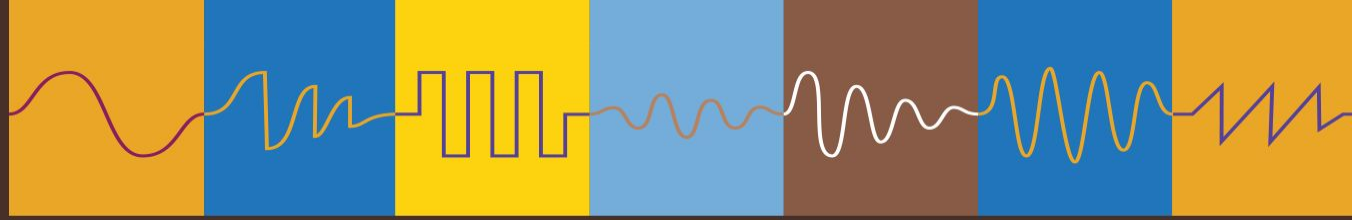
Web Audio Modules 2.0: Audio Plugins for the Web Platform!

Michel Buffa

ADC22



WEBAUDIO MODULES 2.0:

AUDIO PLUGINS FOR THE WEB PLATFORM!

MICHEL BUFFA

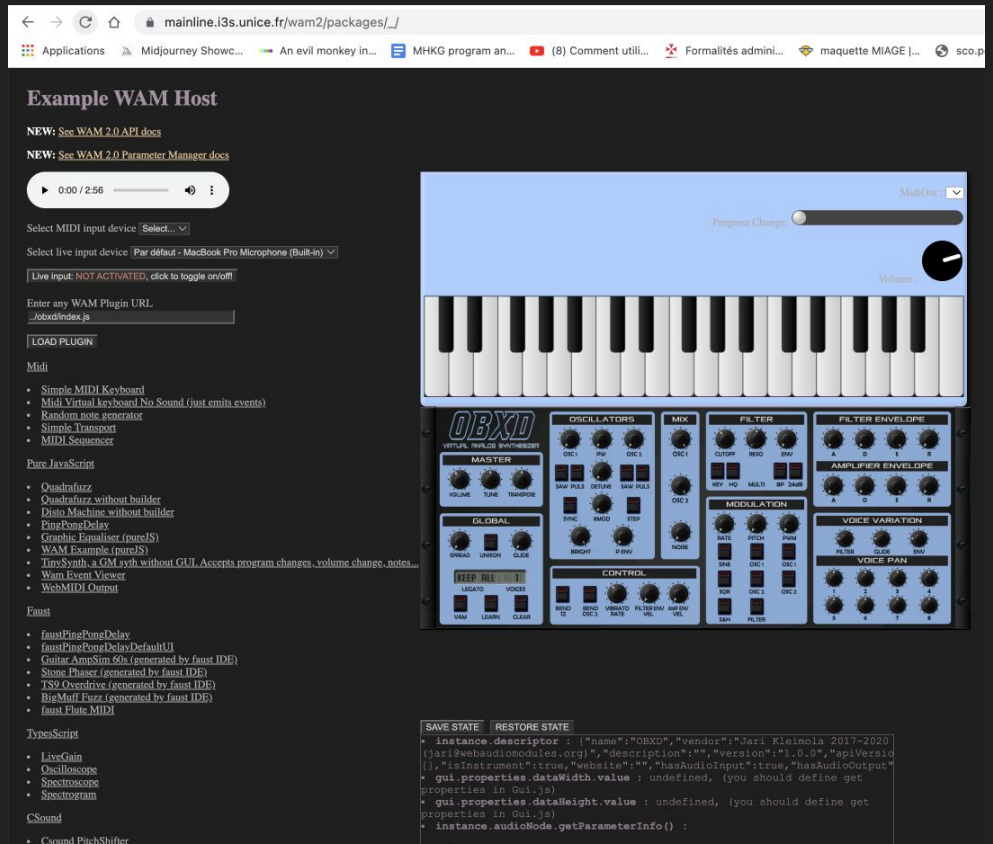
Who am I? What is our group?

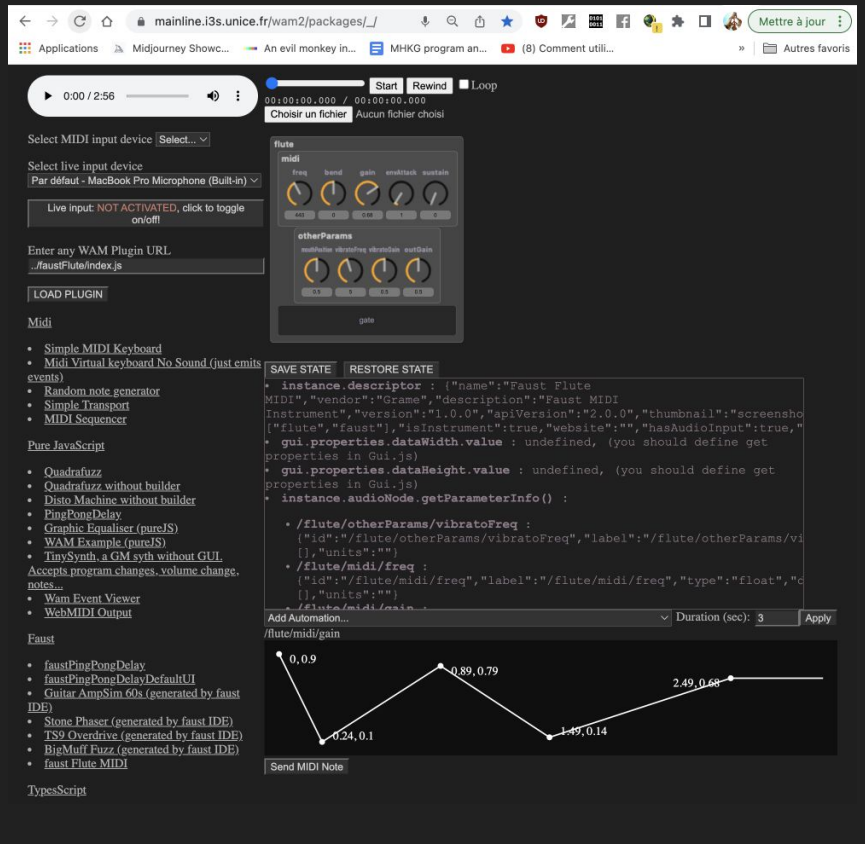
- Professor / researcher at Université Côte d'Azur (UCA), France
  - Member of the WIMMICS research group common to INRIA and I3S lab from CNRS
  - W3C Advisory Committee Representative for UCA
  - Member of the W3C WebAudio Working Group since 2014
  - michel.buffa@univ-cotedazur.fr, @micbuffa
- Other 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…

Web Audio Modules: VSTs for the Web!


Delay
Presets
Modulations
Time
Fdbk
Wet
Base
Width
Stereo
PingPong
SpectrumModalModule
Contour
Bow
Blow
Strike
Model
Modal
Flow
Mallet
Geometry
Brightness
Bow Timbre
Blow Timbre
Strike Timber
Damping
Position
Space
webCZ-101
<CZ-101>
P-01
BRASS ENS.1
webOBXD //
bank
ABS-OBXD-Custom Shop v1.fxb
patch
#2 ABS * Synth
OBXD
VIRTUAL ANALOG SYNTHESIZER
OSCILLATORS
MIX
FILTER
FILTER ENVELOPE
AMPLIFIER ENVELOPE
MASTER
GLOBAL
MODULATION
CONTROL
VOICE VARIATION
VOICE PAN
highX
highBand
mDepth
midX
loudBand
mFreq
lowX
earlyDiff
damp
size
wet
160
JPVerb
TEMPER
CUTOFF
RESONANCE
FEEDBACK
DRIVE
CURVE
Depth
Notch Width
Feedback Gain
NotchFreq
MaxNotchFreq
Speed
MinNotchFreq
level
VibratoMode
invert
Phaser Rix
Mix
Shift
ShiftR
WindowSize
DualPitchShifter
Volume
Bass
Middle
Treble
Master
Clean / Crunch
VoxAmp 30
TS9 Overdrive
QuadraFuzz
Sweet Wah
DISTORTION
Owlbeat
Pitch
Formant
Csound
PitchShifter
FuzzFace
Dallas Wah
Boss Cut
Effect
Humbucker Sim
damping
diffusion
modFreq
delayTime
feedback
size
modDepth
GREYHOLE
Input Gain
Noise Gate
Utility
L R
OUT
BIG MUFF
WAHBABY
VOLUME
MASTER
DRIVE
BASS
MIDDLE
TREBLE
REVERB
PRESENCE
Folk Metal String
CLEAN MACHINE
L-R Offset
Mix
Shift Scalar
Shift
StereoFreqShifter
Delay
Depth
L-R Offset
Rate
ThruZeroFlanger
Feedback
L-R Offset
Rate Scalar
Rate
WeirdPhaser
VOLUME
MASTER
DRIVE
BASS
MIDDLE
TREBLE
REVERB
PRESENCE
DISTO MACHINE
AUGUR (TRACK 1)
OSC
FILTER
AMP
GLOBAL
AUGUR
MODE
2P LP
CHORUS
MIXENV LFO
ENV MATRIX WAVES
DEXED
Volume
Master
Drive
Bass
Middle
Treble
Reverb
Presence
Clean
Blues Machine
BASEPITCH
MIX
PITCHMOD RELEASE
BLIPPER
DEAD ZONE
NOISE GATE
DEATHGATE
feedback
KBVerb
Delay
Feedback
Interpolation
Dry/Wet
Super Wet
Smooth Delay
DECAY
MIX
SHIMMER
TONE
FaustEQ
VOLUME
MASTER
DRIVE
BASS
MIDDLE
TREBLE
REVERB
PRESENCE
Warm Clean
MODERN METAL MACHINE
DrumSampler
Kick
Load
OH
Crash
Ride
Low Tom
Mid Tom
High Tom
CH
Kick
Rimshot
Snare
Clap
Output
Tone
Pan
Gain
PianoRollModule
Settings
C3
LFO FREQ
FEEDBACK DEPTH
FEEDBACK BASS CUT
MIX
STEREO PHASE
DEPTH
STONE PHASER
Synth101
Presets
Portamento
LFO
Oscillator
Mixer
Filter
Time
Mode
Rate
Waveform
Tune
Mod
PW
PWM
Range
Sub Range
Pulse
Saw
Sub
Noise
Freq
Res
Env
Mod
Kybd
Fuzz
Resonance
Vibrato
OscTube
Low
Med
Freq
Gain

