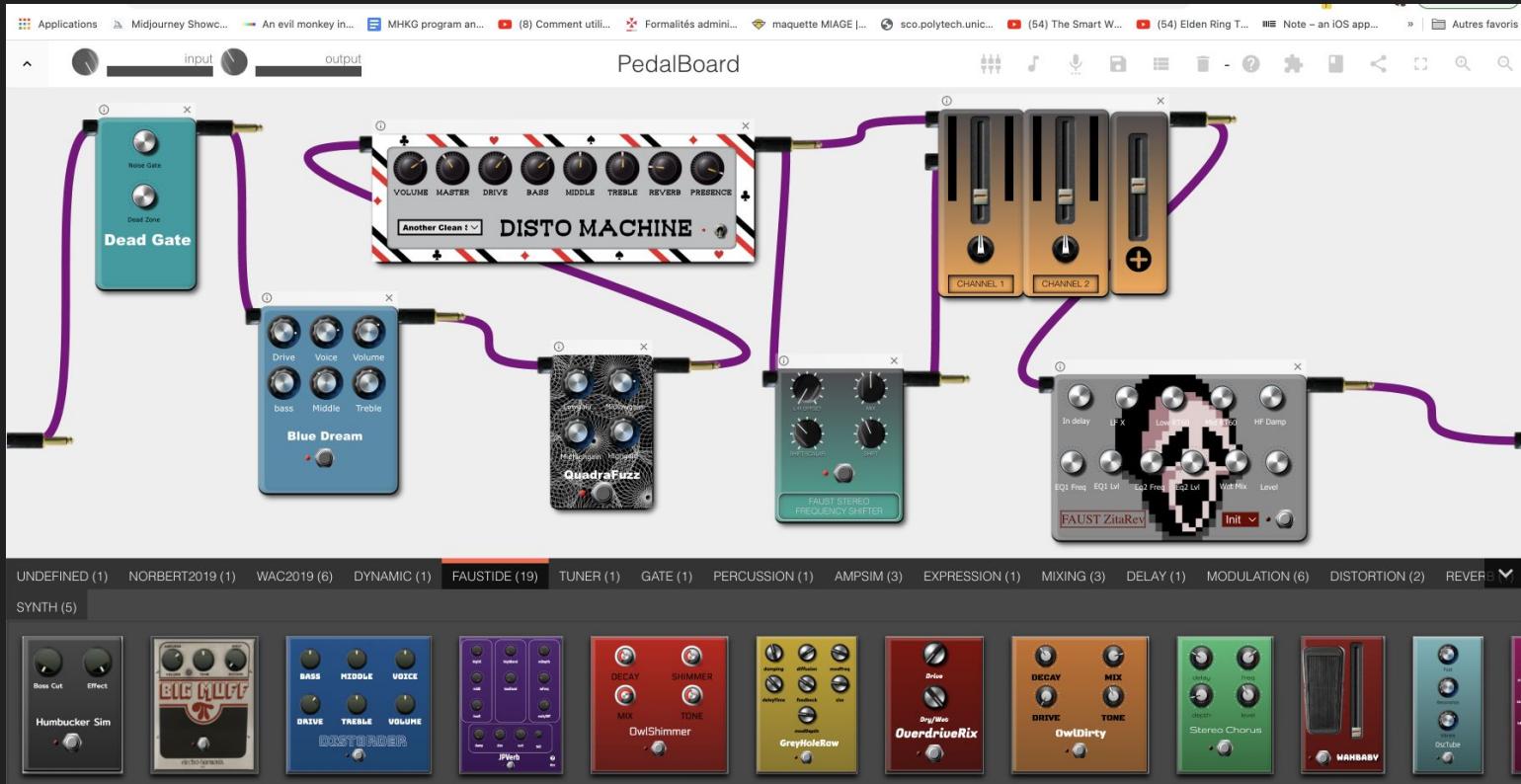
Send MIDI Note

Typescript

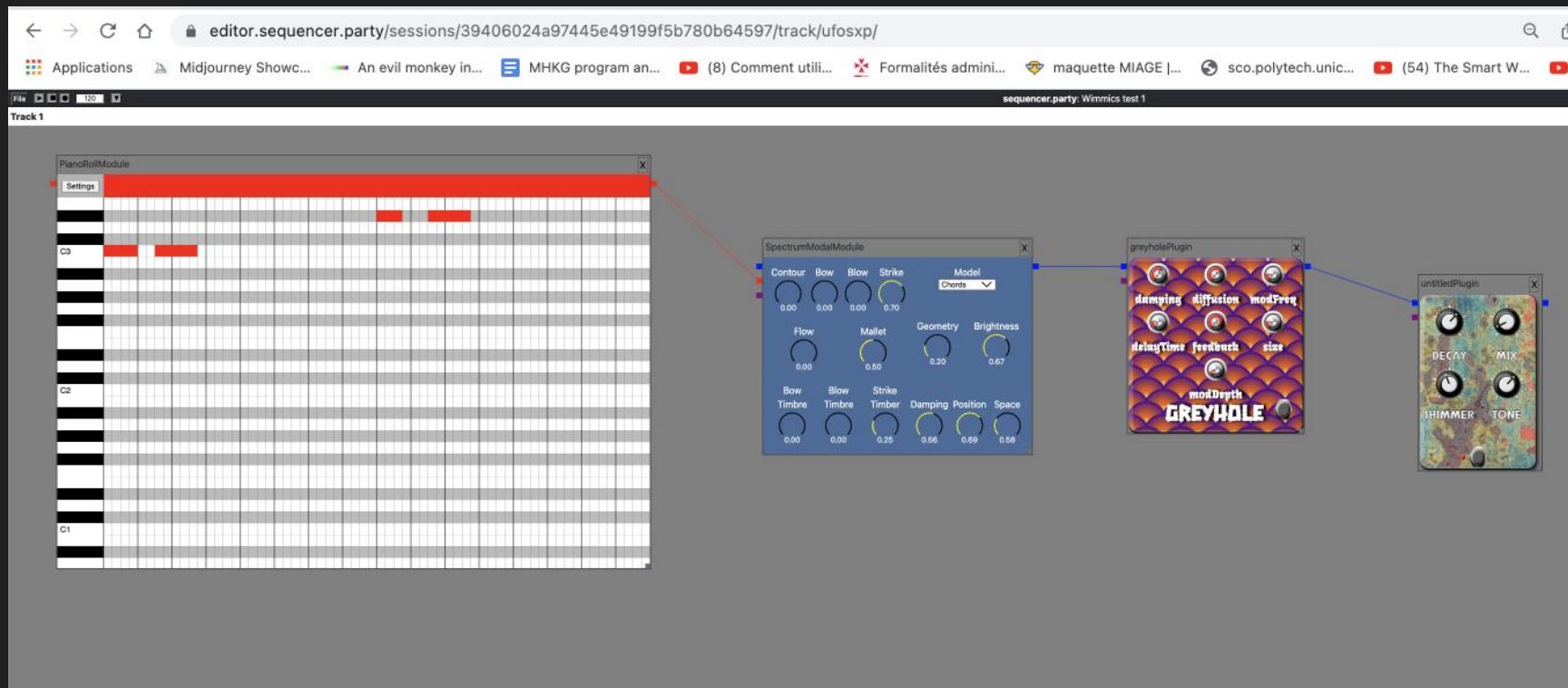
WAMs in hosts... WIP open source DAW (100% WAM-based)



