



Playing with Web Audio Modules: the Concept of Distributive Creation and Live Performance

Michel Buffa, Jean-François Trubert

► To cite this version:

Michel Buffa, Jean-François Trubert. Playing with Web Audio Modules: the Concept of Distributive Creation and Live Performance. Digital Week Conference 2023, University Côte d'Azur, Jun 2023, Nice, France. hal-04337545

HAL Id: hal-04337545

<https://inria.hal.science/hal-04337545v1>

Submitted on 12 Dec 2023

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.

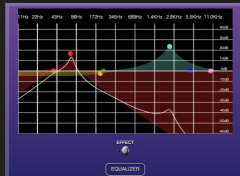
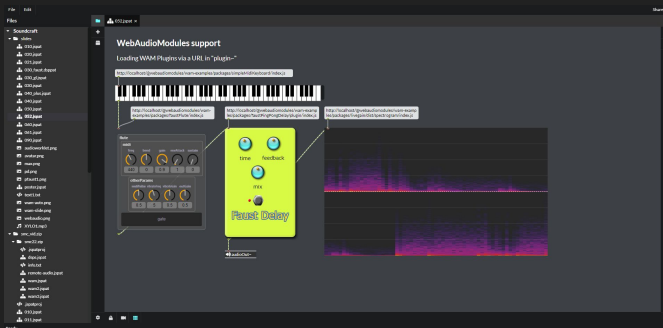


Distributed under a Creative Commons Attribution 4.0 International License

Playing with Web Audio Modules: the Concept of Distributive Creation and Live Performance

Michel Buffa, Jean-François Trubert 2023

@micbuffa, (michel.buffa, jean-francois.trubert)@univ-cotedazur.fr



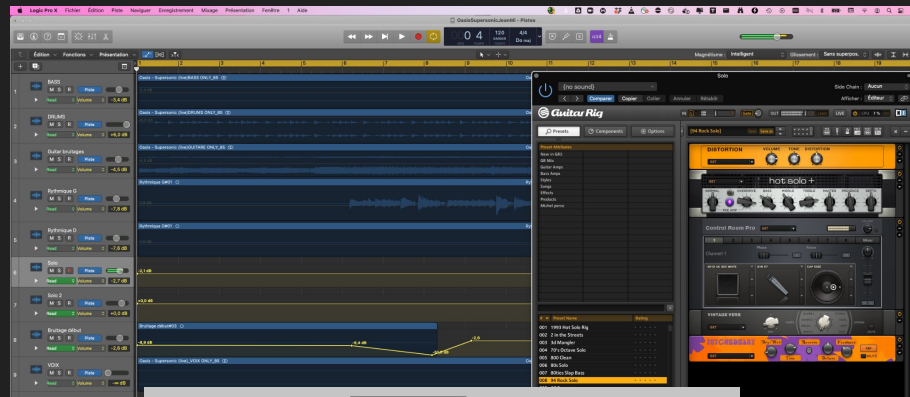
Who am I? What is our group?

- 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 Sintès, Guillaume Etevenard, GRAME friends...



The Electronic Music landscape is organized with the host/plugin paradigm

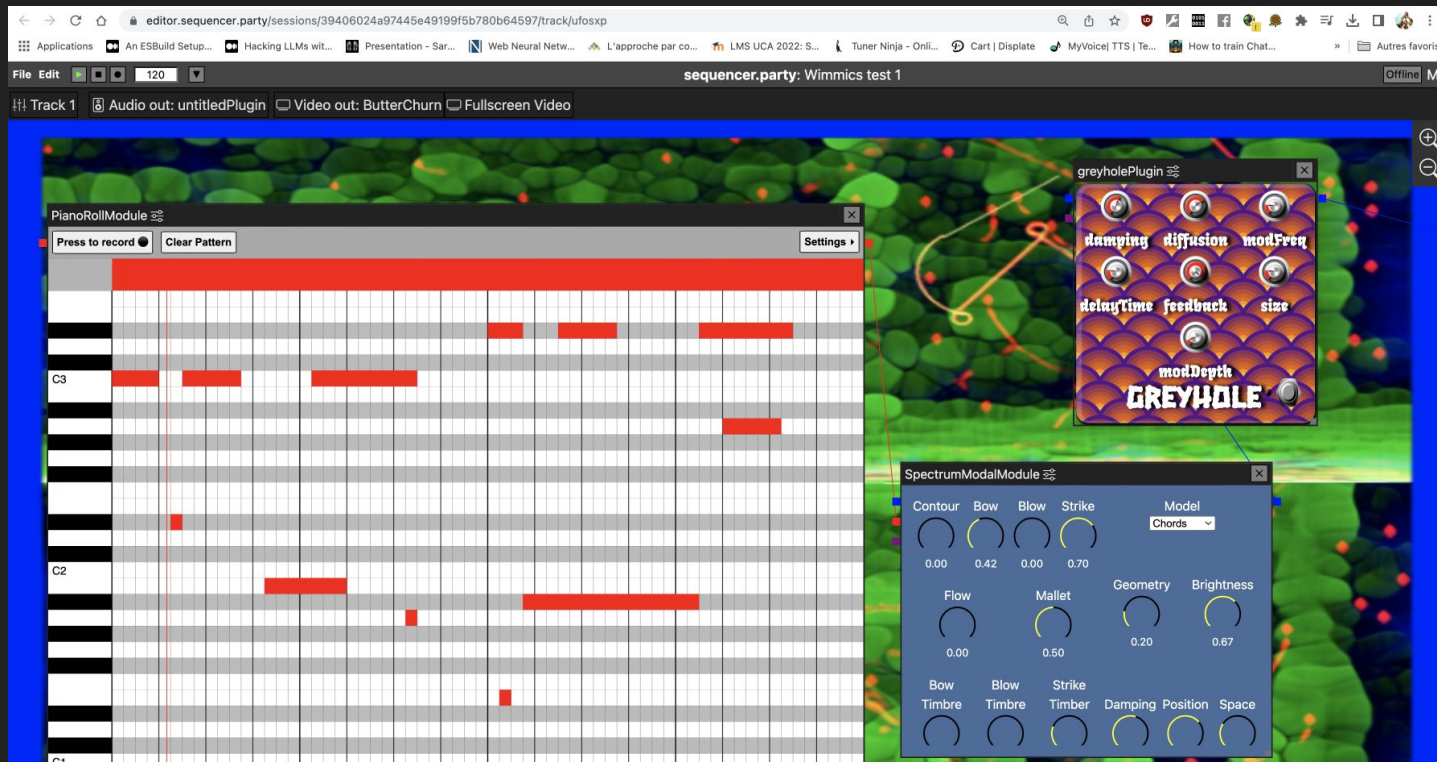
- Market ruled by commercial actors and non Web-based software
- **Articulated around DAW hosts** (Digital Audio Workstations) : Cubase (Steinberg), Logic Pro, Ableton Live, etc.
- **And plugins**: DAWs are “hosts” for plugins (effects, instruments)
- Many closed, commercial standards VST, Apple Audio Units, AVID AAX, etc.
 - Literally thousands of plugins have been developed in C/C++, meta standard exists (JUCE, iPlug2...)



Covid pandemic: make music
together from home

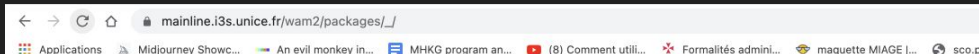
Web Audio Modules: a collaborative
Web-based solution
(and open source)

Real-time, collaborative, music creation across borders,
stream on YT, twitch, etc. <https://sequencer.party/>





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



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

/faust/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
- Dist 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)
- BigMuff Fuzz (generated by faust IDE)
- faust Flute MIDI

Typescript

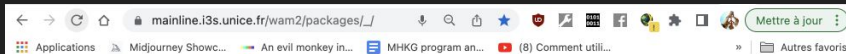
- LiveGain
- Oscilloscope
- Spectroscope
- Spectrogram

Csound

- Csound PitchShifter



```
SAVE STATE | RESTORE STATE
+ instance.descriptor : { "name": "OBXD", "vendor": "Jari Kleimola 2017-2020 (jari@webaudiomodules.org)", "description": "", "version": "1.0.0", "apiVersion": 1, "hasInstrument": 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() :
```



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

/faust/Flute/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
- Dist 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)
- BigMuff Fuzz (generated by faust IDE)
- faust Flute MIDI