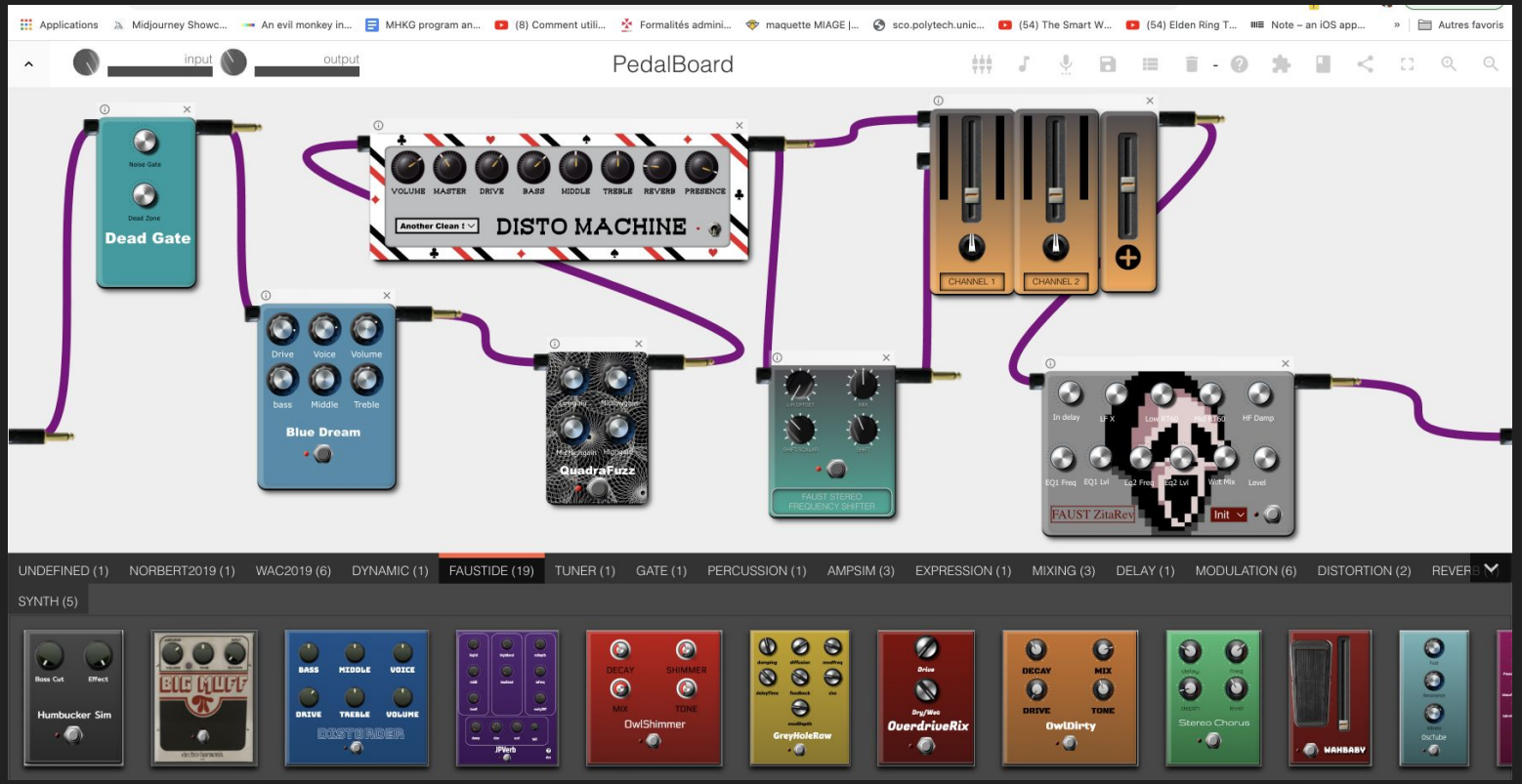
Wams in hosts… here with a host from the WAM distrib https://mainline.i3s.unice.fr/wam2/packages/_/



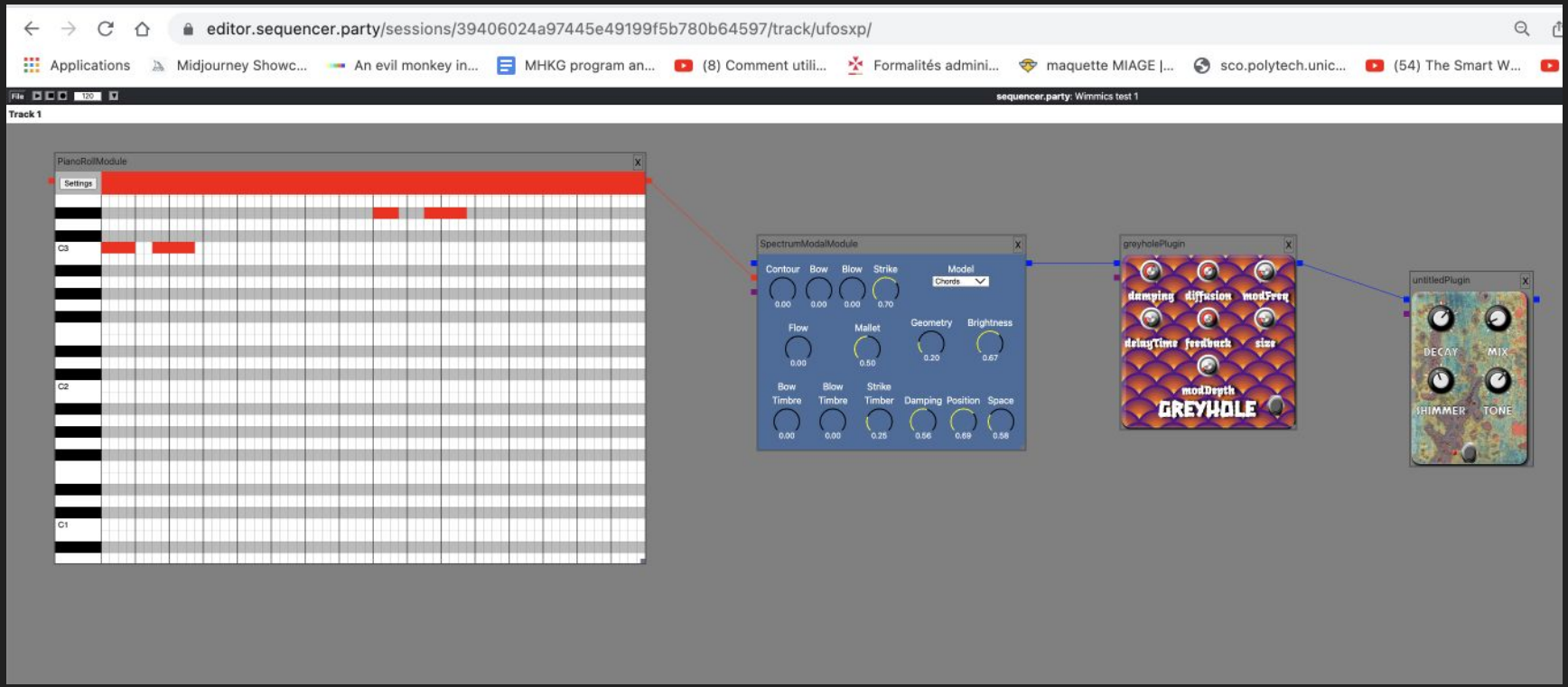


WAMs in hosts… WIP open source DAW (100% WAM-based)