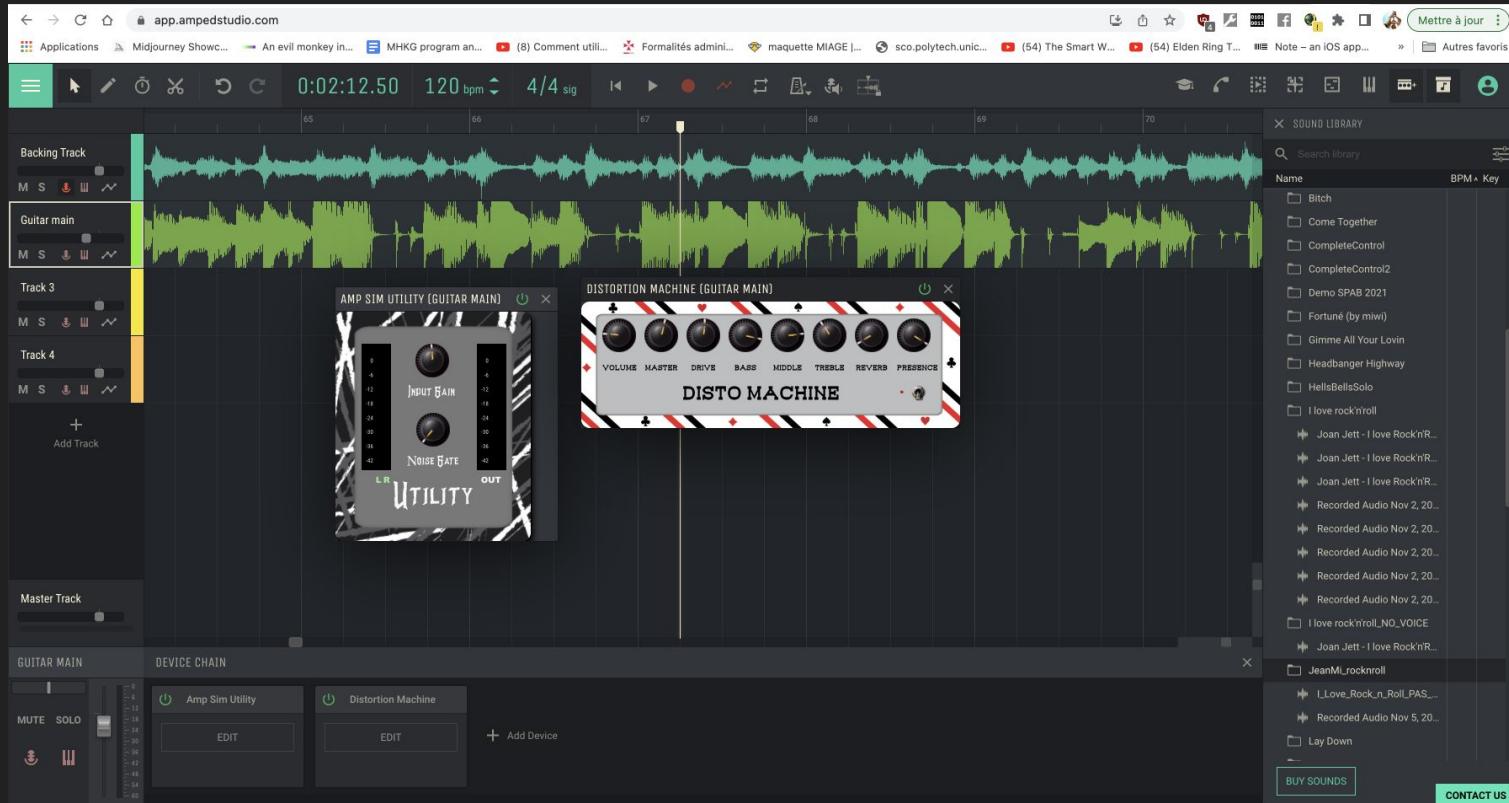
WAMs in hosts... here the WASABI pedalboard



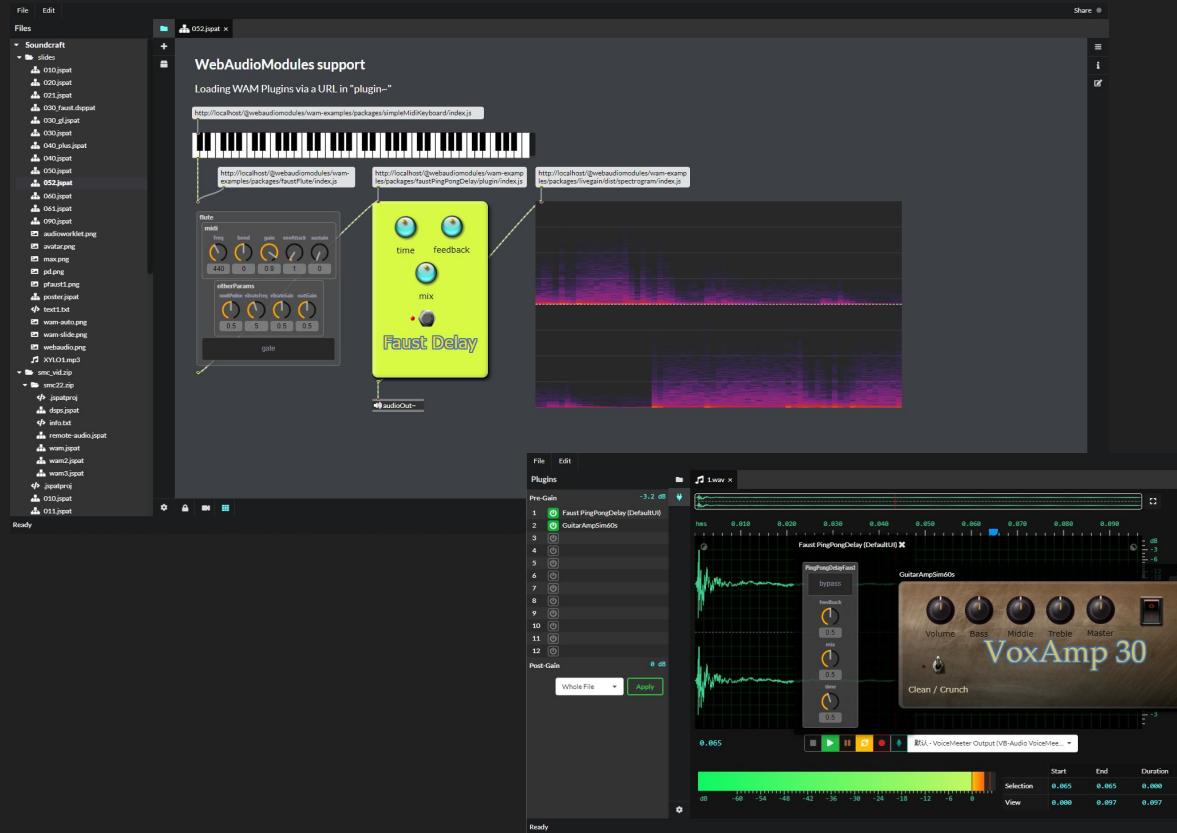
WAMs in hosts... <https://sequencer.party/>