Typescript



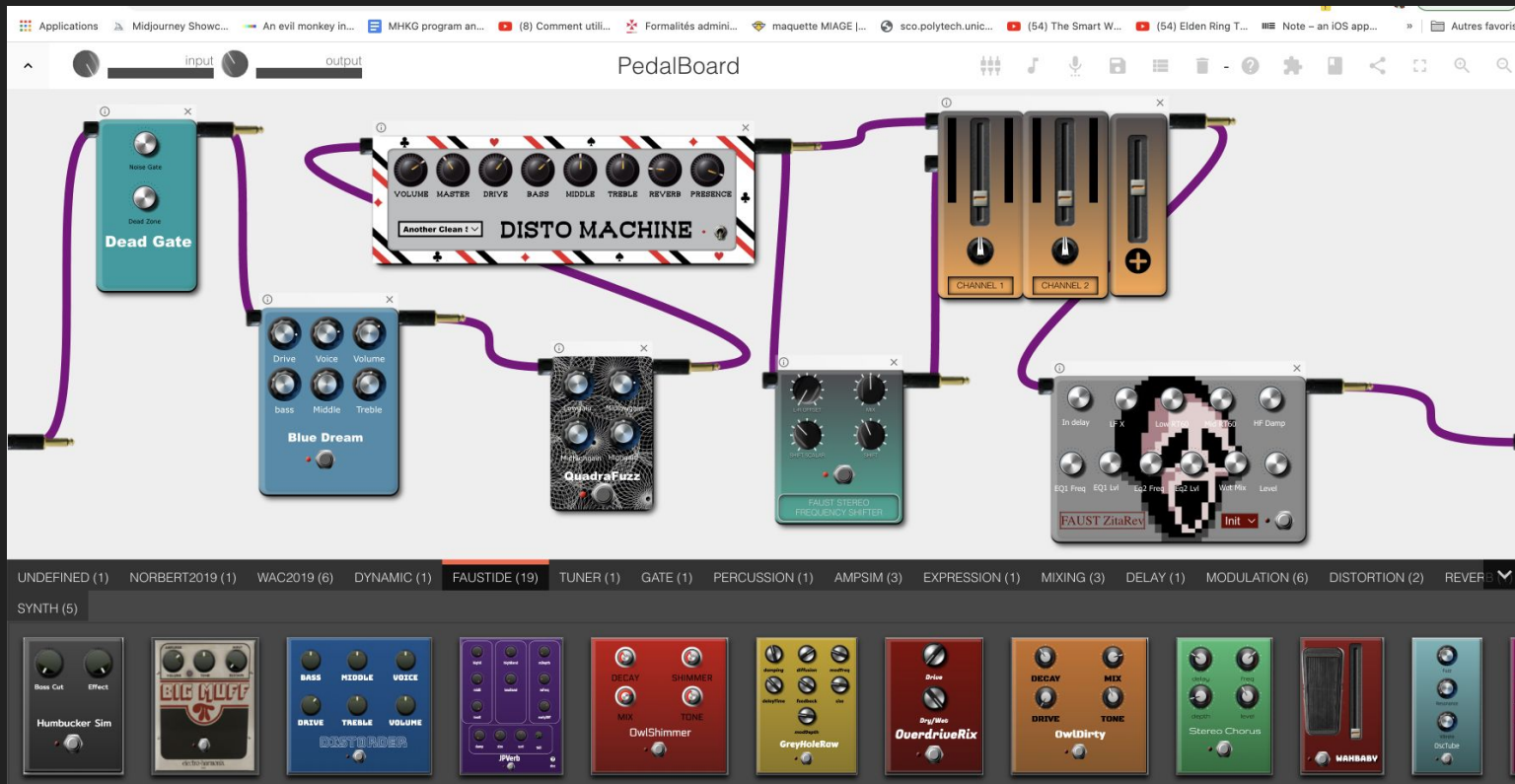
Example: a WAM sampler based on freesound.org



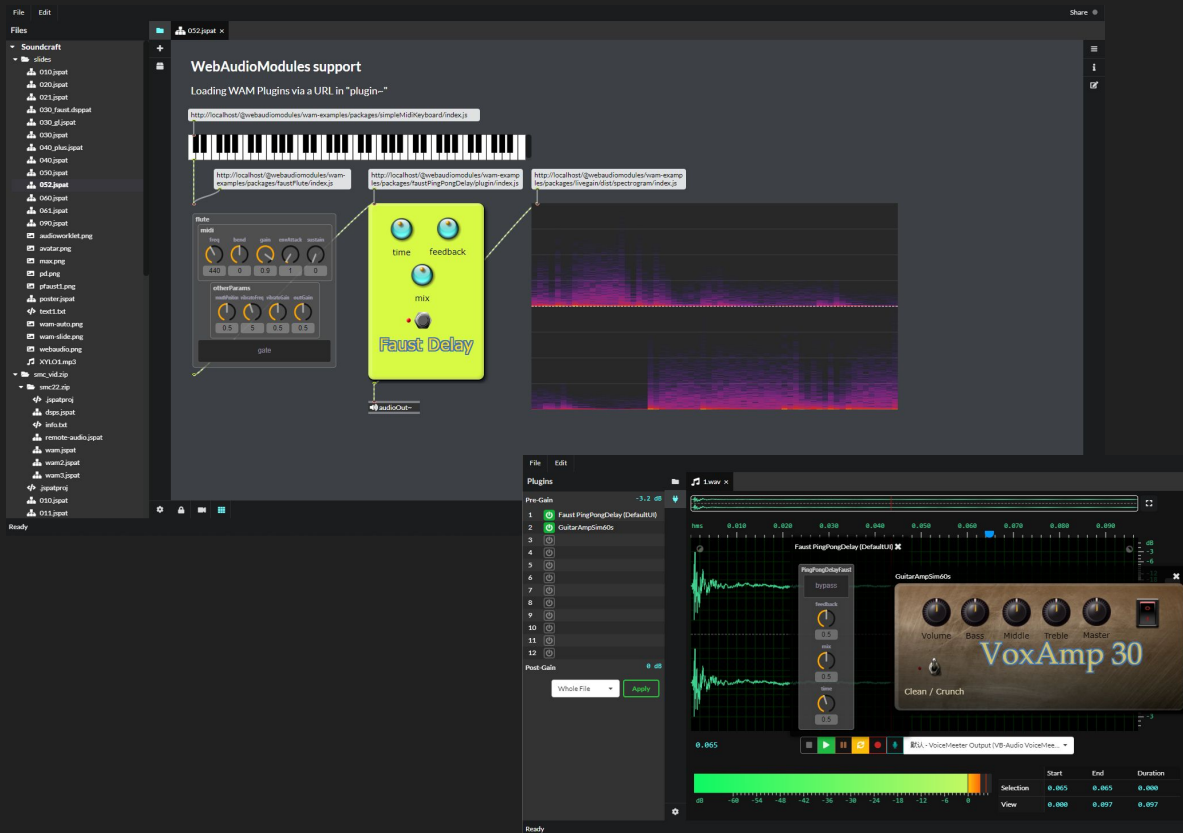
WAMs in hosts... WIP open source DAW (100% WAM-based)



WAMs in hosts... here the WASABI pedalboard



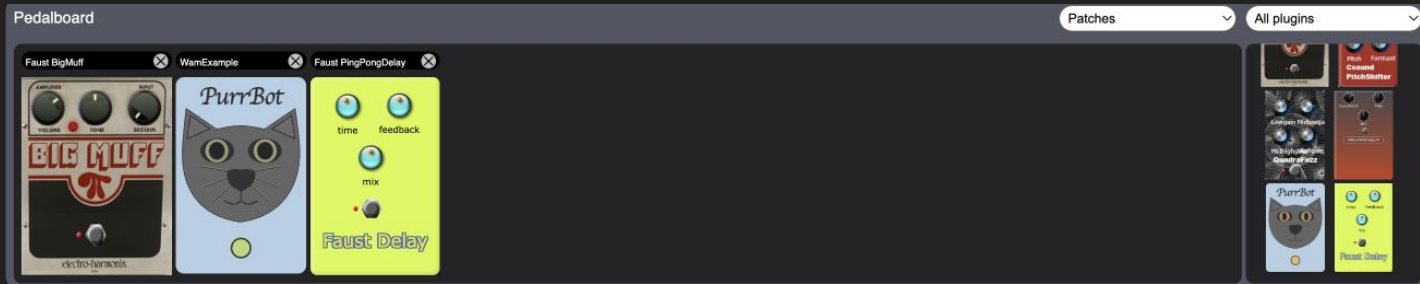
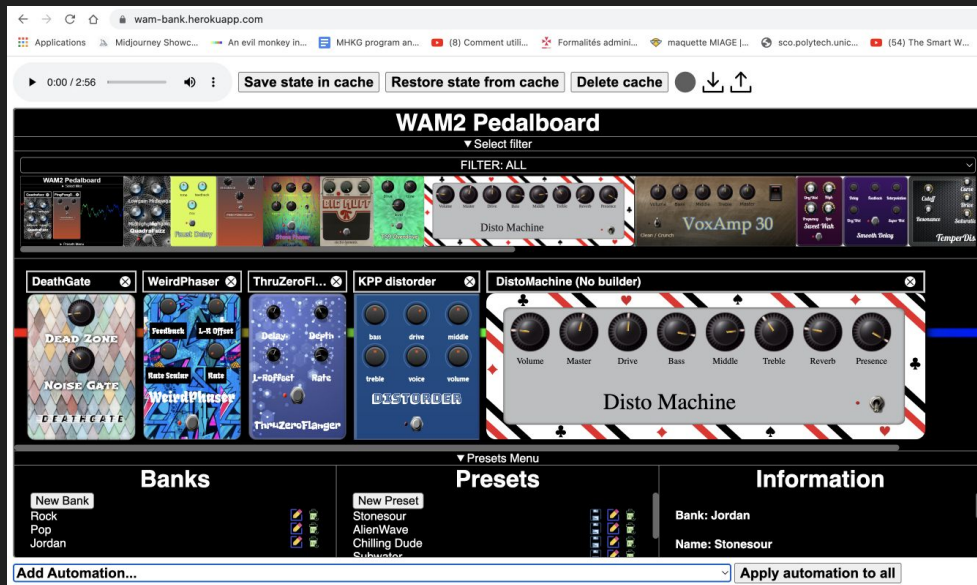
WAMs in hosts... JSPatcher, aka Max MSP in the browser (<https://github.com/Fr0stbyteR/jspatcher>)