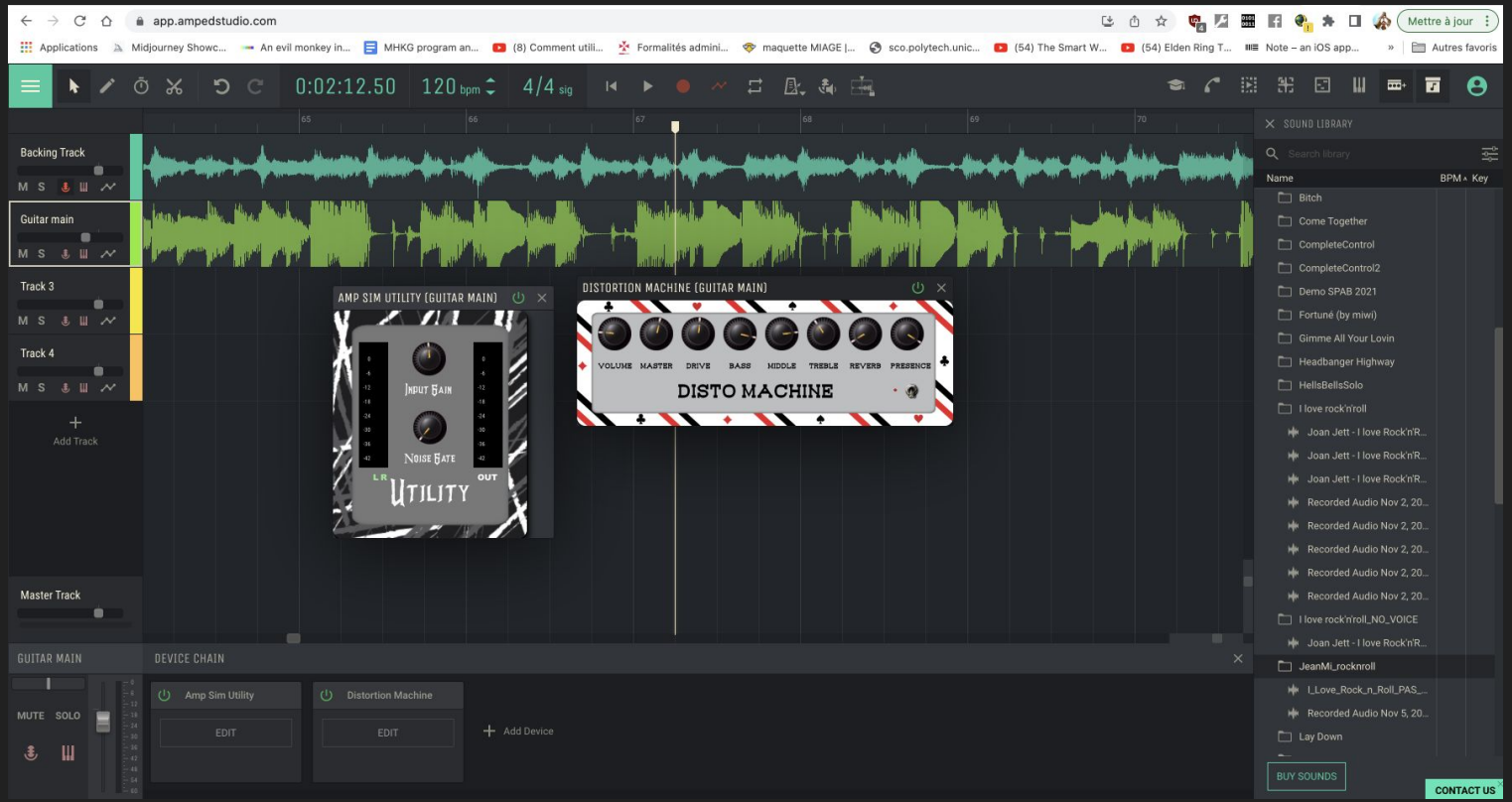
WAMs in hosts… here the WASABI pedalboard



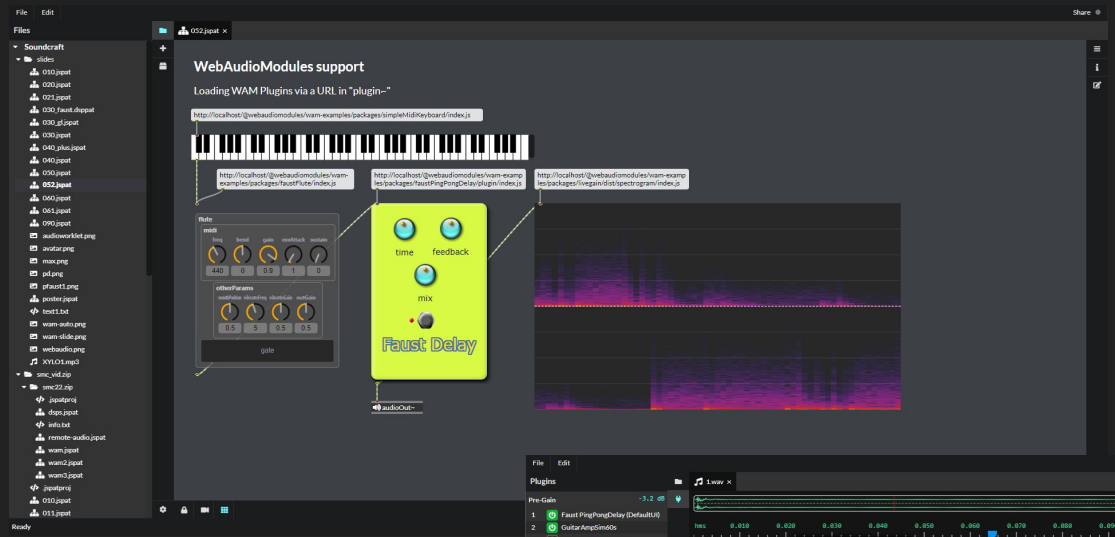
WAMs in hosts… https://sequencer.party/



Wams in hosts… https://app.ampedstudio.com/



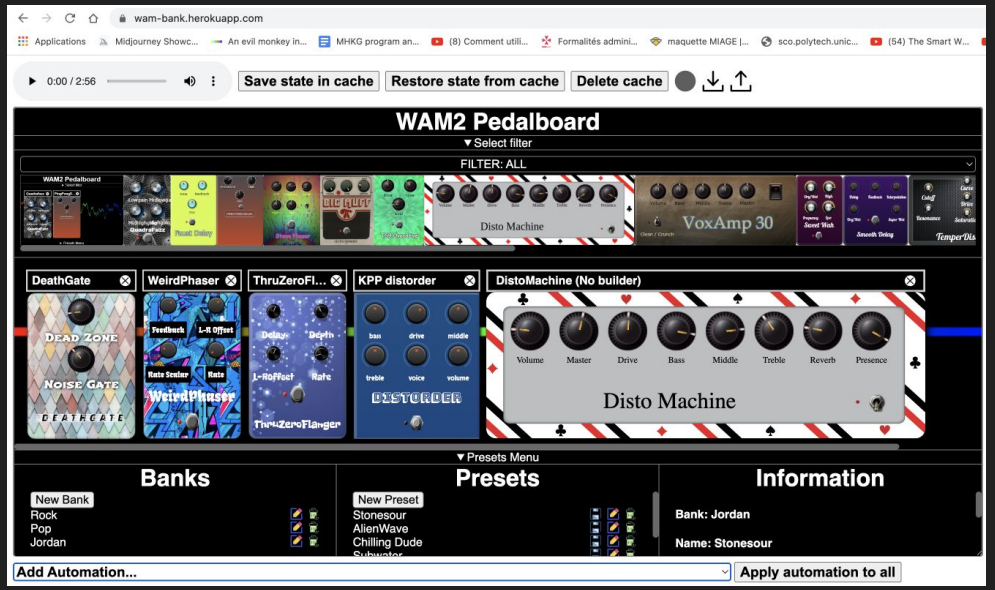
WAMs in hosts… JSPatcher, aka Max MSP in the browser (https://github.com/Fr0stbyteR/jspatcher)

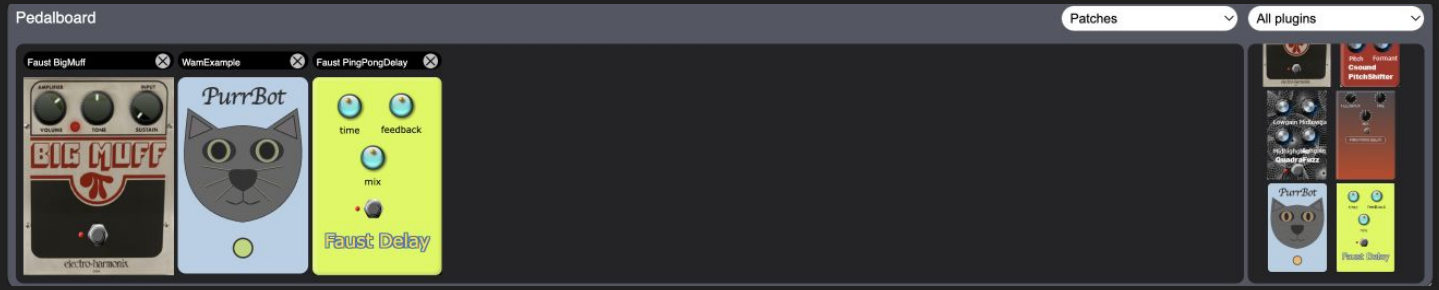


WAMs in plugins that acts as hosts (i.e pedalboard)