Wams in hosts... <https://app.ampedstudio.com/>



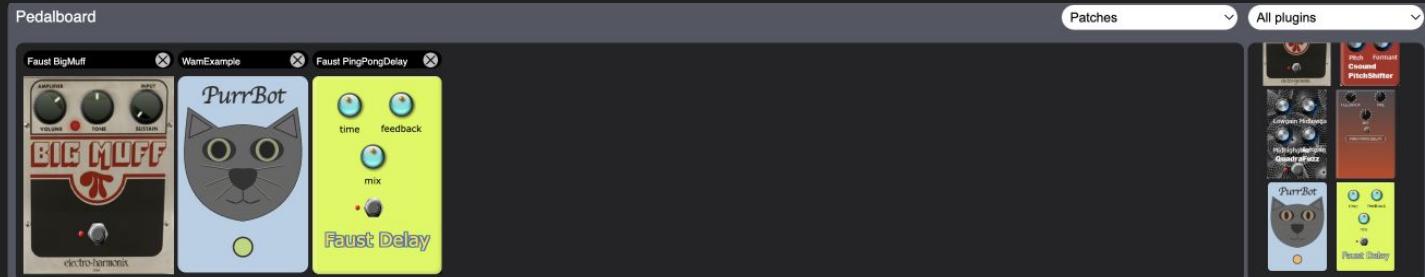
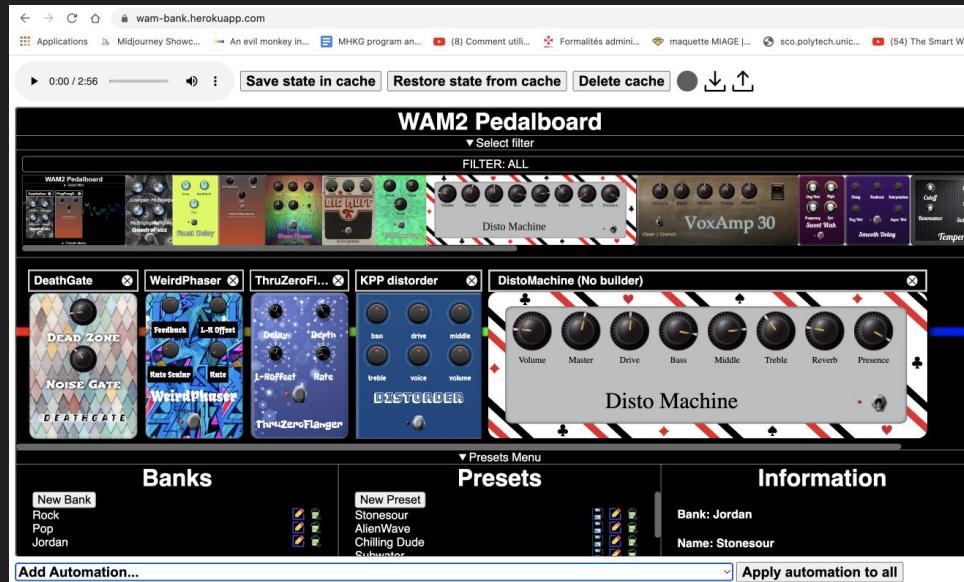
WAMs in hosts... JSPatcher, aka Max MSP in the browser (<https://github.com/FrostbyteR/jspatcher>)



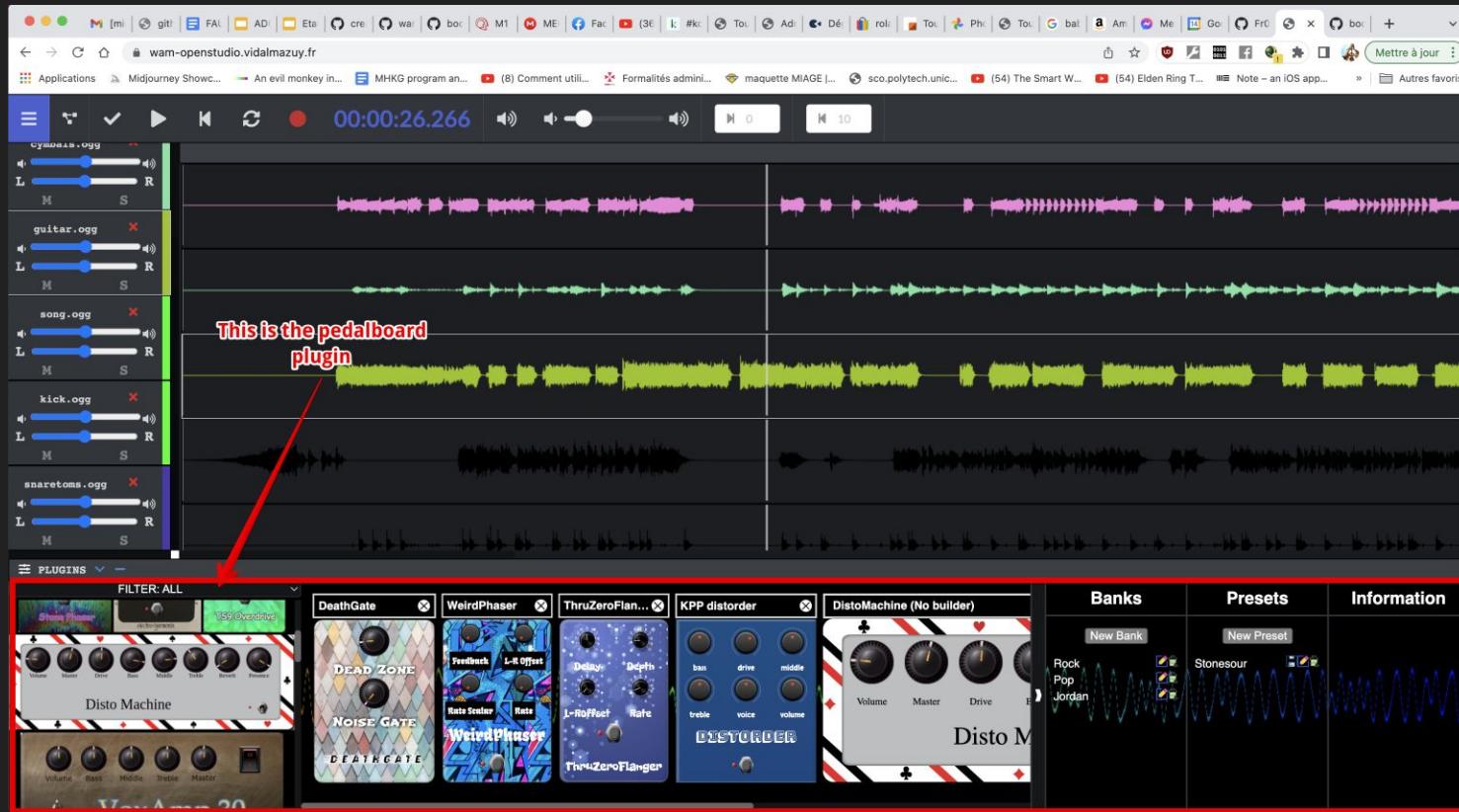
WAMs in plugins that acts as hosts (i.e pedalboard)

These ones
are open
source

(from [wam-examples](#)
and [wam-community](#)
github repositories)



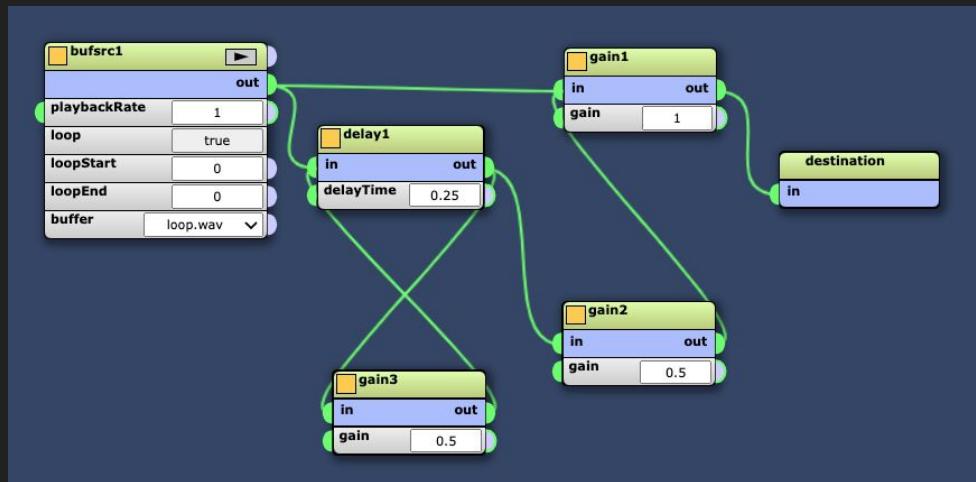
... and loaded in a DAW!



A few reminders before proceeding...

W3C WebAudio API: connecting nodes and build an audio graph, it's a low level API...

Example: an audio delay effect, made with standard audio nodes. You assemble the graph in JavaScript (main thread), then control the parameters, the GUI. Audio rendering is done in the browser, in the audio thread.