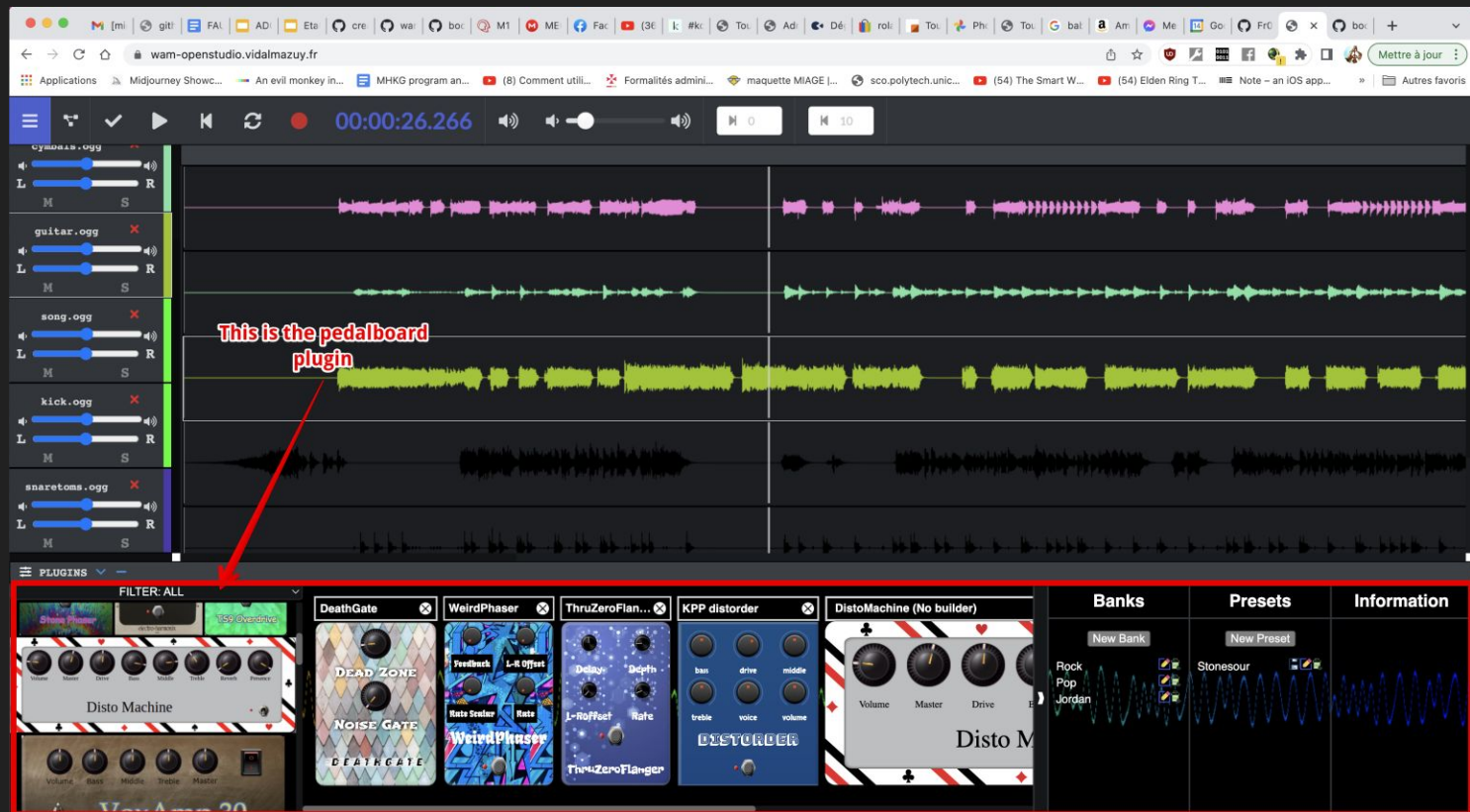
WAMs in plugins that acts as hosts (i.e pedalboard)

These ones
are open
source

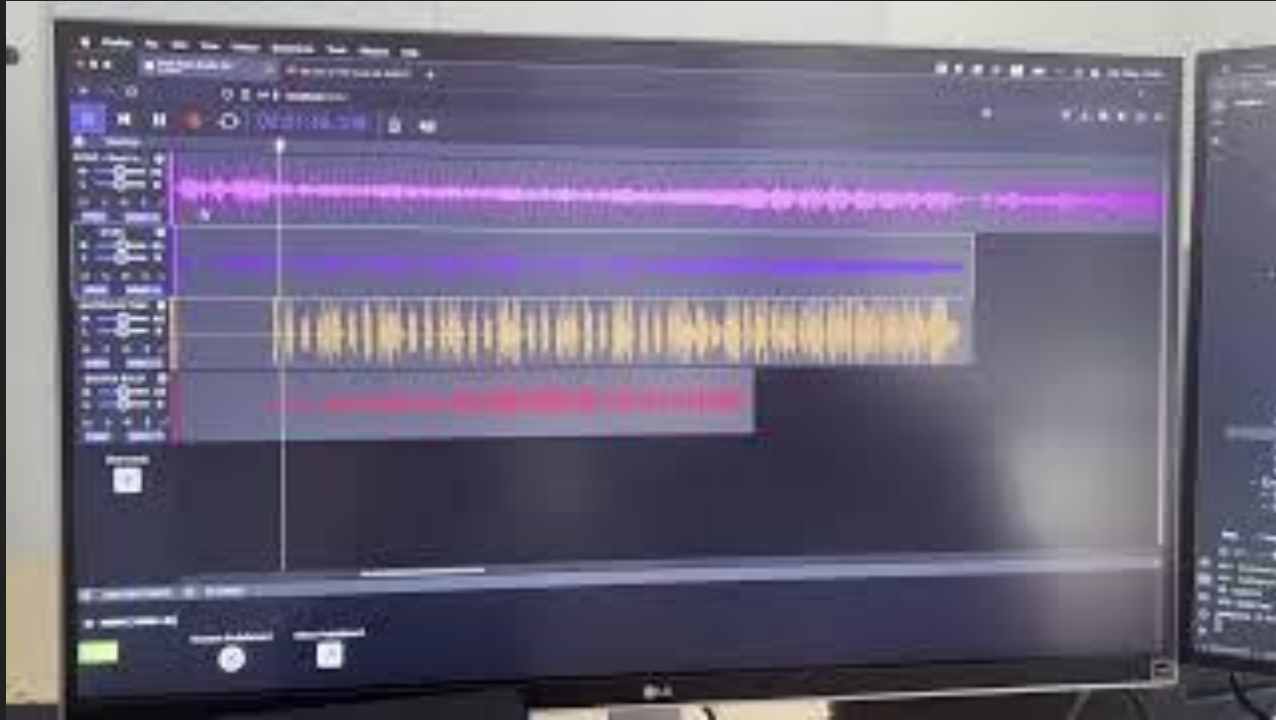
(from [wam-examples](#)
and [wam-community](#)
github repositories)



