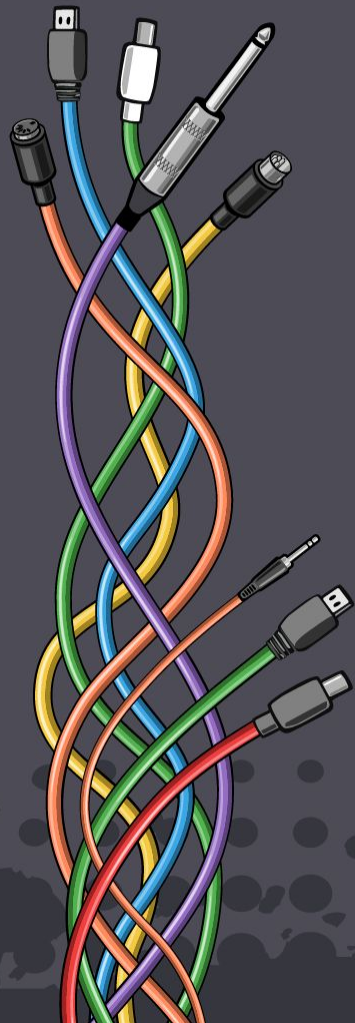
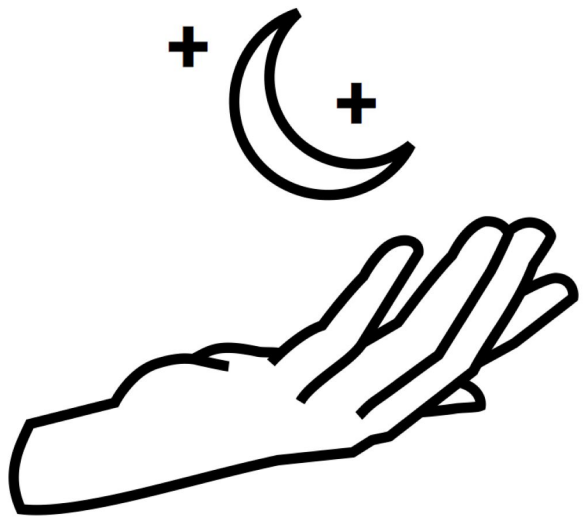




WORKSHOP: WEB UIs FOR MUSIC APPS

**ANNA WSZEBOROWSKA,
HARRIET DRURY, EMMA FITZMAURICE,
PAULINE NEMCHAK & SIMEON JOSEPH**

ADC²⁵
Bristol



@dynamic_cast

We are a
peer-to-peer study
group for under
represented groups
in programming

dynamic-cast.github.io

Presentation slides

<https://github.com/dynamic-cast/ADC25-Web-UIs/wiki>



You Will Be Hearing From...