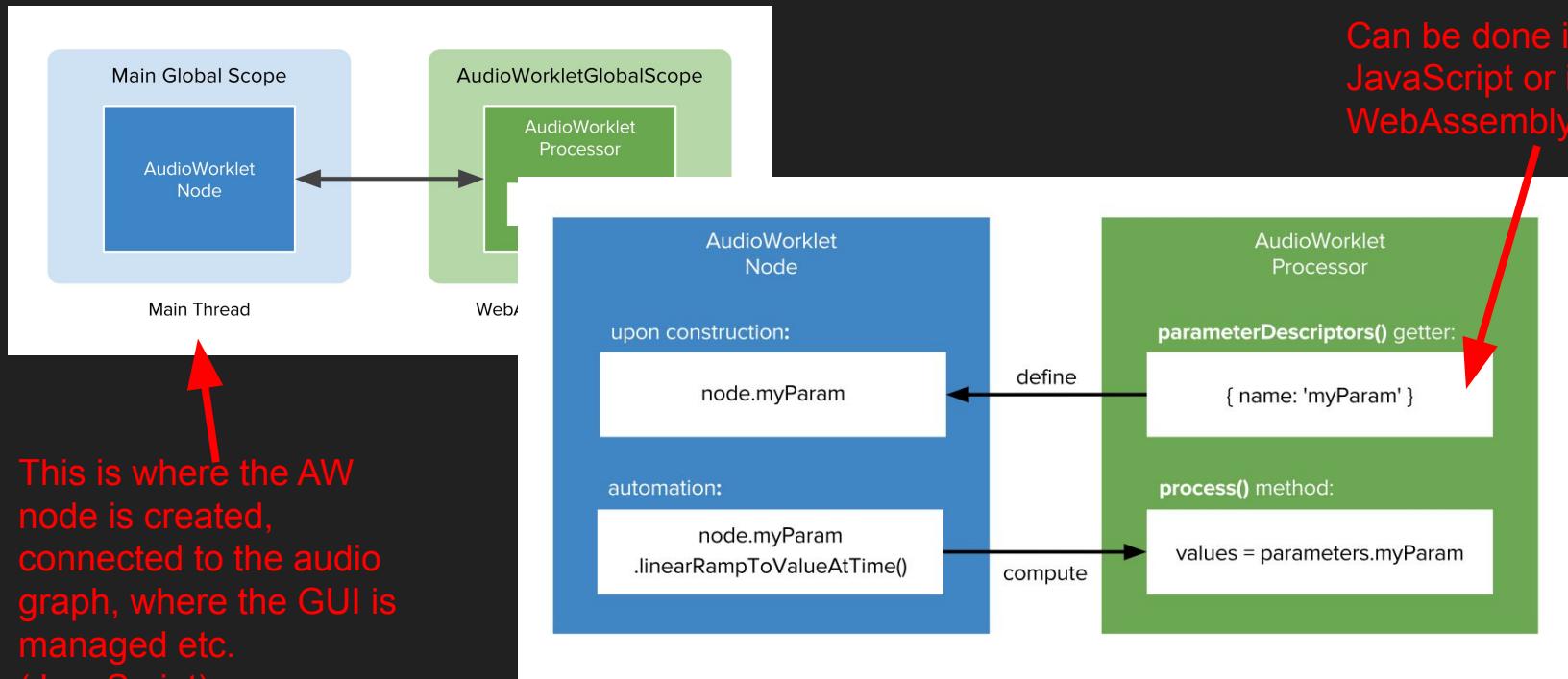


```
this.nodes.gain1 = this.audioctx.createGain();
this.nodes.bufsrc1 = this.audioctx.createBufferSource();
this.nodes.bufsrc1.loop = true;
this.nodes.bufsrc1.buffer = this.buffers['loop.wav'].data;
this.nodes.delay1 = this.audioctx.createDelay();
this.nodes.delay1.delayTime.value = 0.25;
this.nodes.gain2 = this.audioctx.createGain();
this.nodes.gain2.gain.value = 0.5;
this.nodes.gain3 = this.audioctx.createGain();
this.nodes.gain3.gain.value = 0.5;

this.nodes.gain1.connect(this.nodes.destination);
this.nodes.bufsrc1.connect(this.nodes.delay1);
this.nodes.bufsrc1.connect(this.nodes.gain1);
this.nodes.delay1.connect(this.nodes.gain2);
this.nodes.delay1.connect(this.nodes.gain3);
this.nodes.gain2.connect(this.nodes.gain1);
this.nodes.gain3.connect(this.nodes.delay1);
```

The Web Audio API also supports custom DSP programming with the AudioWorklet

This is where custom DSP processing is done!



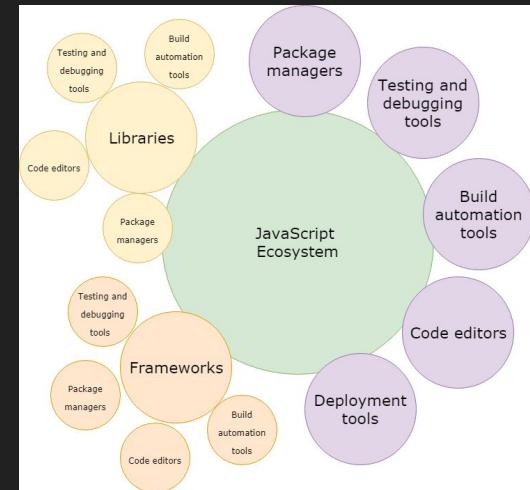
Web Audio development

Web developers

- Use plain HTML/CSS/JavaScript but very often also bundlers/minifiers (webpack, parcel, rollup), npm modules, frameworks (react, vueJS), and also code with TypeScript, etc.

Audio developers

- Use C++/Rust, DSLs like FAUST, Csound, CMajor, patchers like Max, etc.
- Use plugin standards: VSTs, AU, AAX, RTAS, JUCE, CLAP, iPlug2, etc.



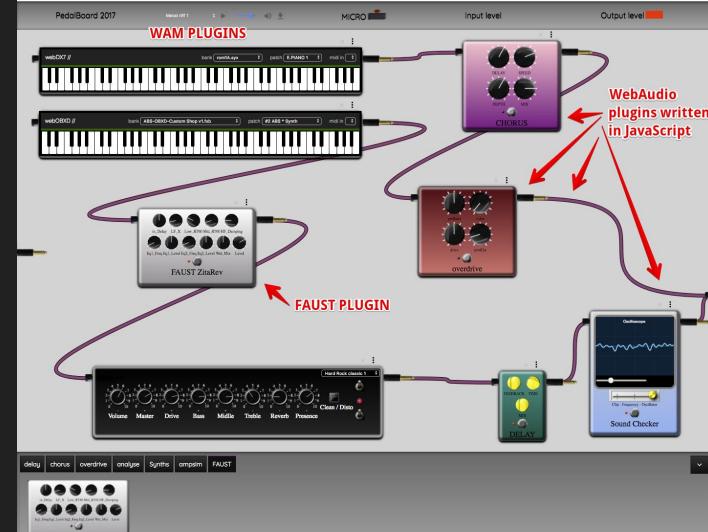
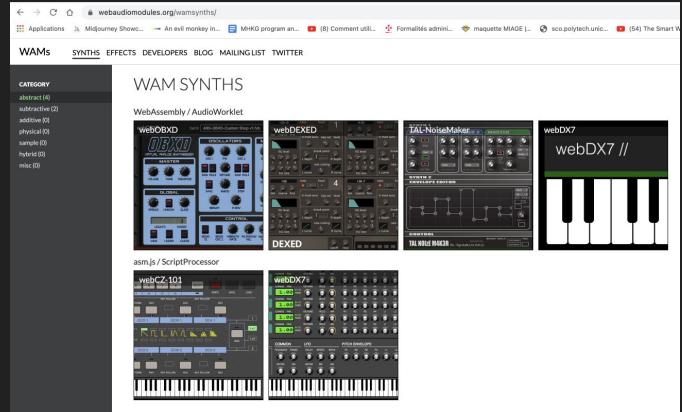
How to combine all this?