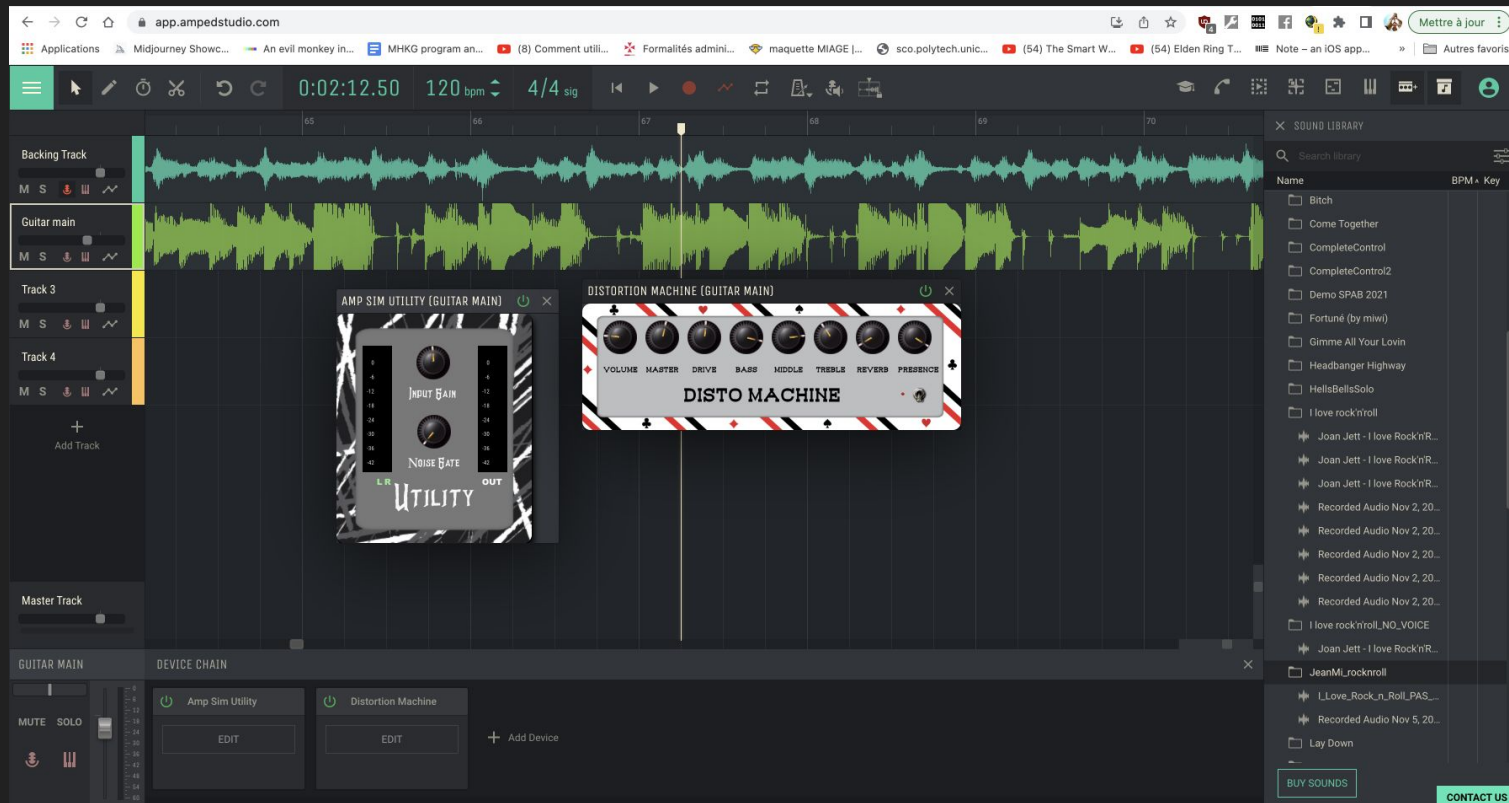
... and loaded in a DAW!



A small projet from recording to mixing



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



A few reminders before proceeding...

Concepts de l'API WebAudio

Opérations audio dans un Audio Context

```
let ctx = new AudioContext();
```

Conception modulaire :

- Opérations audio : **audio nodes**
...qui forment un **graphe audio**

```
let osc1 = ctx.createOscillator();  
osc1.frequency.value = 440;  
let gain1 = ctx.createGain();  
gain1.gain.value = 0.1;
```

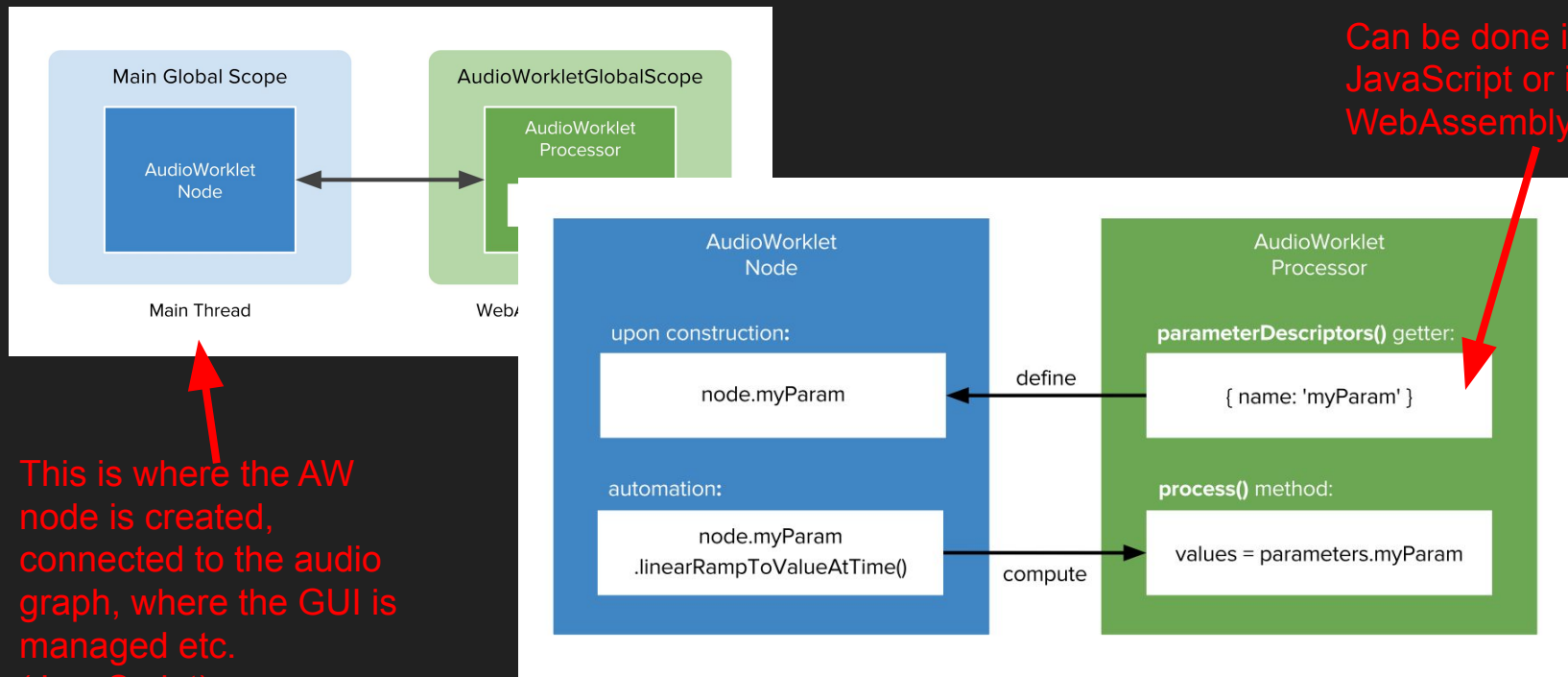
```
osc1.connect(gain1).connect(ctx.destination);
```



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

This is where custom DSP processing is done!

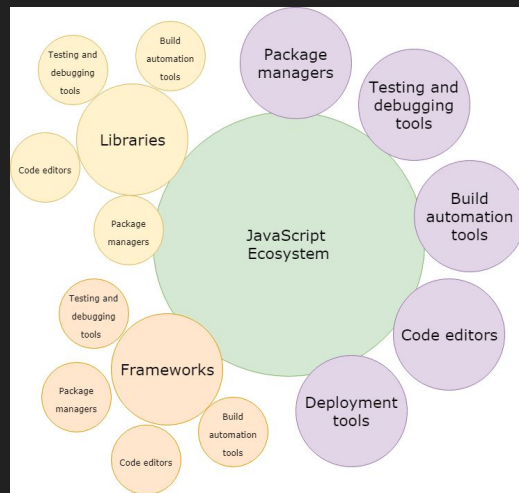
Can be done in JavaScript or in a WebAssembly module!



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.