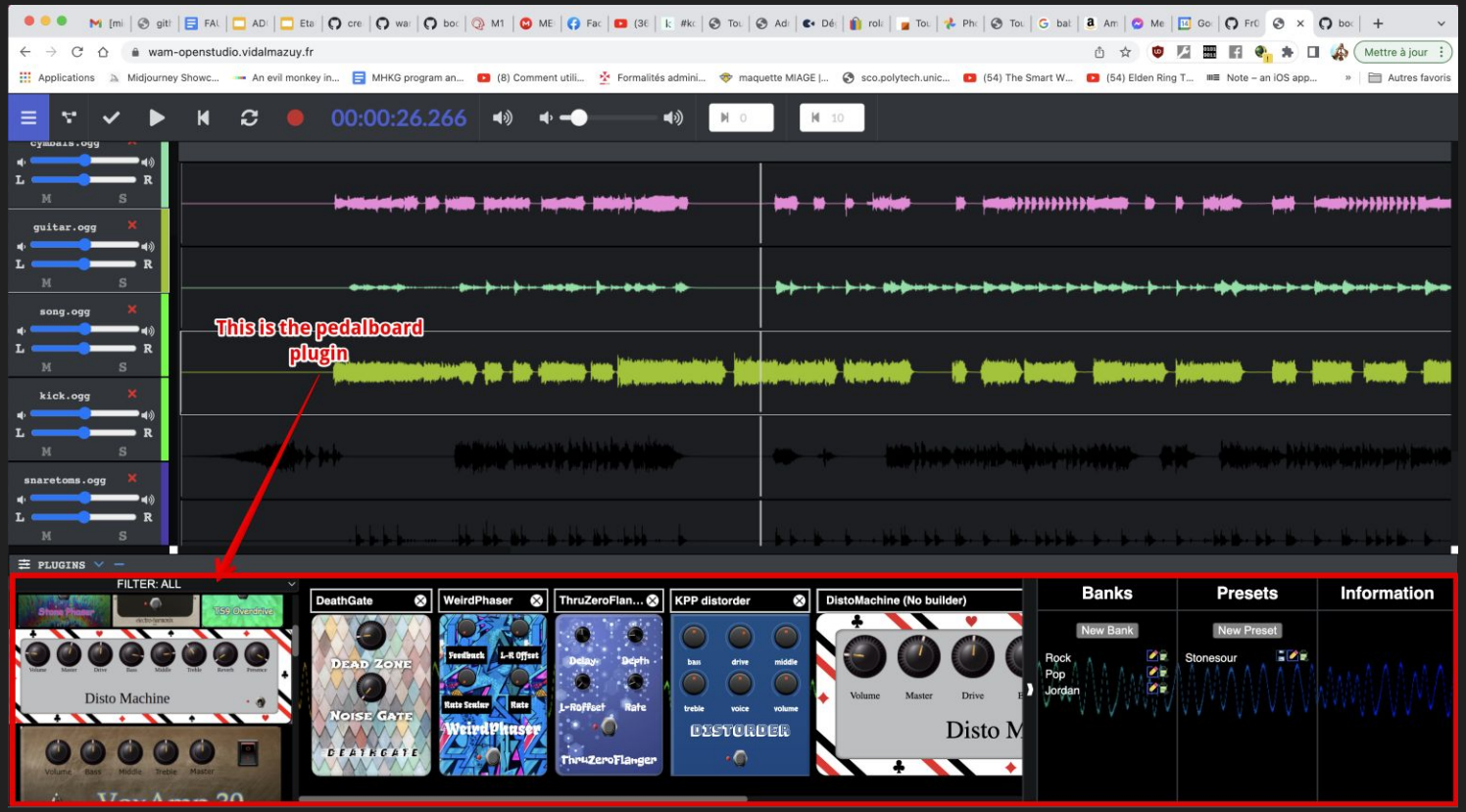
These ones are open source

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





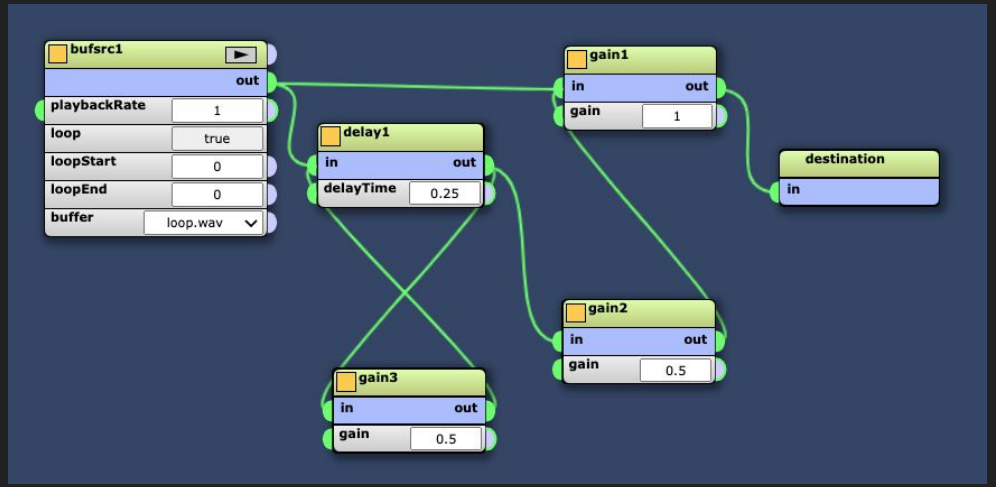
… and loaded in a DAW!



A 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.