2015: First WAM proposal by J.Kleimola and O.Larkin

- Attract native developers, help going from C++ plugins to AudioWorklet/ASM.js and later WebAssembly,
- <http://webaudiomodules.org> has impressive synths ported from VST/JUCE/iPlug2

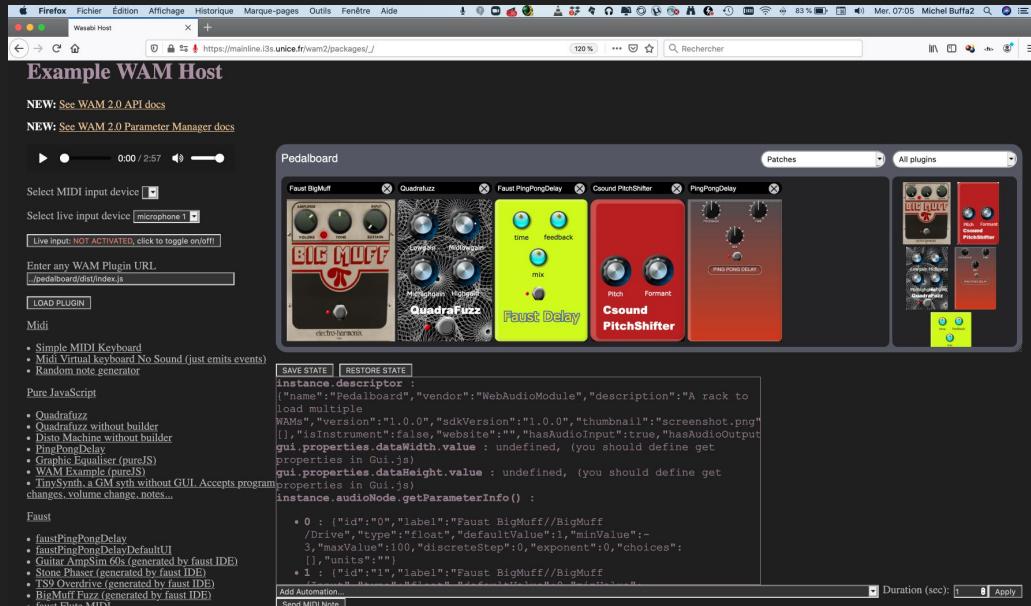
2018: Enlarge the proposal (Buffa and al.)

- Please web developers,
- Support DSLs like FAUST, other improvements...



WebAudioModules version 2 (aka WAM or WAM2)

- **2021-2022: WebAudio Modules 2.0**
 - A WAM plugin can be loaded using a simple URI!
 - A WAM plugin is a JavaScript module,
 - A WAM can be made of a single AudioWorklet Node, or made of multiple nodes, it will behave like a single AudioNode.
 - Plugin parameters are handled by the WamParamMgr,
 - Focus on performance (ring buffer, audio thread isolation)
 - Plain modern JS or build systems for JS / TS / frameworks
 - Support for C/C++
 - Support for DSL (Faust, CSound)
 - Parameter Automation, MIDI support,
 - host/plugin interaction as an API (+ rich SDK). The API can be entirely re-implemented for low-level plugins

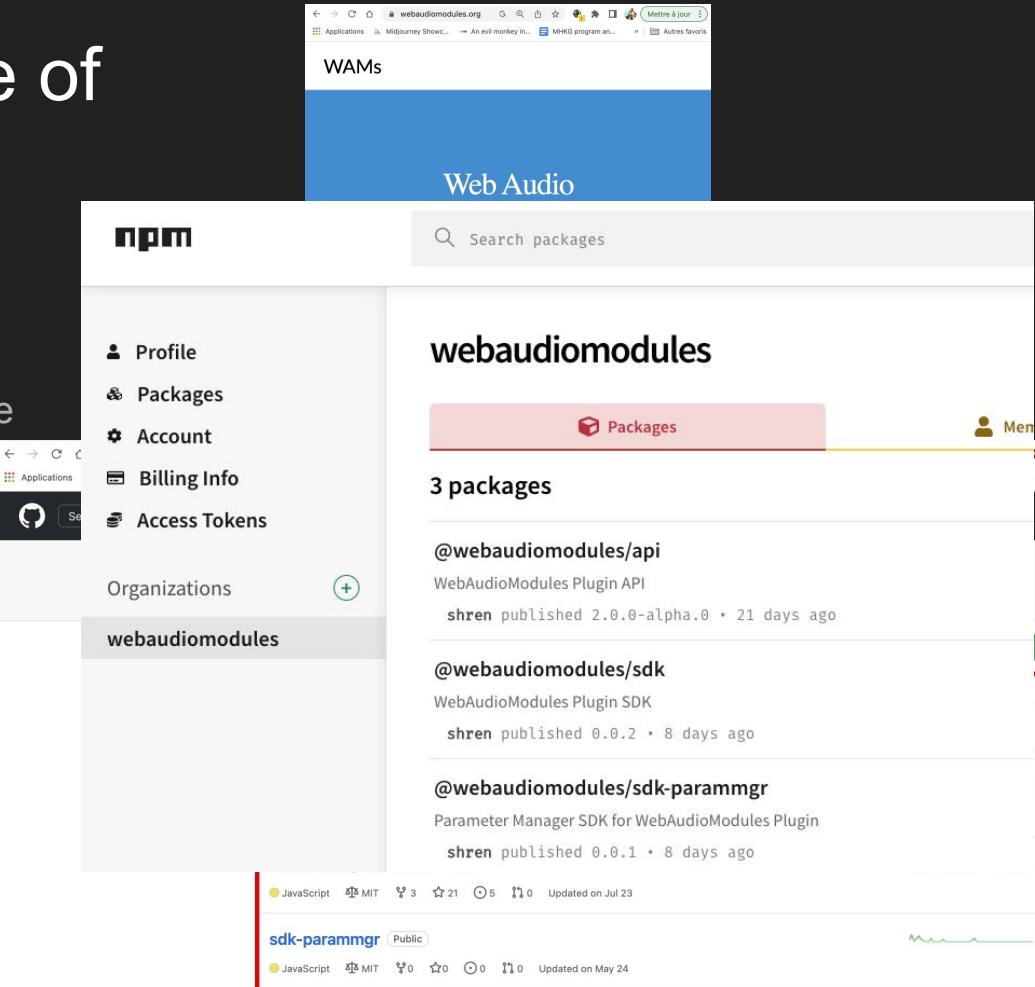


How to start with WAMs!

Github repo, Home page of the project...

WebAudioModules (WAM) is an old standard (2015), and WAM2 is the updated version:

- webaudiomodules.org will remain the home of the project (not yet up to date! Soon with a section about WAM2!)
- The official github repo is the regular webaudiomodules one:
<https://github.com/webaudiomodules>.
- Everything is under MIT/MPL/Apache 2.0 open source licence...
- Also available as [npm modules](#)



API vs SDK

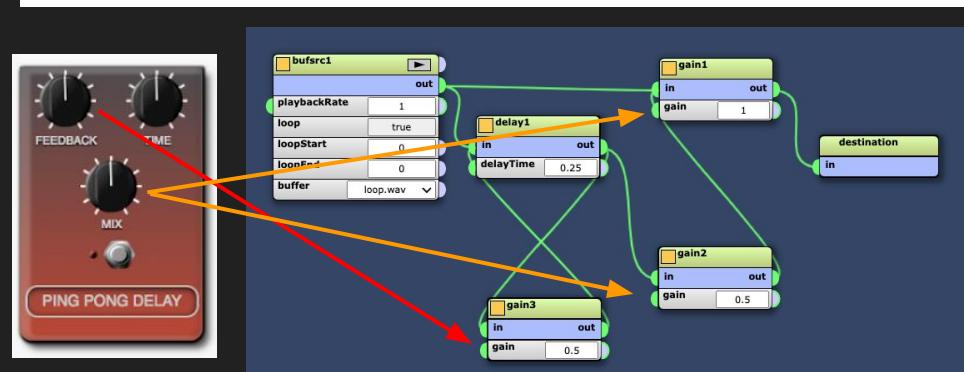
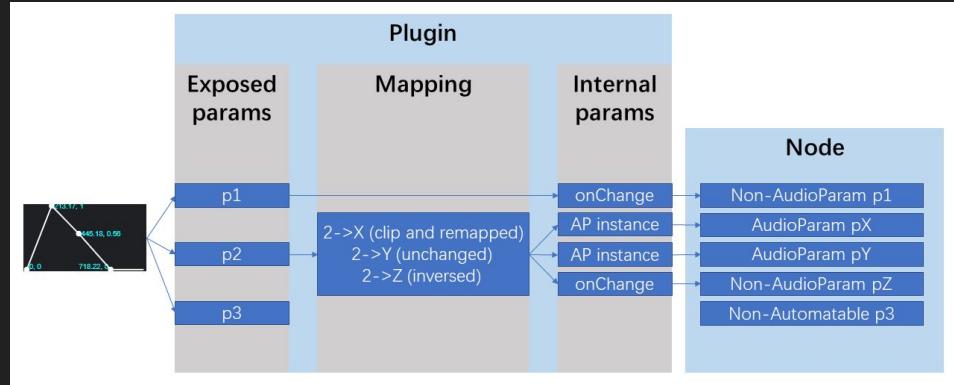
API (Standard)	SDK (Implementation/Tools)
<ul style="list-style-type: none">- Defines required methods- Abstract classes	<ul style="list-style-type: none">- Reference API implementations- Utility classes and example plugins

- Developers can choose to adapt their existing code to the API
- Others can use the SDK that implements the API (much easier), inherit classes etc.