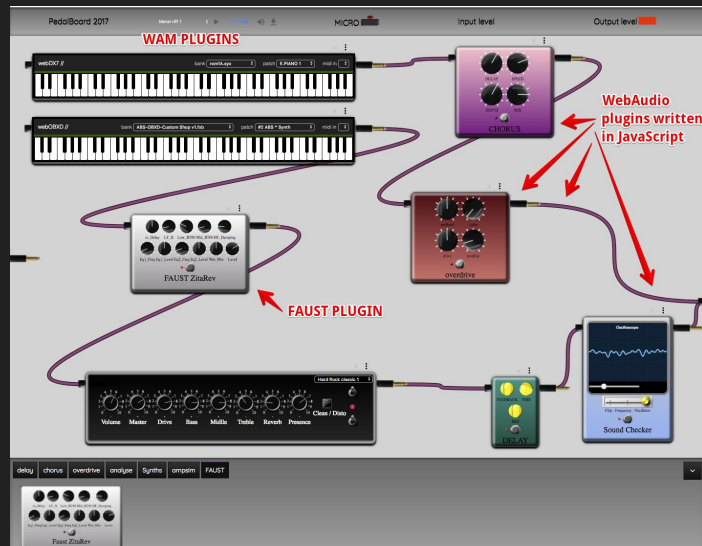
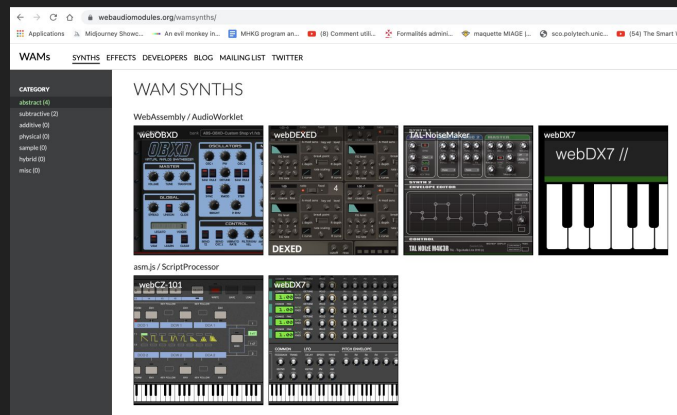
How to combine all this?

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.)

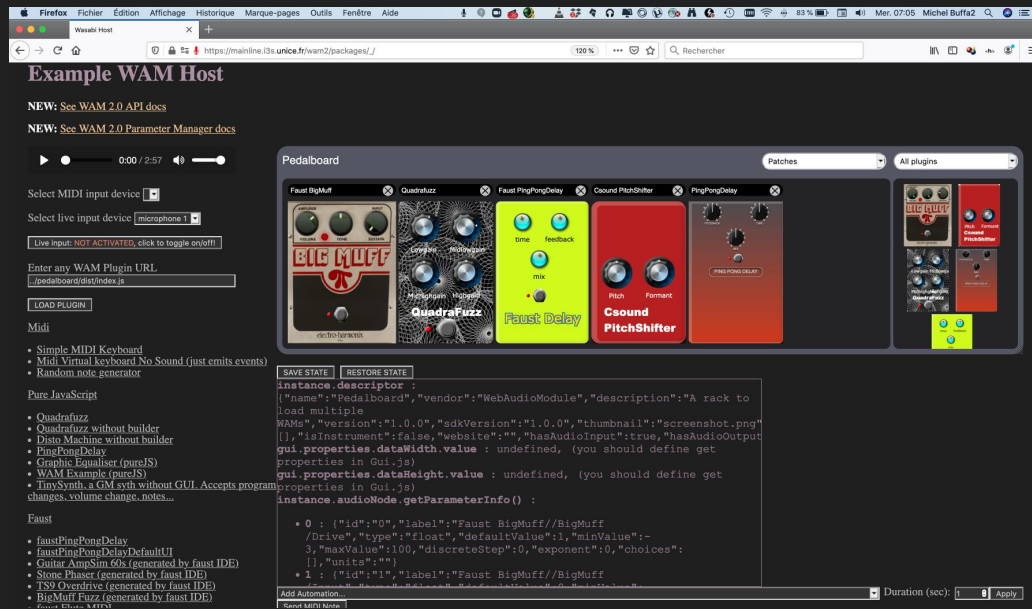
- Please web developers,
- Support DSLs like FAUST, other improvements...



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

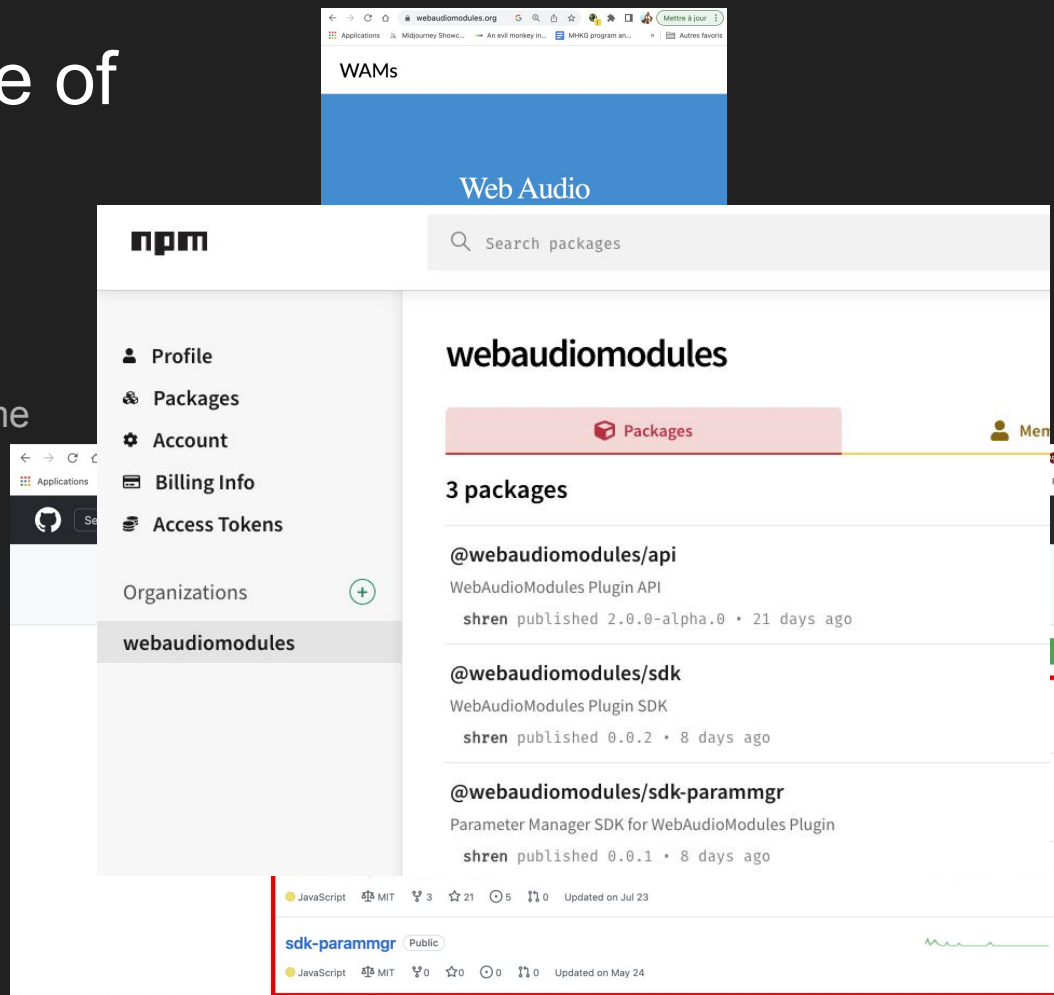


How to start with WAMs!

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

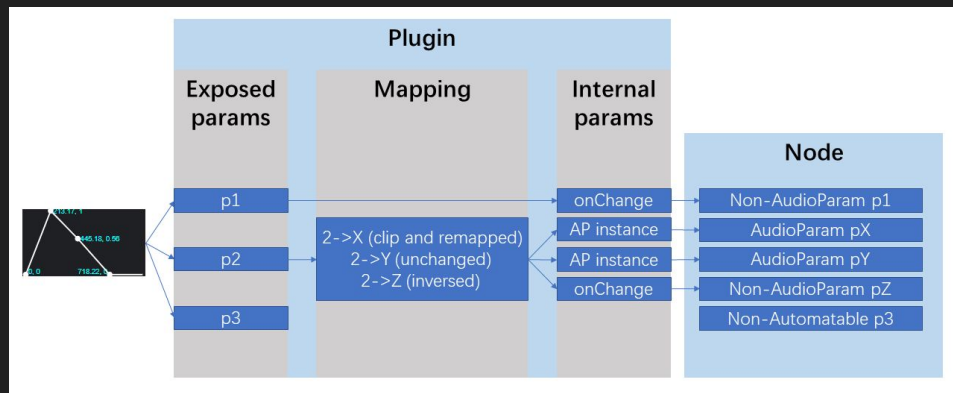
API (Standard)	SDK (Implementation/Tools)
<ul style="list-style-type: none">- Defines required methods- Abstract classes	<ul style="list-style-type: none">- Reference API implementations- Utility classes and example plugins

- 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.