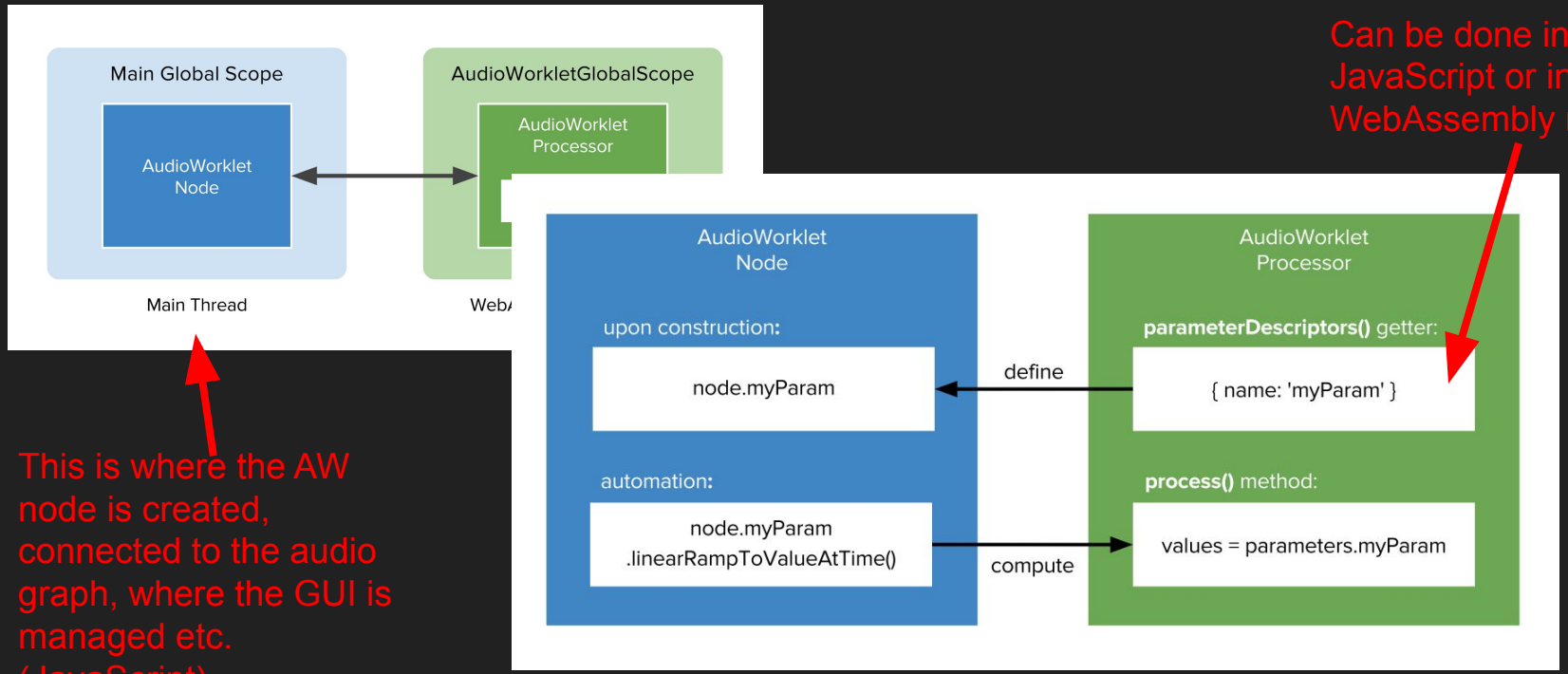

```
this.nodes.gain1 = this.audioctx.createGain();
this.nodes.bufsrc1 = this.audioctx.createBufferSource();
this.nodes.bufsrc1.loop = true;
this.nodes.bufsrc1.buffer = this.buffers['loop.wav'].data;
this.nodes.delay1 = this.audioctx.createDelay();
this.nodes.delay1.delayTime.value = 0.25;
this.nodes.gain2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

Can be done in JavaScript or in a WebAssembly module!



This is where the AW node is created, connected to the audio graph, where the GUI is managed etc. (JavaScript)

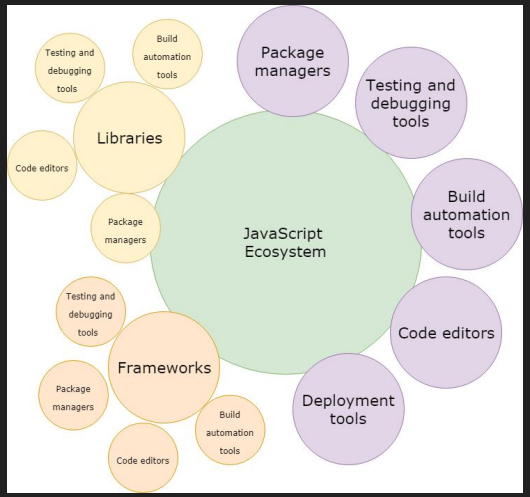
Web Audio development

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.



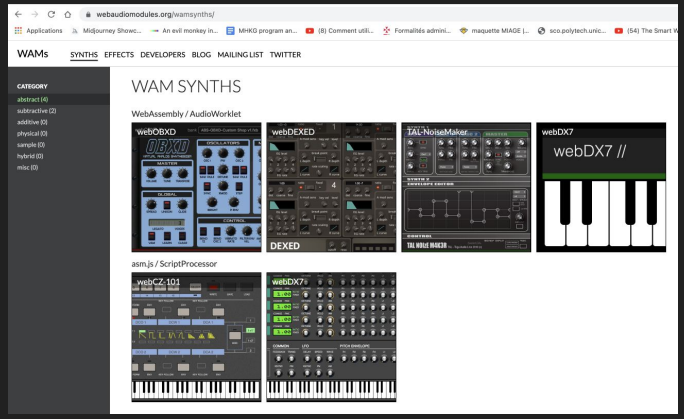
How to combine all this?

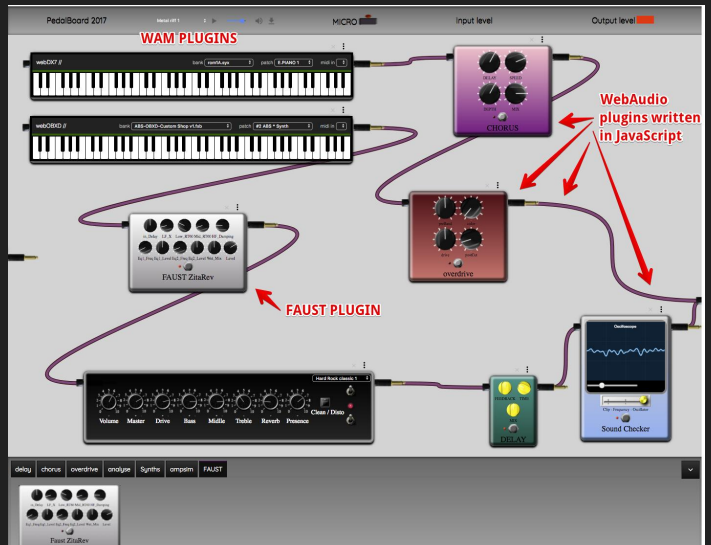
2015: First WAM proposal by J.Kleimola and O.Larkin

- 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.)

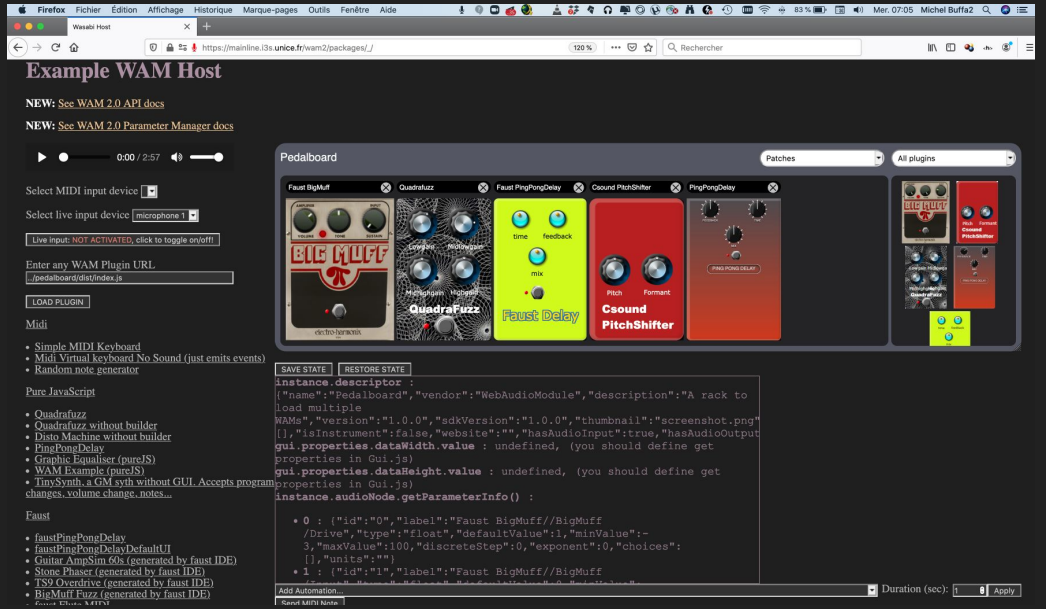
- Please web developers,
- Support DSLs like FAUST, other improvements…





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

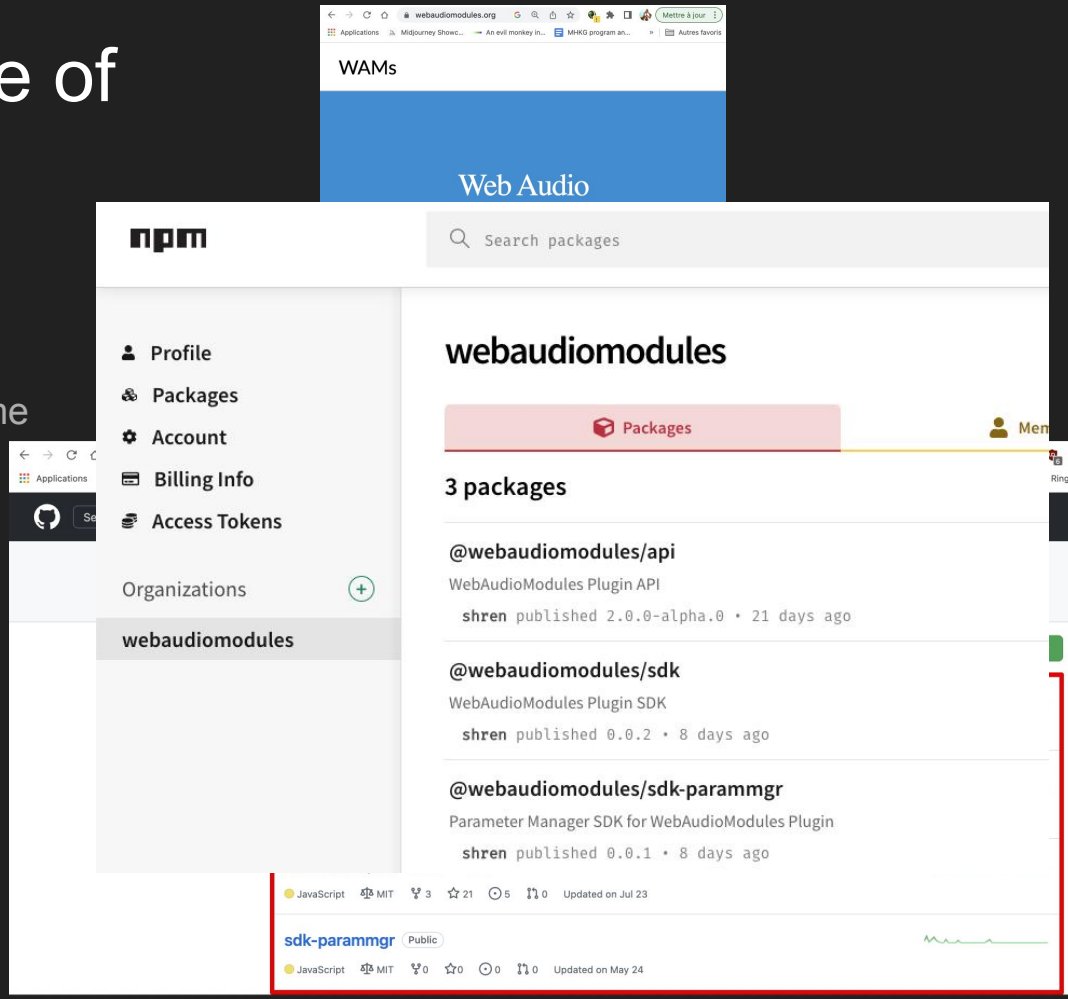


How to start with WAMs!

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

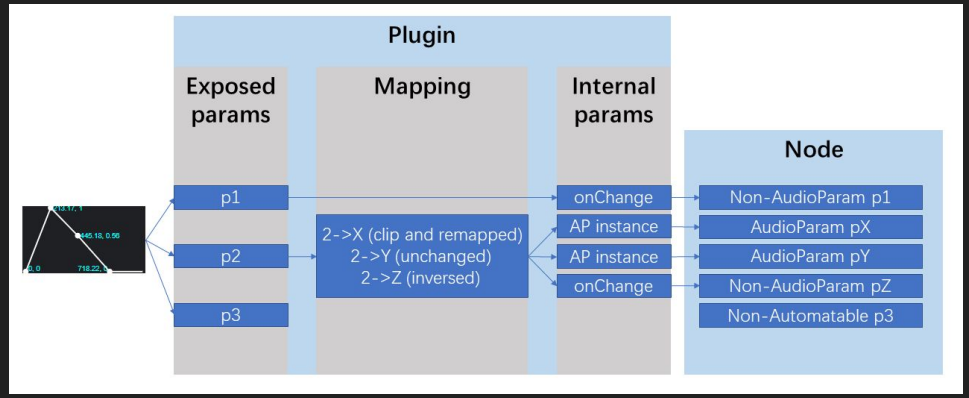
API (Standard)	SDK (Implementation/Tools)