The sdk-parammgr repository

Dedicated to plugins made of an audio graph:

- Exports a **CompositeNode** class, the plugin will be seen as a single node!
- Deals with parameter mapping and automation (figure).



3 exposed parameters

Dozens of internal parameters

The wam-examples repository

github.com/webaudiomodules/wam-examples

Installation

1. Install dependencies

```
git submodule update --init --recursive --remote
yarn install
```
2. Initialize monorepo dependencies using lerna

```
yarn lerna bootstrap
```

Getting started

Commands

Available scripts :

- `yarn build`: builds sdk and plugins (you may also use scripts `build:sdk`, `build:pingpongdelay` etc.)
- `yarn start`: starts the host example (for development only). Open <http://localhost:1234>
- `yarn clean`: deletes built code

(other scripts can be found in `/package.json`)

Create a plugin

For this example we will create a very basic plugin named simplegain

1. Create a plugin package

Create a package under packages directory using [lerna create](#)

Example WAM Host

NEW: [See WAM 2.0 API docs](#)

NEW: [See WAM 2.0 Parameter Manager docs](#)

0:00 / 2:56

Select MIDI input device [Select...](#)

Select live input device [Par défaut - MacBook Pro Microphone \(Built-in\)](#)

Live input: NOT ACTIVATED, click to toggle on/off

Enter any WAM Plugin URL [..quadrafuzz_without_builder/src/index.js](#)

LOAD PLUGIN

Midi

- Simple MIDI Keyboard
- Midi Virtual keyboard No Sound (just emits events)
- Random note generator
- Simple Transport
- MIDI Sequencer

Pure JavaScript

- Quadrafuzz
- Quadrafuzz without builder
- Disto Machine without builder
- PingPongDelay
- Graphic Equaliser (pureJS)
- WAM Example (pureJS)
- TinySynth, a GM synth without GUI. Accepts program changes, volume change, notes...
- Wam Event Viewer
- WebMIDI Output

SAVE STATE RESTORE STATE

```
instance.descriptor : {"name": "Quadrafuzz (No builder)", "vendor": "WebAudioModule", "description": "Quadrafuzz written in native WebAudio nodes", "version": "1.0.0", "apiVersion": "2.0.0", "thumbnail": "Screenshot.png"}, {"isInstrument": false, "website": "", "hasAudioInput": true, "hasAudioOutput": true}, gui.properties.dataWidth.value : undefined, (you should define get properties in Gui.js), gui.properties.dataHeight.value : undefined, (you should define get properties in Gui.js), instance.audioNode.getParameterInfo() : lowGain : {"id": "lowGain", "label": "lowGain", "type": "float", "defaultValue": 0.6, "[]}, {"units": ""}, midLowGain : {"id": "midLowGain", "label": "midLowGain", "type": "float", "defaultValue": [1], "units": ""}, Add Automation... Send MIDI Note
```



WAM2 step by step tutorials

Several tutorials are available at <https://wam-examples.vidalmazuy.fr/>

wam-examples.vidalmazuy.fr/example3/index.html#demo

Example : Simple JavaScript Web Audio Modules 2.0 Processor

Guitar Song : <https://www.chosic.com/download-audio/29514/>

Start Loop

Specifications

Prerequisites

Main script

Initialization

Audio node connection

Audio processor

Conclusion

Example 2

Demo

Specifications

Prerequisites

Maketile compiling Emscripten

C++ File

Custom Audio Node

Instantiate Web Assembly

Initialize the heaps

Conclusion

Example 3

Demo

Specifications

Prerequisites

Web Audio Modules

WAM SDK

Instantiate plugins

Custom WAM processor

Conclusion

Specifications :

In this example, we will use a simple JavaScript processor, to keep the code simple. But of course, you can use the Web Assembly that we've seen in the example 2. Before jumping into the code, be sure to check the [Web Audio Modules API](#) first.

wam-examples.vidalmazuy.fr/example5/index.html#demo

Example : Simple MIDI Keyboard

OBXD VIRTUAL ANALOG SYNTHESIZER

MASTER

OSCILLATORS

MIX

FILTER

AMPLIFIER ENVELOPE

MODULATION

GLOBAL

CONTROL

VOICE VARIATION

Specifications :

Thanks to WAM, we will easily host a simple MIDI virtual keyboard and use the OBXD synthesizer. See the MIDI section in the [wiki](#) of the Web Audio Modules.

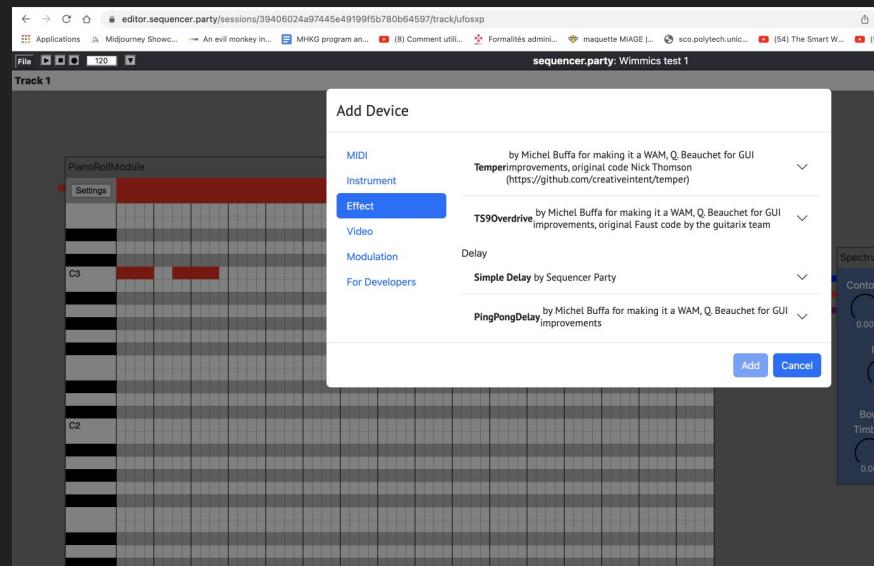
Prerequisites :

To make our host communicates with the plugin, we will use the [SDK](#) of Web Audio Modules. You can either download the full source and build it on your own, or use the pre-build version.

Also, check the wam-community repository
<https://github.com/boourns/wam-community>

Used by the community to publish and share “ready to use” plugins!

- Remember that a plugin is just a URI!
- **Several dozens of plugins available,** 99% also available with source code to study (github.com/boourns/burns-audio-wam)
- Cover all classic effects, proposes some instruments and utility plugins.
- All plugins available in the <https://sequencer.party> host.





Functional
Audio
Stream

Build a WebAssembly WAM in seconds with FAUST DSL

FAUST: a DSL for DSP programming, born in 2002 at GRAME-CNCM, France

Used in artistic productions, education and research, open source projects and commercial applications.

Faust offers end-users a high-level alternative to C/C++ to develop audio applications for a large variety of platforms.

The role of the Faust compiler is to synthesize the most efficient implementations for the target language (C, C++, LLVM, **WebAssembly**, etc.).