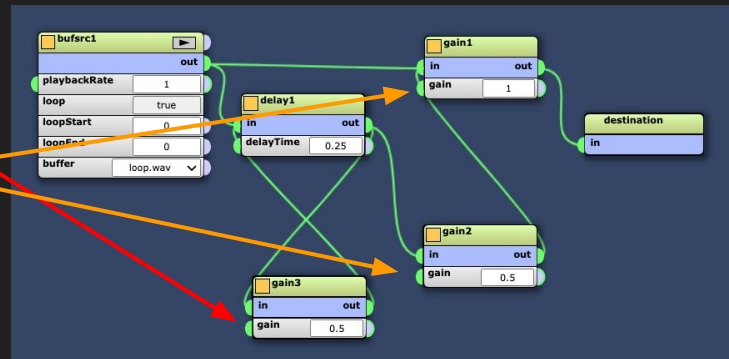
The sdk-parammgr repository

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/webaudiomodels/wam-examples

Applications Midjourney Showc... An evil monkey in... MHKG program an... (8) Comment utili... Formalités admini... maquette MIAGE [...] sco.polytech.unic... (54) Th

README.md

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](#)

mainline.i3s.unice.fr/wam2/packages/_/

Applications Midjourney Showc... An evil monkey in... MHKG program an... (8) Comment utili... Formalités admini... maquette MIAGE [...] sco.polytech.unic... (54) Th

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
./quadr fuzz_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

- Quadr fuzz
- Quadr fuzz 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



```
* instance.descriptor : { "name": "Quadr fuzz (No builder)", "vendor": "WebAudioModule", "description": "Quadr fuzz 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": 0.1, "units": "" } } }
```

SAVE STATE **RESTORE STATE**

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


Applications Midjourney Showc... An evil monkey in... MHKG program an... (8) Comment utili... Formalités admini... maquette MIAGE [...] sco.polytech.univ... (54) The Sm

WAM Web Audio Modules



- Example 1
 - Demo
 - Specifications
 - Prerequisites
 - Index.html
 - Operable audio buffer
 - Main script
 - Initialization
 - Audio node connection
 - Audio processor
 - Conclusion
- Example 2
 - Demo
 - Specifications
 - Prerequisites
 - Makefile compiling Emcripten
 - 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

Example : Simple JavaScript Web Audio Modules 2.0 Processor

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



Start ☐ Loop ☒



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/example6/index.html#demo

Applications Midjourney Showc... An evil monkey in... MHKG program an... (8) Comment utili... Formalités admini... maquette MIAGE [...] sco.polytec

WAM Web Audio Modules

- Example 1
 - Demo
 - Specifications
 - Prerequisites
 - Index.html
 - Operable audio buffer
 - Main script
 - Initialization
 - Audio node connection
 - Audio processor
 - Conclusion
- Example 2
 - Demo
 - Specifications
 - Prerequisites
 - Makefile compiling Emcripten
 - 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
- Example 4
 - Demo
 - Specifications
 - Prerequisites

Example : Simple MIDI Keyboard



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 :

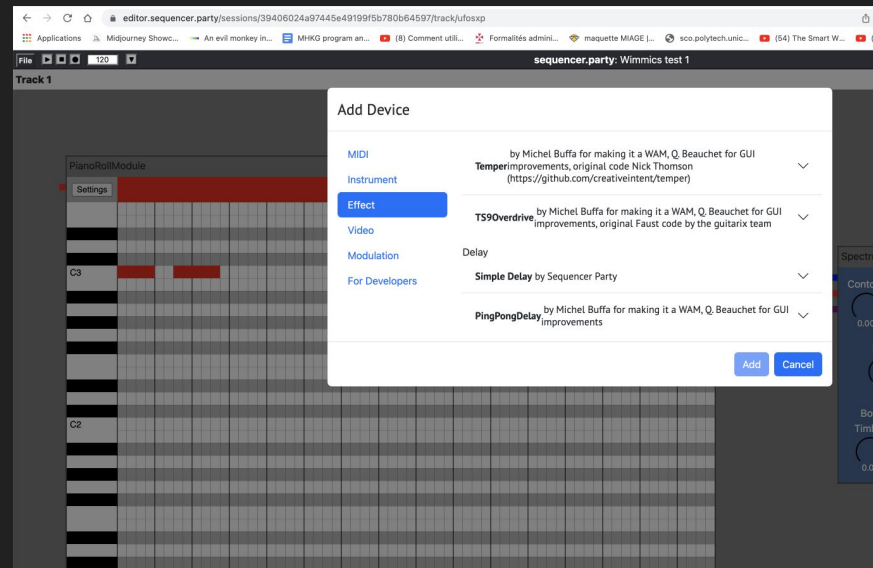
To make our host communicates with the plugin, we will use the *SDK* of Web Audio Modules. You can either download the full source and build it on your own, or use the pre-build version.

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/bourns-audio-wam)
- Cover all classic effects, proposes some instruments and utility plugins.
- All plugins available in the <https://sequencer.party> host.



Build a WebAssembly WAM in seconds with FAUST DSL



Functional
Audio
Stream

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

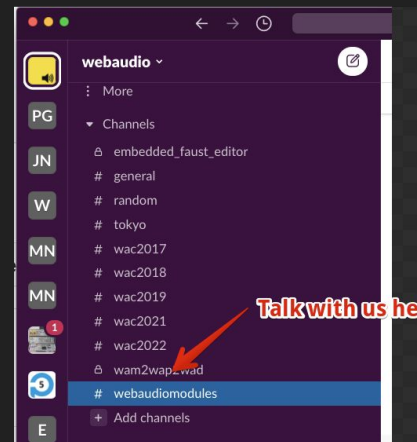
The WAM standard is stable now and comes with many examples.

OUR MAIN CONCERN NOW: grow adoption, more users, more developers!

Things that will come soon:

- The wam-community repo is growing every week :-)
- The WAM SDK [has been extended to support 3D WebGL/GLSL/Video extensions](#)
- A WAM based DAW has been released and is open source
- More examples for developers: using WASM, in particular C++/WASM
- Remote plugin server with API

Join us on slack WebAudio channel / #webaudiomodules!



- Renwick, Robin. (2012). "SOURCENODE: A NETWORK SOURCED APPROACH TO NETWORK MUSIC PERFORMANCE (NMP).hdl:2027/spo.bbp2372.2012.057.

David Kim-Boyle (2009) Network Musics: Play, Engagement and the Democratization of Performance, Contemporary Music Review, 28:4-5, 363-375, DOI: 10.1080/07494460903422198 "the strategies employed in the work of these artists have helped redefine a new aesthetics of engagement in which play, spatial and temporal dislocation are amongst the genre's defining characteristics".

digital creativity

Tanzi, D. (2005) - 'Musical objects and digital domains'. Proceedings of EMS-05 Conference. Montreal, Quebec, October 19-22, 2005

- the role of processes seems to assume more importance than their results
- the notions of non-linearity and erraticism have increasingly become part of creative processes
- digital creativity : digital objects can be modified by human-computer interaction more than ever, and that the formal connotations of a musical work may be re-tracked in an ever-changing way
- the interpretation of sonorous events has to deal with dynamism of inter-medial relationships
- relationships between musical subjectivity and the emotional indexes of sonorous space

- Schroeder, Franziska (2013). "Network[ed] Listening—Towards a De-centering of Beings". *Contemporary Music Review*. 32 (2–03): 215. doi:10.1080/07494467.2013.775807
 - fragile state of listening and de-centered kind of performative being
 - network[ed] listening posits listening as a corporeal and multi-dimensional experience that is continuously being re-shaped by technological, socio-political and cultural concerns.

The question of “DEVICE” and digital creativity

α Simondon (1958, p. 175): coupling [man-machine] occurs when a single, complete function is fulfilled by both beings. Such a possibility exists whenever a technical function involves definite self-regulation [...] In the craft, this control by means of information gathering is frequent: man, being both the motor of the tool and the perceiving subject, regulates his action according to the instantaneous partial results. The tool is both a tool and an instrument [emphasis added], i.e. a means of action extending the organs and a channel for recurrent information.

α Moles (1972): the object is situated at the crossroads of an interaction when there is a conjunction between situation and act, so it is always included in a praxeological dimension.