- Defines required methods - Abstract classes	- Reference API implementations - Utility classes and example plugins

- 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.

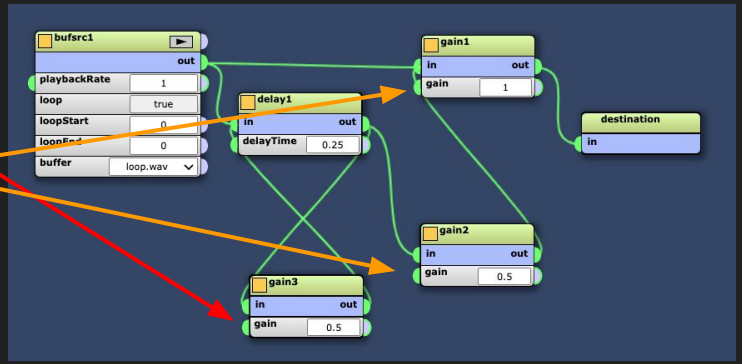
The sdk-parammgr repository

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).

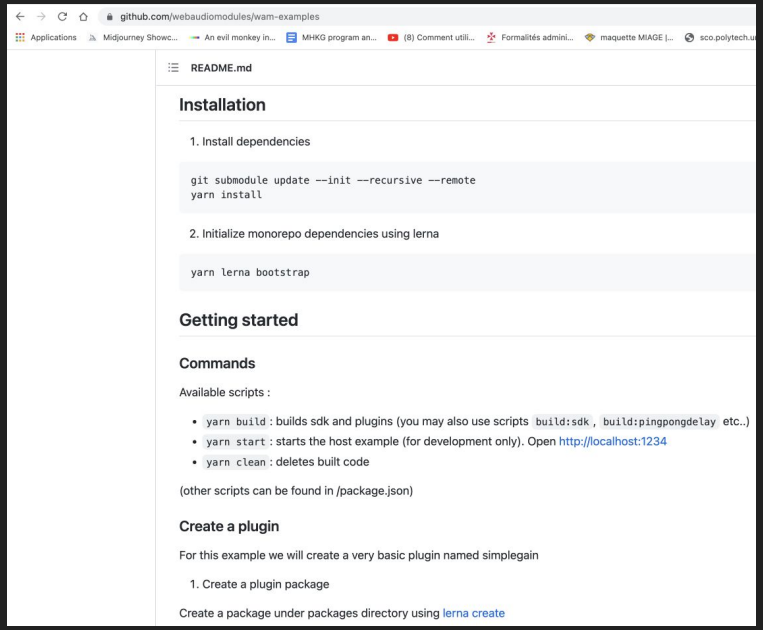


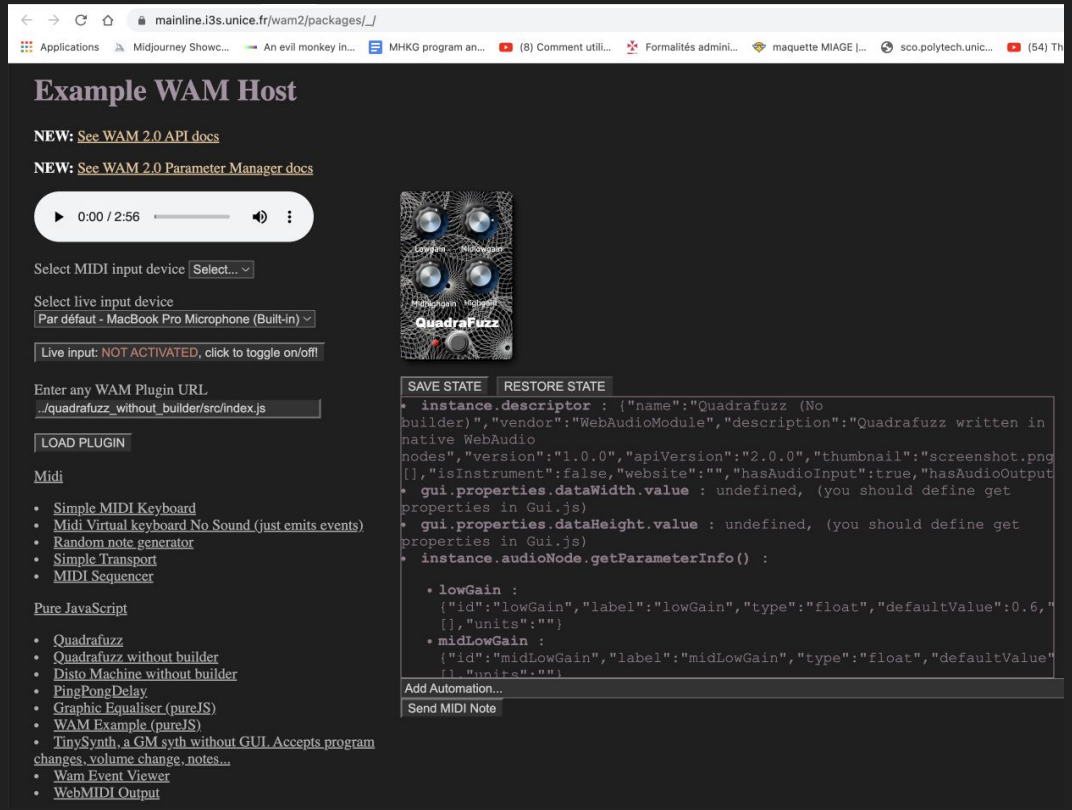
3 exposed parameters



Dozens of internal parameters

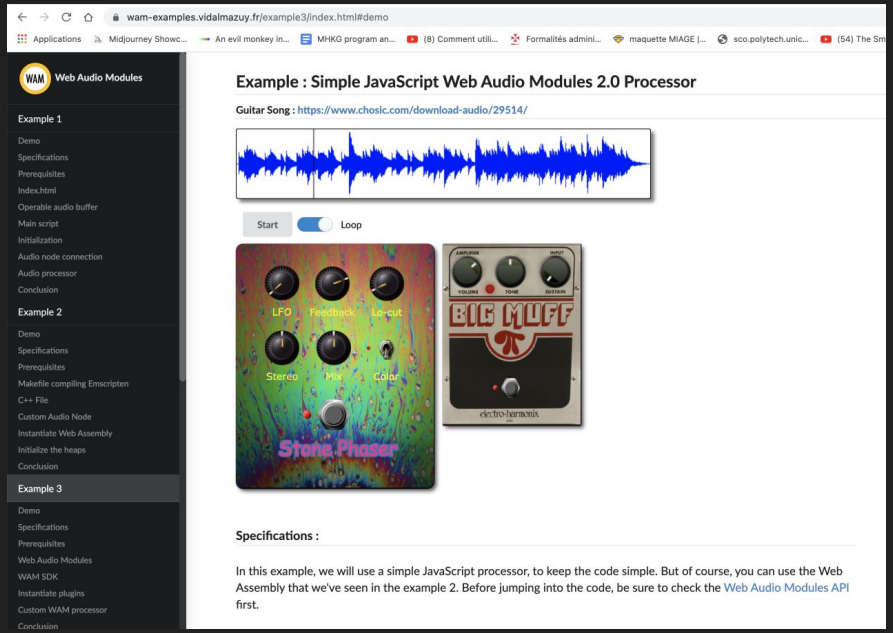
The wam-examples repository

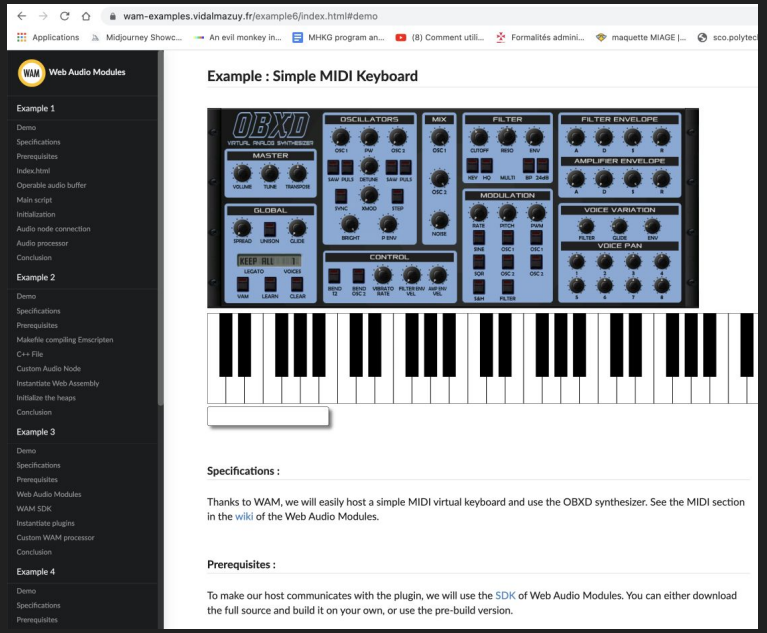




WAM2 step by step tutorials

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

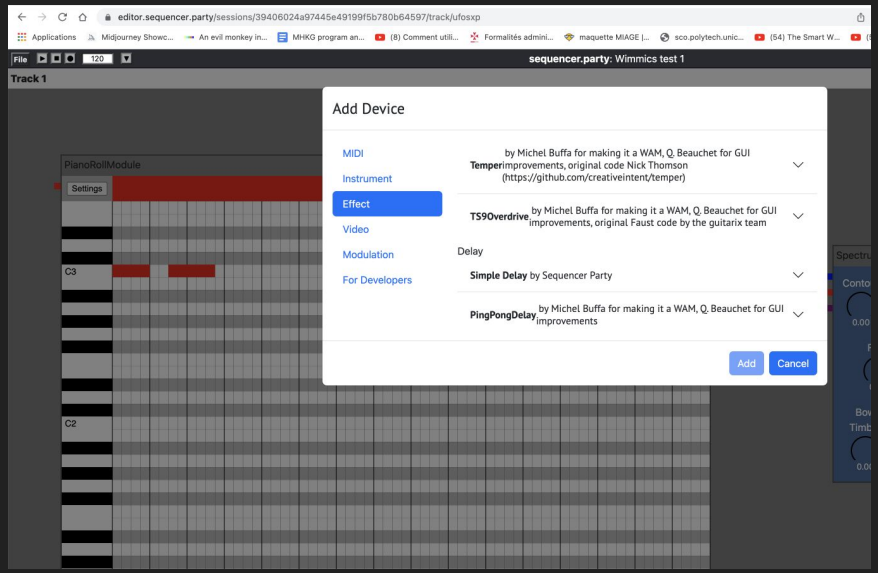




Also, check the wam-community repository https://github.com/boourns/wam-community

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 utilitary 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.

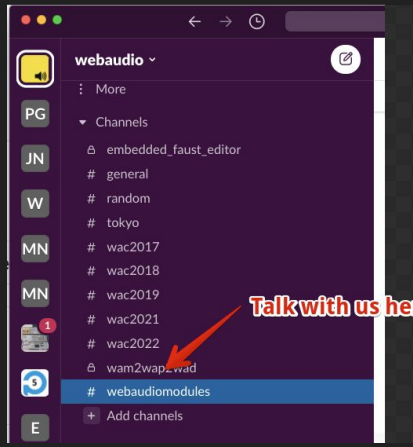
Conclusion / Perspectives

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

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

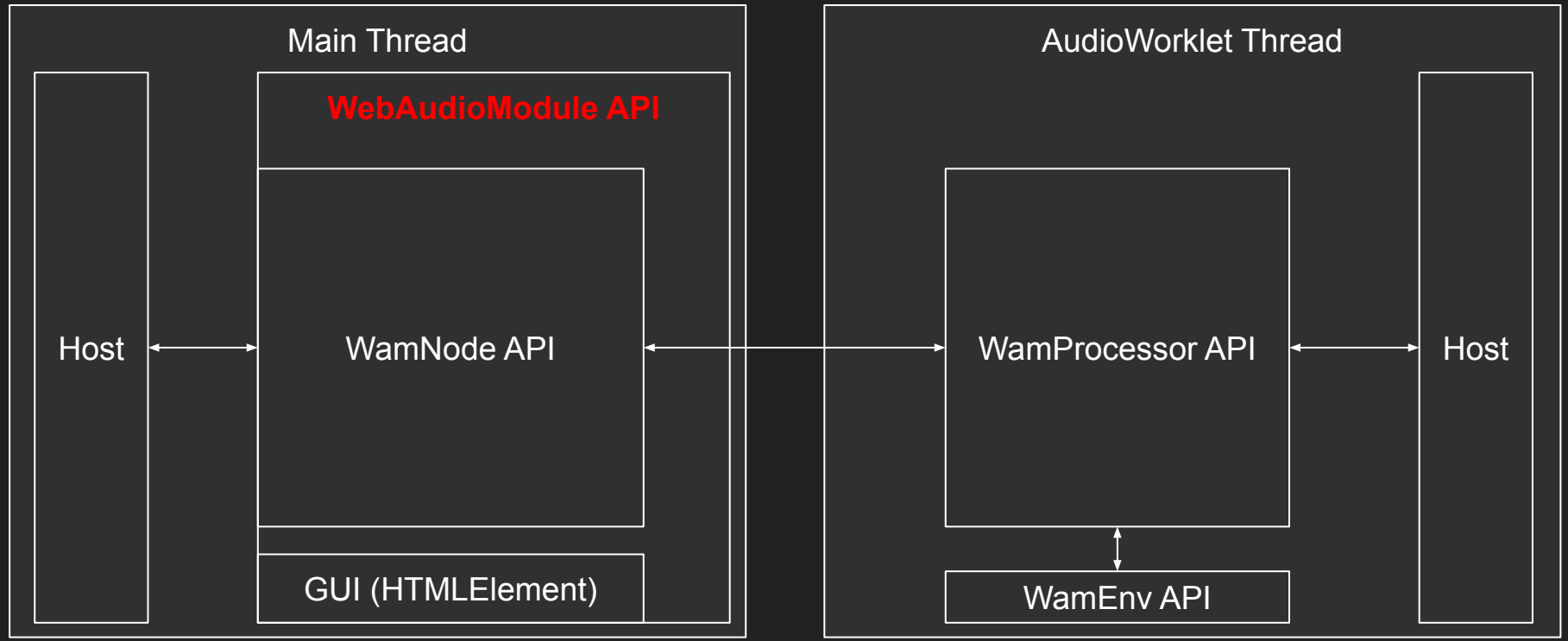
Things that will come soon:

- 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

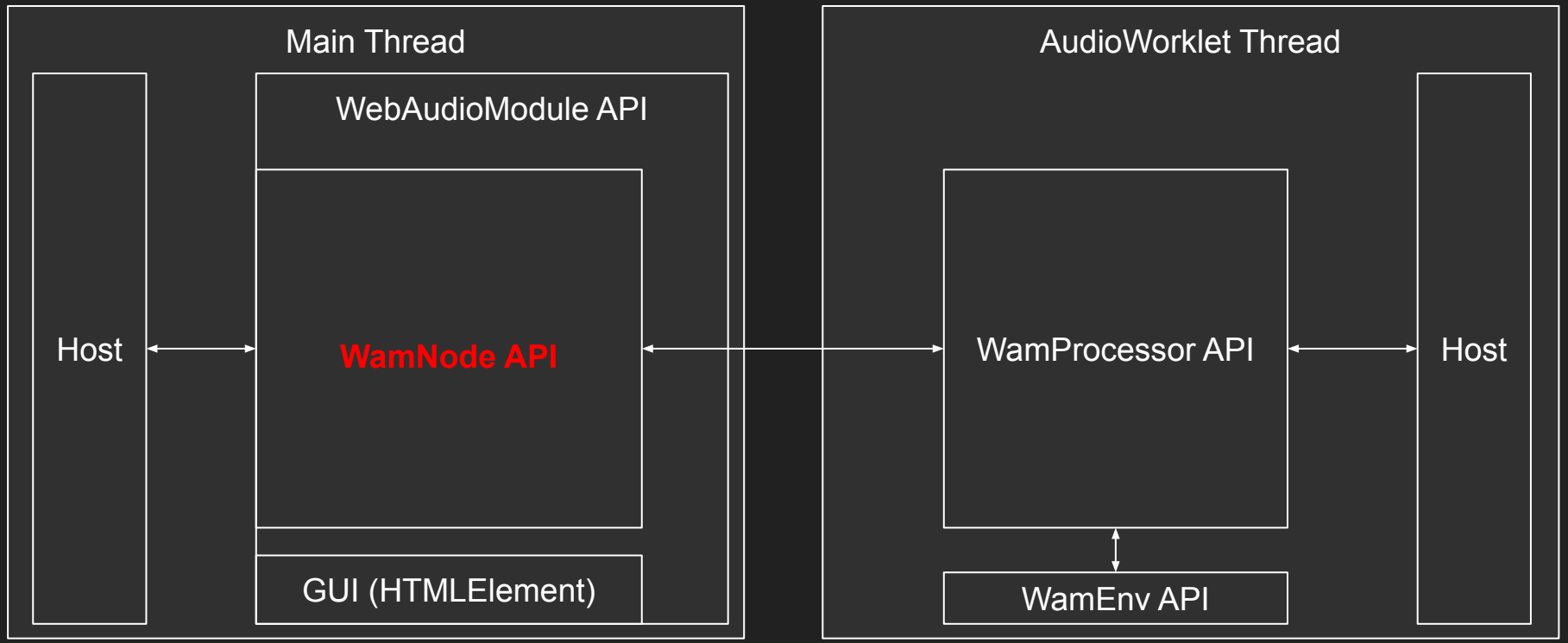


A plugin = instance of a WAM = “a WAM”
API - WebAudioModule