Online doc / tutorial so that you can experiment yourself, create, build GUI, export WAM2 plugins directly from the **FAUST online IDE**.

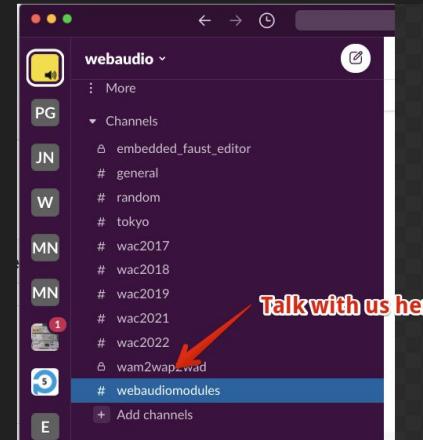
Conclusion / Perspectives

The WAM standard is stable now and comes with many examples.

**OUR MAIN CONCERN NOW: get inputs from developers
regarding barriers to entry / adoption!**

Things that will come soon:

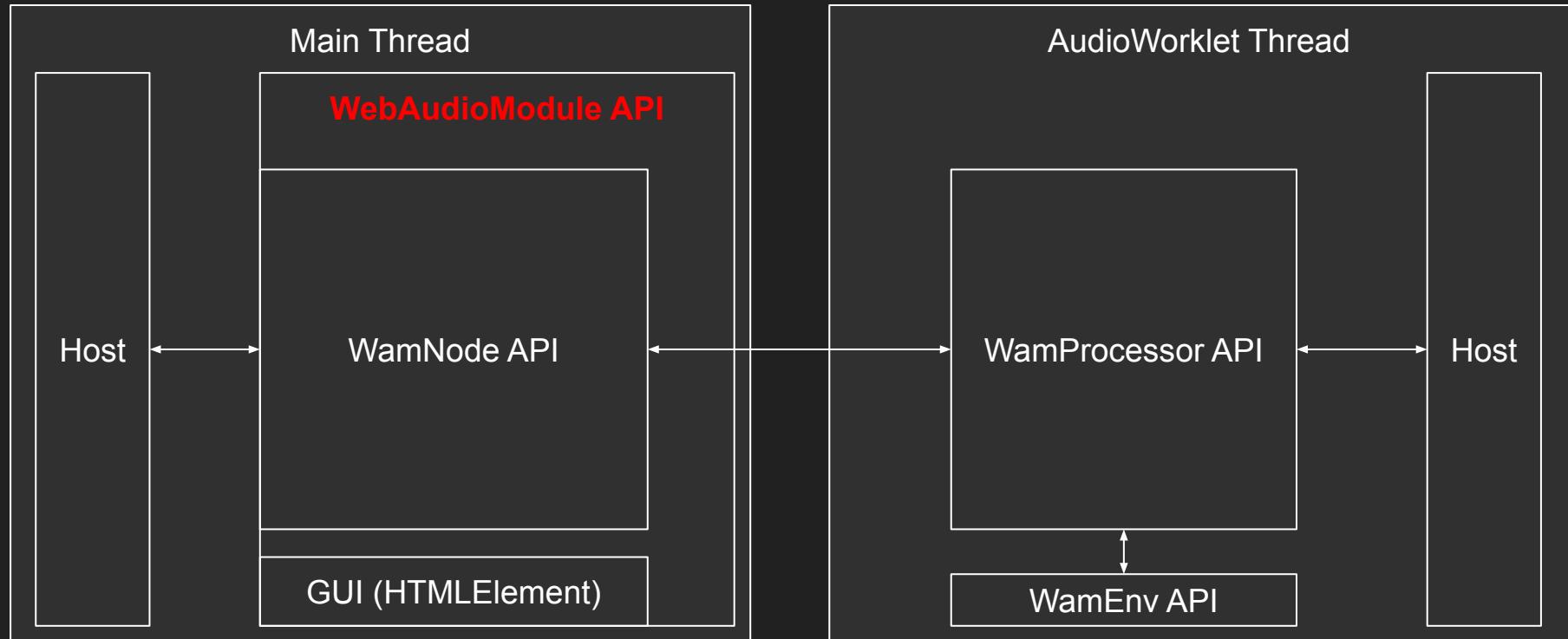
- The wam-community repo is growing :-)
- The WAM SDK has been extended to support 3D WebGL/GLSL/Video extensions
- A WAM based DAW is under development and will be open source
- More examples using WASM, in particular C++/WASM
- Remote plugin server with API



Join us on slack WebAudio channel / #webaudiomodules!

Interact with a host, be it in main or audio thread

1 - The WebAudioModule API



A plugin = instance of a WAM = “a WAM”

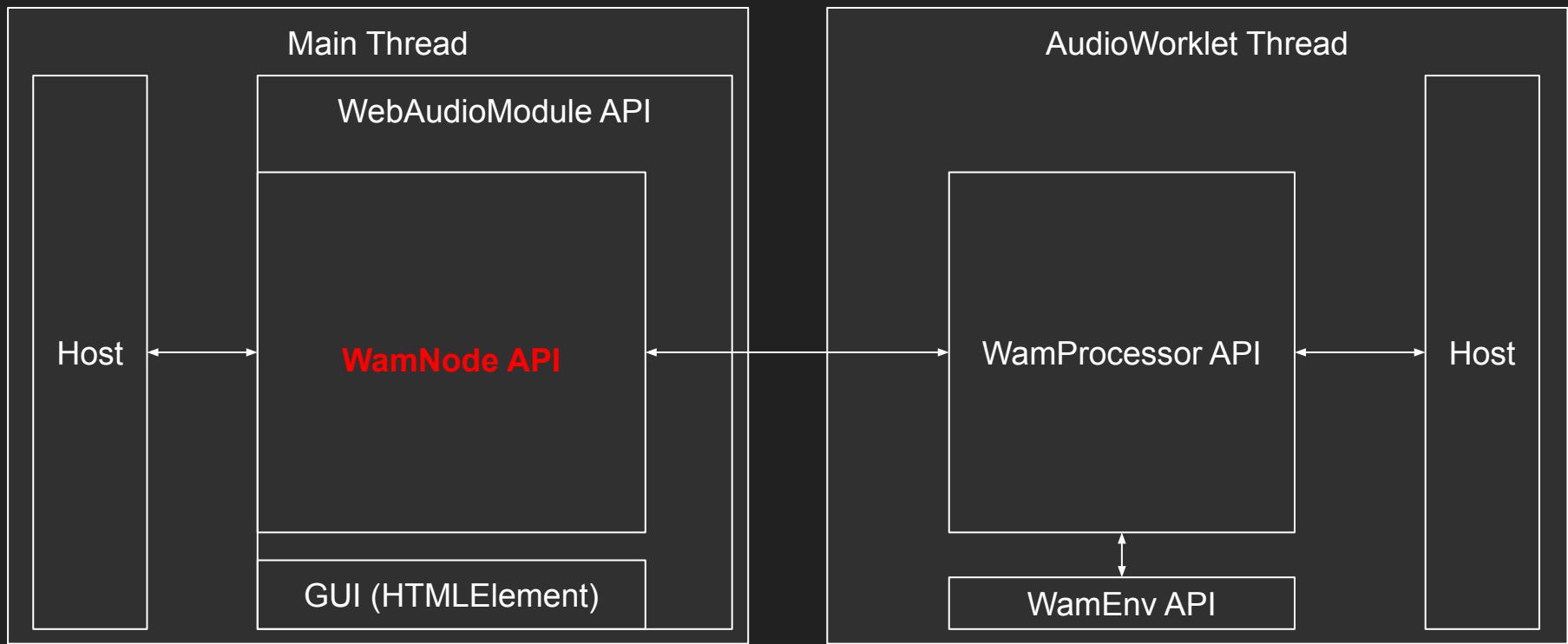
API - WebAudioModule

```
export interface WebAudioModule<Node extends WamNode = WamNode> {
    /**
     * This async method must be redefined to get `AudioNode` that
     * will connected to the host
     * It can be any object that extends `AudioNode` and implements `WamNode`
     */
    createAudioNode(initialState?: any): Promise<WamNode>;
    /**
     * The host will call this method to initialize the WAM with an initial state.
     *
     * In this method, WAM devs should call `createAudioNode()`
     * and store its return `AudioNode` to `this.audioNode`,
     * then set `initialized` to `true` to ensure that
     * the `audioNode` property is available after initialized.
     *
     * These two behaviors are implemented by default in the SDK.
     *
     * The WAM devs can also fetch and preload the GUI Element in while initializing.
     */
    initialize(state?: any): Promise<WebAudioModule>;
    /**
     * Redefine this method to get the WAM's GUI as an HTML `Element`.
     */
    createGui(): Promise<Element>; ←
    /**
     * Clean up an element previously returned by `createGui` */
    destroyGui(gui: Element): void
}
```