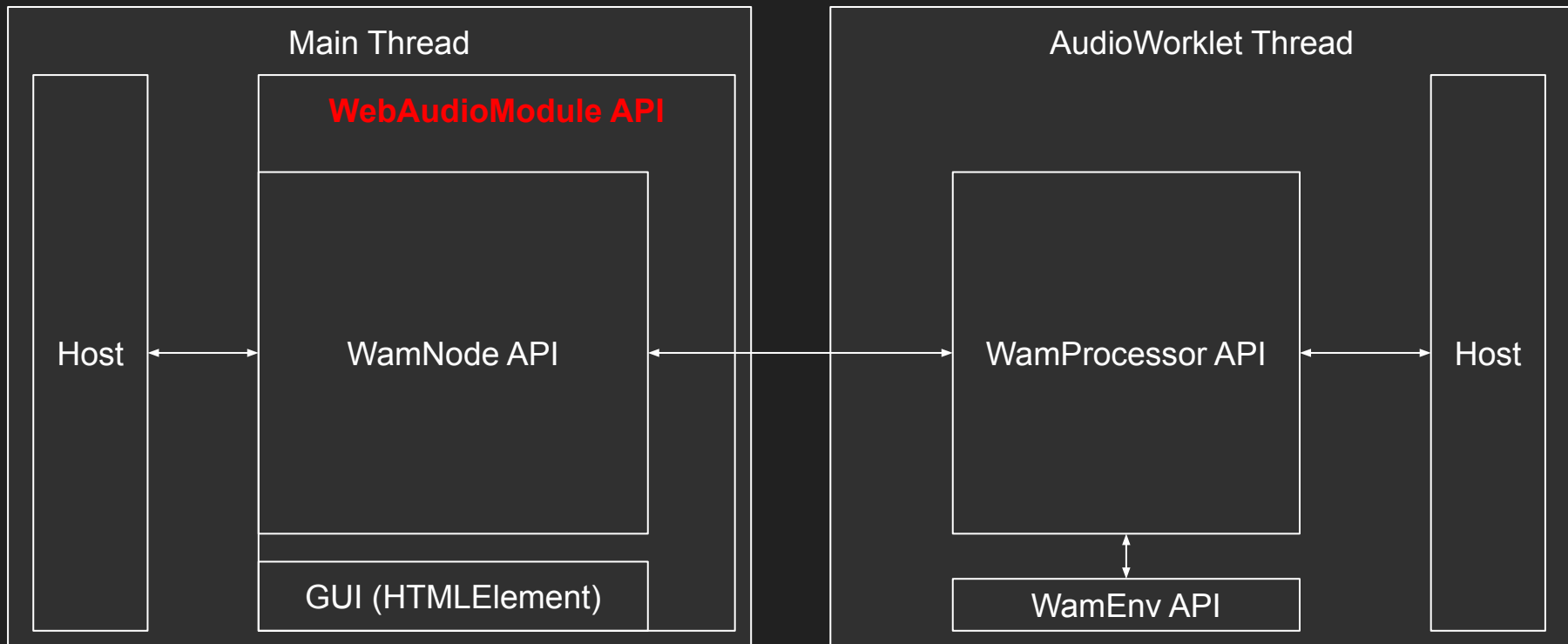
Device and creativity

Agamben (2014): "I call a device (p.31) anything that has, in one way or another, the capacity to capture, orientate, determine, intercept, model, control and ensure the gestures, conducts, opinions and discourses of living beings."

- The device ranges from the school to the pen, from the asylum to the cell phone.
- Foucault has thus shown how, in a disciplinary society, devices aim, through a series of practices and discourses, knowledge and exercises, to create docile yet free bodies that assume their identity and freedom as subjects in the very process of their subjection.
- The device is therefore, above all, a machine that produces subjectivation. Digital Creativity is a constant negotiation between the engagement in a NMP process or WAM interaction, non linear music production, and the freedom of music agency and literacy, in which serendipity could have a central role.

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

1 - The WebAudioModule API



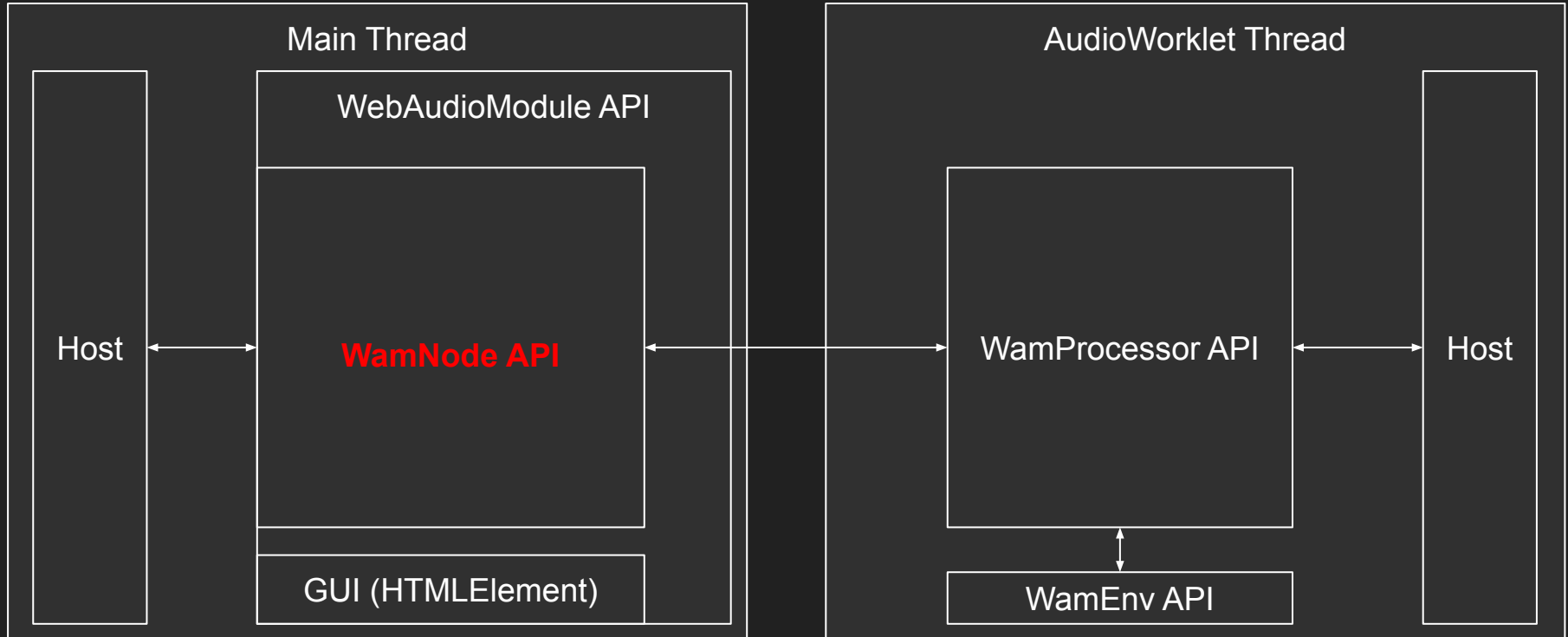
A plugin = instance of a WAM = “a WAM”

API - WebAudioModule

```
export interface WebAudioModule<Node extends WamNode = WamNode> {  
  /** should return `true` */  
  readonly isWebAudioModule: boolean;  
  /** The `AudioContext` where the plugin's node lives in */  
  audioContext: BaseAudioContext;  
  /**  
   * The `AudioNode` that handles audio in the plugin  
   * where the host can connect to/from  
   */  
  audioNode: Node;  
  /** This will return true after calling `initialize()` */  
  initialized: boolean;  
  /** The identifier of the current WAM, composed of vendor + name */  
  readonly moduleId: string;  
  /** The unique identifier of the current WAM instance. */  
  readonly instanceId: string;  
  /** The values from `descriptor.json` */  
  readonly descriptor: WamDescriptor;  
  /** The WAM's name */  
  readonly name: string;  
  /** The WAM Vendor's name */  
  readonly vendor: string;  
}
```

```
/**  
 * This async method must be redefined to get `AudioNode` that  
 * will be 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
```