```ts
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 vender + 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 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
}
```

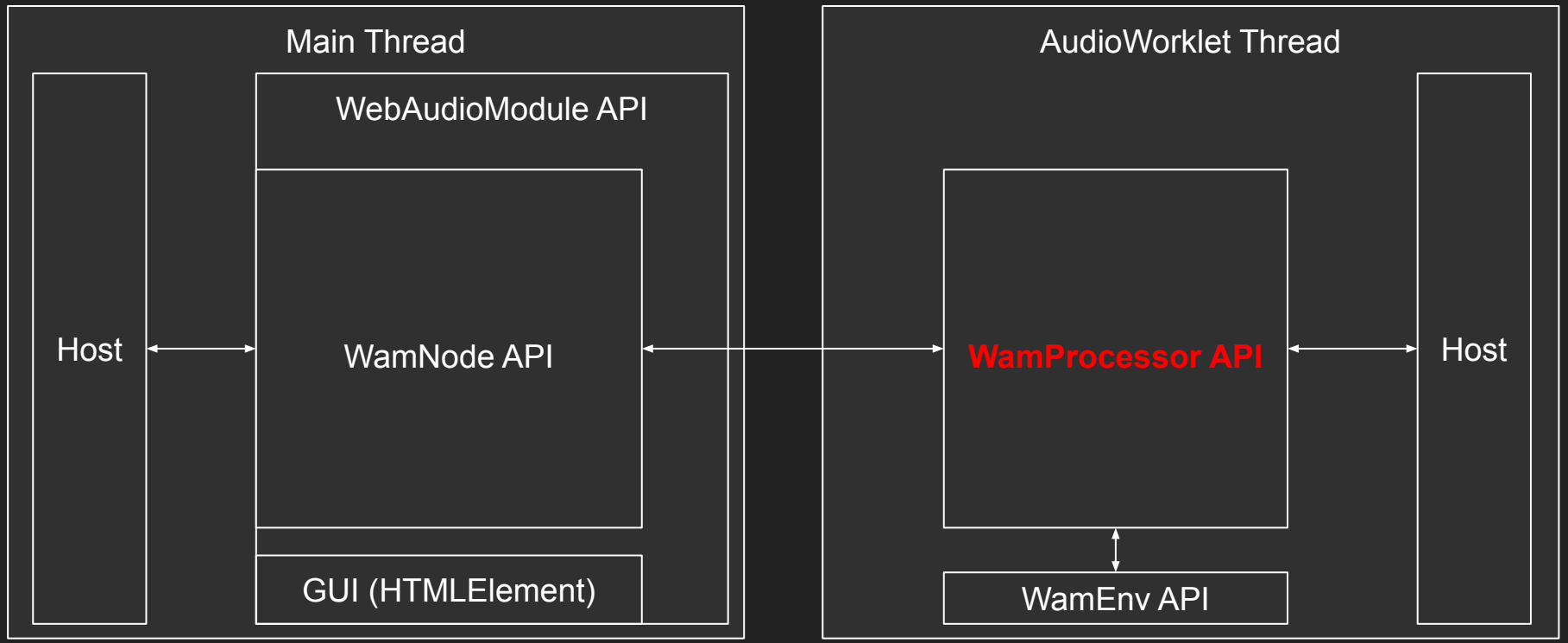
2 - The WamNode API



A WAM contains a WamNode, here is the API