The code block shows the definition of the `WebAudioModule` interface. Several properties and methods are annotated with red arrows pointing to specific parts of the interface definition:

- `audioNode: Node;` → points to the `audioNode` property.
- `moduleId: string;` → points to the `moduleId` property.
- `instanceId: string;` → points to the `instanceId` property.
- `createAudioNode(initialState?: any): Promise<WamNode>;` → points to the `createAudioNode` method.
- `createGui(): Promise<Element>;` → points to the `createGui` method.

2 - The WamNode API



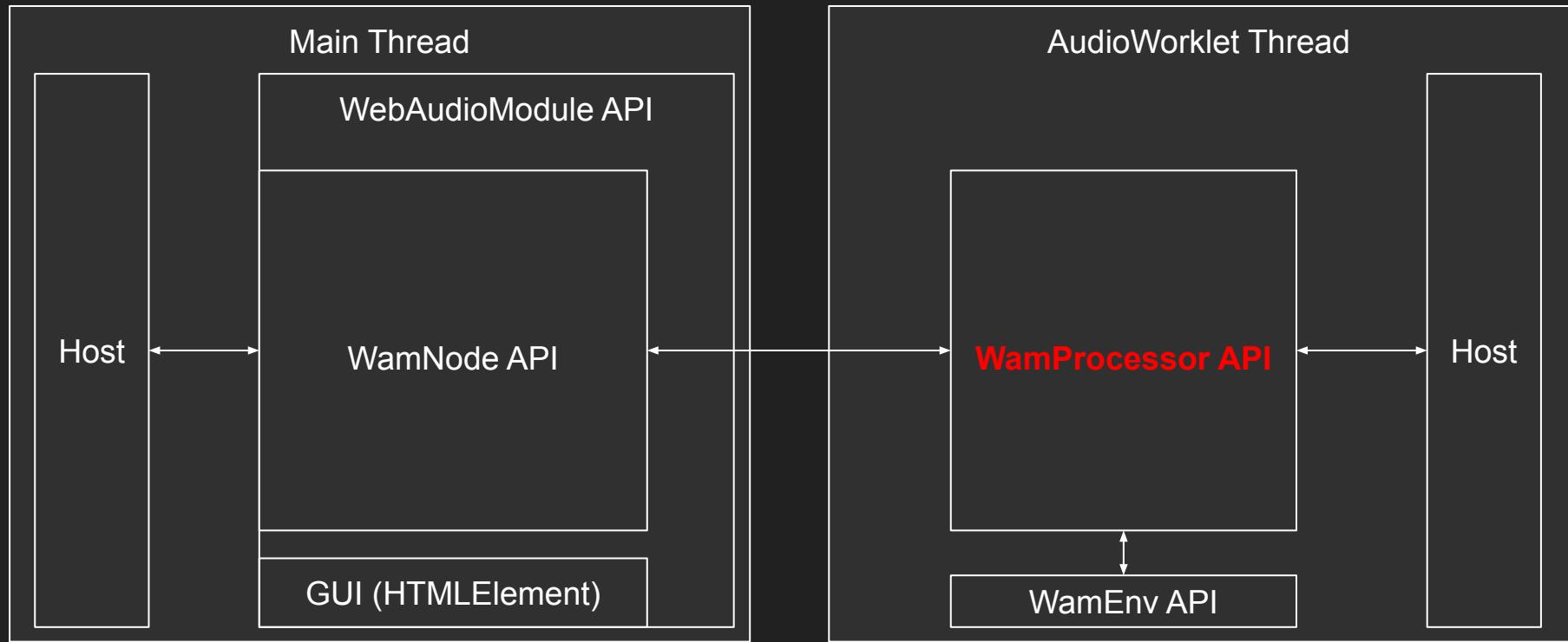
A WAM contains a WamNode, here is the API

```
export interface WamNode extends AudioNode, Readonly<WamNodeOptions> {

    readonly module: WebAudioModule;

    /** Get parameter info for the specified parameter ids, or omit argument to get info for all parameters. */
    getParameterInfo(...parameterIdQuery: string[]): Promise<WamParameterInfoMap>;
    /** Get parameter values for the specified parameter ids, or omit argument to get values for all parameters. */
    getParameterValues(normalized?: boolean, ...parameterIdQuery: string[]): Promise<WamParameterDataMap>;
    /** Set parameter values for the specified parameter ids. */
    setParameterValues(parameterValues: WamParameterDataMap): Promise<void>;
    /** Returns an object (such as JSON or a serialized blob) that can be used to restore the WAM's state. */
    getState(): Promise<any>;
    setState(state: any): Promise<void>;
    /** Compensation delay hint in samples */
    getCompensationDelay(): Promise<number>;
    /** Register a callback function so it will be called when matching events are processed. */
    addEventListener<K extends keyof WamEventMap>(type: K, listener: (this: this, ev: CustomEvent<WamEventMap[K]>) => any, options?: boolean | AddEventListenerOptions): void;
    addEventListener(type: string, listener: (this: this, ev: CustomEvent) => any, options?: boolean | AddEventListenerOptions): void;
    addEventListener(type: string, listener: EventListenerOrEventListenerObject, options?: boolean | AddEventListenerOptions): void;
    /** Deregister a callback function so it will no longer be called when matching events are processed. */
    removeEventListener<K extends keyof WamEventMap>(type: K, listener: (this: this, ev: CustomEvent<WamEventMap[K]>) => any, options?: boolean | EventListenerOptions): void;
    removeEventListener(type: string, listener: (this: this, ev: CustomEvent) => any, options?: boolean | AddEventListenerOptions): void;
    removeEventListener(type: string, listener: EventListenerOrEventListenerObject, options?: boolean | EventListenerOptions): void;
    /** Schedule a WamEvent. Listeners will be triggered when the event is processed. */
    scheduleEvents(...event: WamEvent[]): void;
    /** Clear all pending WamEvents. */
    clearEvents(): void;
    /** Connect an event output stream to another WAM. If no output index is given, assume output 0. */
    connectEvents(to: WamNode, output?: number): void;
    /** Disconnect an event output stream from another WAM. If no arguments are given, all event streams will be disconnected. */
    disconnectEvents(to?: WamNode, output?: number): void;
    /** Stop processing and remove the node from the graph. */
    destroy(): void;
}
```

3 - The WamProcessor API



WAMs also have an explicit/implicit WamProcessor API - WamProcessor

```
export interface WamProcessor extends AudioWorkletProcessor {  
  readonly moduleId: string;  
  readonly instanceId: string;  
  /** Compensation delay hint in seconds. */  
  getCompensationDelay(): number;  
  /** Schedule a WamEvent. Listeners will be triggered when the event is processed. */  
  scheduleEvents(...event: WamEvent[]): void;  
  /** Schedule events for all the downstream WAMs */  
  emitEvents(...events: WamEvent[]): void;  
  /** Clear all pending WamEvents. */  
  clearEvents(): void;  
  /** Process a block of samples. Note that `parameters` argument is ignored. */  
  process(inputs: Float32Array[][], outputs: Float32Array[][], parameters: Record<string, Float32Array>): boolean;  
  /** Stop processing and remove the node from the WAM event graph. */  
  destroy(): void;  
}
```

WamEnv and WamGroup: manage plugin chains

Send events downstream to a list of chained plugins..., manage group (states) etc.