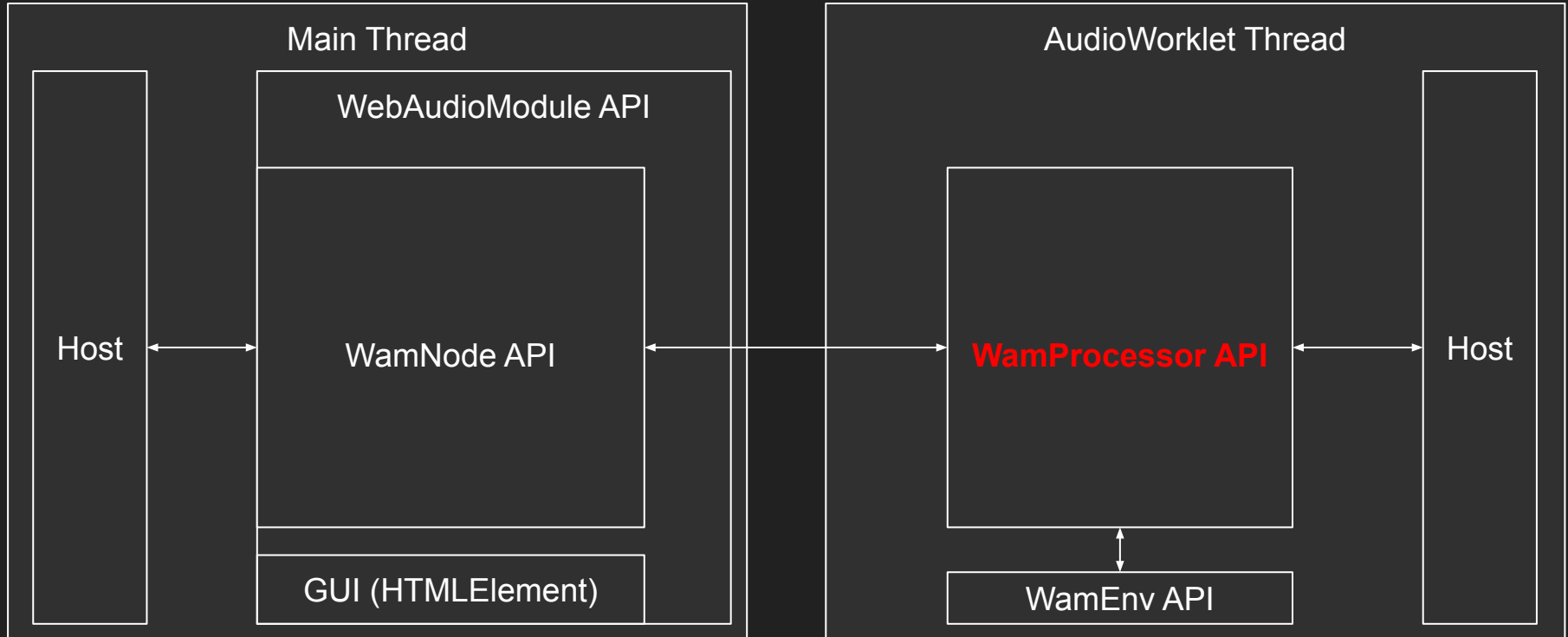
2 - The WamNode API



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

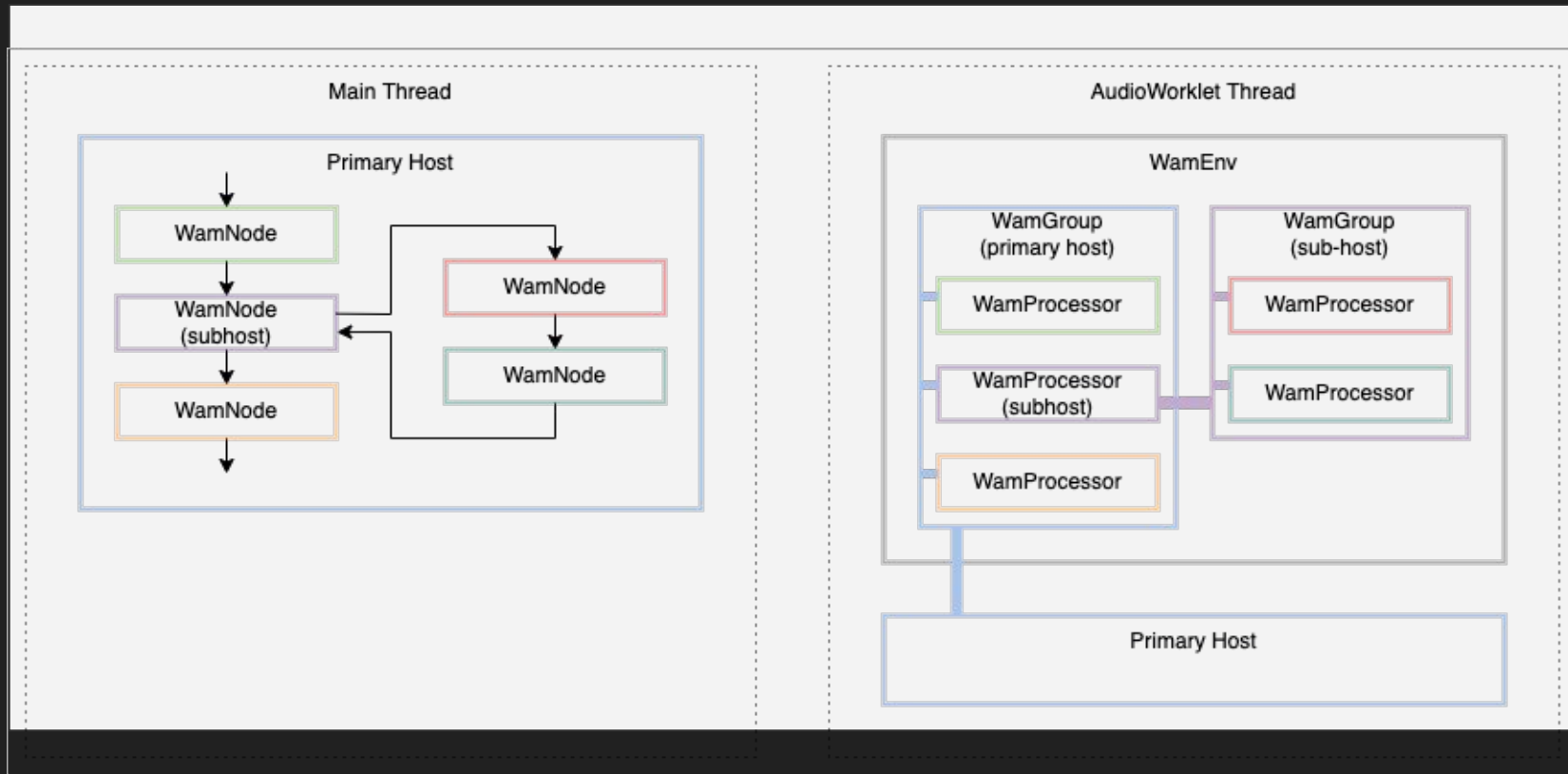


WAMs also have an explicit/implicit WamProcessor API - WamProcessor

```
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.



A dense collage of various digital audio workstation (DAW) plugins and software instruments, arranged in a grid-like fashion. The image features a wide variety of software, including synthesizers like 'SpectrumMod', 'webCZ-101', 'OBXD', and 'VoxAmp 30'; effects processors such as 'JPVerb', 'DualPitchShifter', 'StereofreqShifter', 'ThruZeroFlanger', 'WeirdPhasers', 'Blues Machine', 'Modern Metal Machine', 'DrumSampler', 'PianoRollModule', 'C3', 'Stone Phaser', 'FuzzFace', 'Disto Machine', 'AUGUR', 'MIXEN LFO', 'ENV MATRIX WAVES', 'OscTube', 'Fuzz', 'Pitch', 'Formant', 'PitchShifter', 'Csound', 'Oscillator', 'Filter', 'Filter Envelope', 'Amplifier Envelope', 'Voice Variation', 'Voice Pan', 'Modulation', 'Control', 'Global', 'Mix', 'Master', 'Volume', 'Tune', 'Transpose', 'Spread', 'Unison', 'Glide', 'Bright', 'P ENV', 'Noise', 'Sync', 'MAD', 'STEP', 'Rate', 'Ritch', 'PUM', 'SRA', 'OSC1', 'OSC2', 'OSC3', 'FOH', 'OSC1', 'OSC2', 'OSC3', 'SAR', 'Filter', 'RENO', 'BEND', 'VIBRATO', 'RATE', 'VEL', 'AMP', 'ENV', 'VEL', 'SAR', 'Filter', 'JPVerb', 'DualPitchShifter', 'StereofreqShifter', 'ThruZeroFlanger', 'WeirdPhasers', 'Blues Machine', 'Modern Metal Machine', 'DrumSampler', 'PianoRollModule', 'C3', 'Stone Phaser', 'FuzzFace', 'Disto Machine', 'AUGUR', 'MIXEN LFO', 'ENV MATRIX WAVES', 'OscTube', 'Fuzz', 'Pitch', 'Formant', 'PitchShifter', 'Csound', 'Oscillator', 'Filter', 'Filter Envelope', 'Amplifier Envelope', 'Voice Variation', 'Voice Pan', 'Modulation', 'Control', 'Global', 'Mix', 'Master', 'Volume', 'Tune', 'Transpose', 'Spread', 'Unison', 'Glide', 'Bright', 'P ENV', 'Noise', 'Sync', 'MAD', 'STEP', 'Rate', 'Ritch', 'PUM', 'SRA', 'OSC1', 'OSC2', 'OSC3', 'FOH', 'OSC1', 'OSC2', 'OSC3', 'SAR', 'Filter', 'RENO', 'BEND', 'VIBRATO', 'RATE', 'VEL', 'AMP', 'ENV', 'VEL', 'SAR', 'Filter'. The plugins are displayed in a grid-like fashion, showcasing their diverse interfaces and functionalities. The overall aesthetic is a mix of professional and experimental, with a focus on digital sound manipulation.