```ts
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

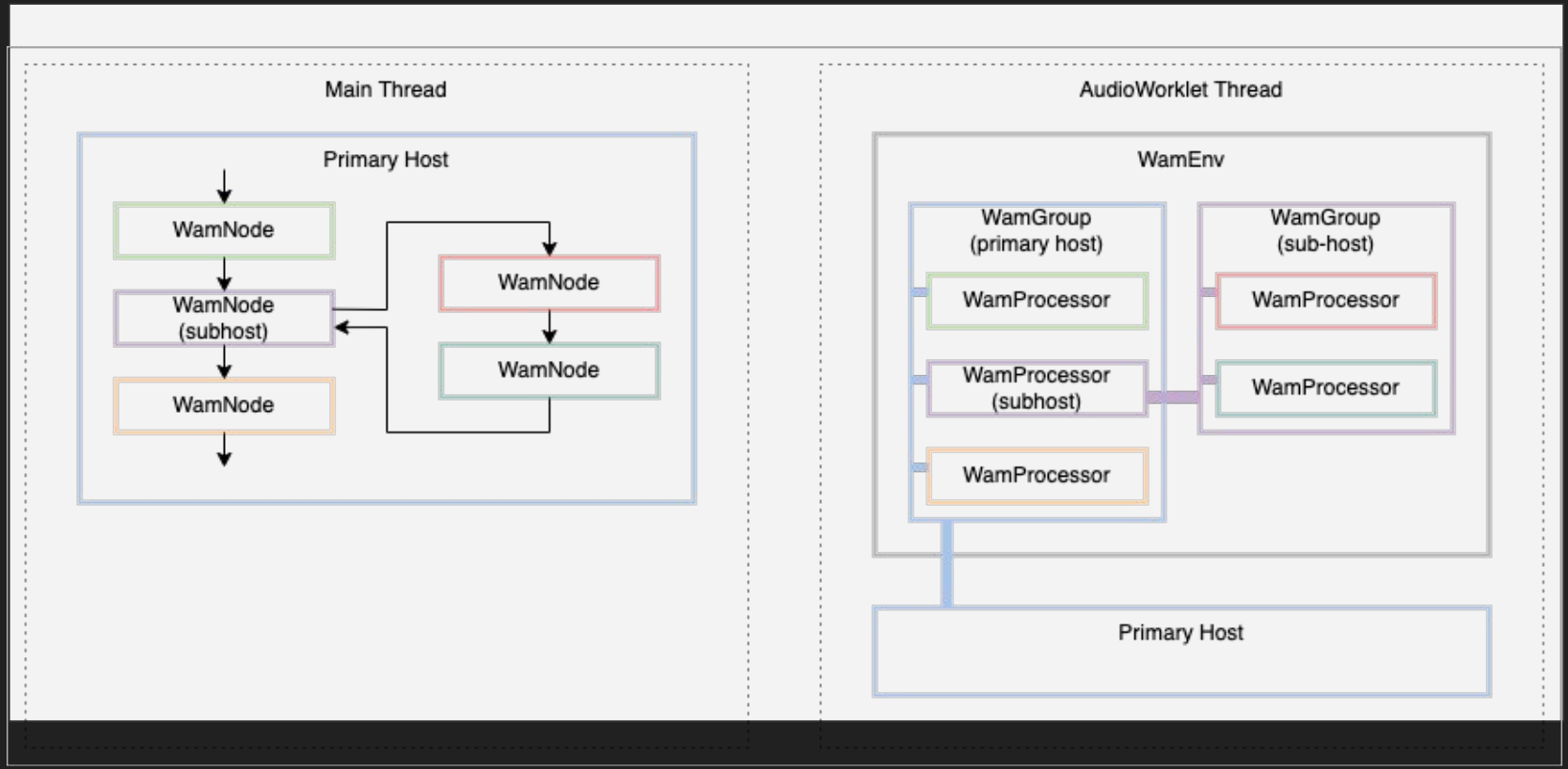


WAMs also have an explicit/implicit WamProcessor API - WamProcessor

```
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.




Delay
Presets
Modulations
Time
Fdbk
Wet
Base
Width
Stereo
Ping/Pong
SpectrumModalModule
Contour
Bow
Blow
Strike
Model
Modal
Flow
Mallet
Geometry
Brightness
Bow Timbre
Blow Timbre
Strike Timber
Damping
Position
Space
webCZ-101
<CZ-101>
P-01
BRASS ENS,1
webOBXD //
bank
ABS-OBXD-Custom Shop v1.fxb
patch
#2 ABS * Synth
OBXD
VIRTUAL ANALOG SYNTHESIZER
OSCILLATORS
MIX
FILTER
FILTER ENVELOP
AMPLIFIER ENVEL
MODULATION
VOICE VARIATIO
VOICE PAN
MASTER
GLOBAL
CONTROL
FuzzFace
Fuzz
Level
Dallies Wah
Boss Cut
Effect
Humbucker Sim
highX
highBand
mDepth
midX
louBand
mFreq
louX
earlyDiff
damp
size
wet
t60
JPVerb
TEMPER
CUTOFF
RESONANCE
FEEDBACK
Depth
Notch Width
Feedback Gain
NotchFreq
MaxNotchFreq
Speed
MinNotchFreq
level
VibratoMode
invert
Phaser Rix
Mix
Shift
Shift
WindowSize
DualPitchShifter
VoxAmp 30
Volume
Bass
Middle
Treble
Master
Clean / Crunch
TS9 OverDrive
Quadra Fuzz
Sweet Wah
DISTORTION
OwlDrivg
Pitch
Formant
Csound PitchShifter
damping
diffusion
modFreq
delaytime
feedback
size
modDepth
GREYHOLE
INPUT GAIN
NOISE GATE
L-R
OUT
UTILITY
BIG MUFF
WAHBABY
L-R Offset
Mix
Shift Scalar
Shift
StereoFreqShifter
Delay
Depth
L-R Offset
Rate
ThruZeroFlanger
Feedback
L-R Offset
Rate Scalar
Rate
WeirdPhaser
VOLUME
MASTER
DRIVE
BASS
MIDDLE
TREBLE
REVERB
PRESENCE
Folk Metal String
CLEAN MACHINE
DISTO MACHINE
AUGUR [TRACK 1]
OSC
FILTER
AMP
GLOBAL
AUGUR
MODE
2P LP
CHORUS
MIXENV LFO
ENV MATRIX WAVES
DECAY
MIX
SHIMMER
TONE
Volume
Master
Drive
Bass
Middle
Treble
Reverb
Presence
Clean
Blues Machine
BASEPITCH
MIX
PITCHMOD RELEASE
FLIPPER
DEAD ZONE
NOISE GATE
DEATHGATE
feedback
KBVerb
Delay
Feedback
Interpolation
Dry/Wet
Super Wet
Smooth Delay
DEXED
Low Boost/Cut
Transition Frequency
Peak Boost/Cut
Peak Frequency
Peak Q
High Boost/Cut
FaustEQ
Warm Clean
MODERN METAL MACHINE
DrumSampler
Load
Kick
OH
Crash
Ride
Low Tom
Mid Tom
High Tom
CH
Kick
Rimshot
Snare
Clap
Output
Tone
Pan
Gain
PianoRollModule
Settings
C3
Synth101
Presets
Portamento
LFO
Oscillator
Mixer
Filter
Time
Rate
Tune
Mod
PW
PWM
Mode
Waveform
Range
Sub Range
Pulse
Saw
Sub
Noise
Freq
Res
Env
Mod
Kybd
LFO FREQ
FEEDBACK DEPTH
FEEDBACK BASS CUT
MIX
STEREO PHASE
Bypass
STONE PHASER
Fuzz
Resonance
Vibrato
OscTube
Low
Med
High
Freq
Gain