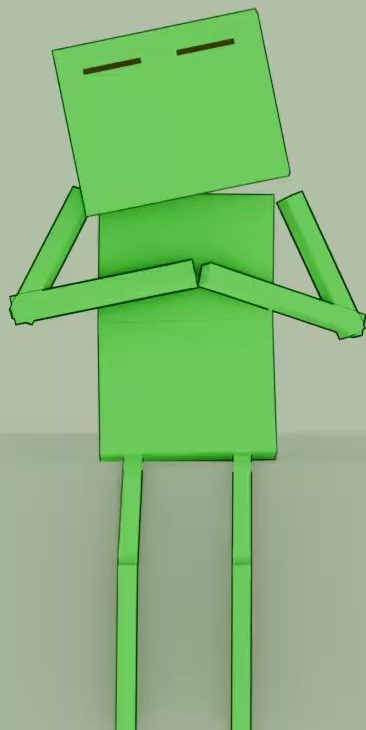
Emma Fitzmaurice

Pauline Nemchak

Harriet Drury

Simeon Joseph

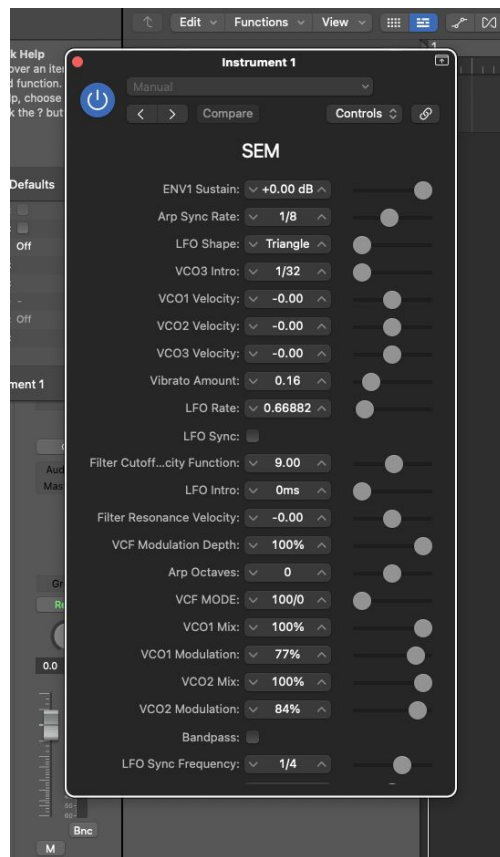
Anna Wszeborowska



Workshop goals

- Learn about what webviews are
- Build a Web UI using modern web techniques
- Hook that UI up to a CMajor patch to create a plugin

Lets Get GUI



Plugin

Standalone



Compositors - The GUI part of your OS

Windows - Desktop Window Manager (DWM)

MacOS - Quartz

Linux - X11/Wayland



GUI Libraries



SwiftUI



JUCE



Flutter



WinUI

Qt



.net MAUI

Slint



Enter the Webview

Ever heard of the world wide web?

We can steal their toys

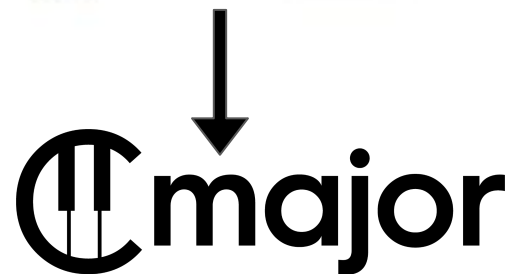
MacOS/Linux - WebKit

Windows - Edge WebView2



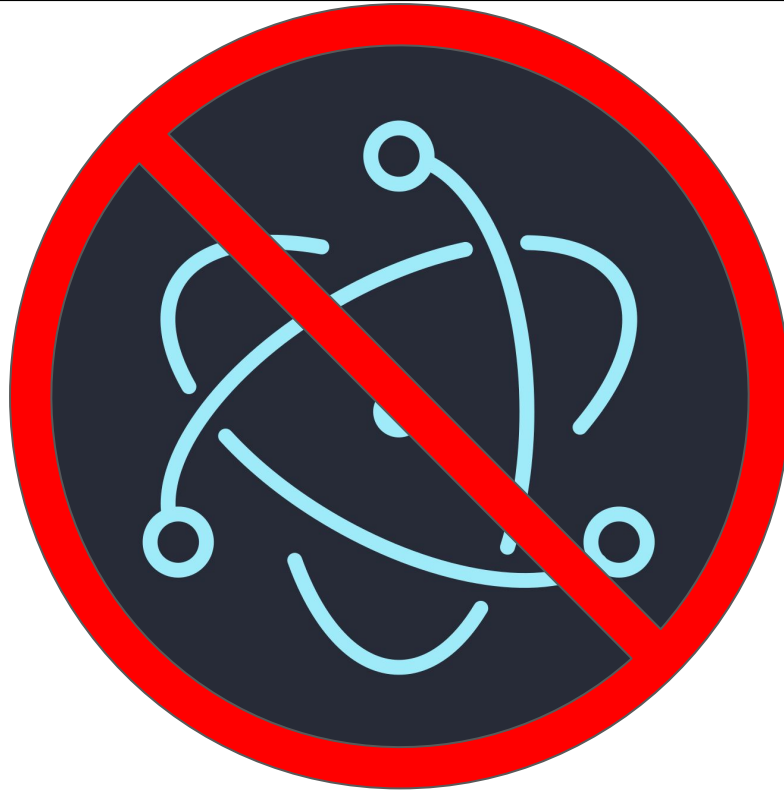
Webview Libraries

Language	Project
Ada	thechampagne/webview-ada
Bun	tr1ckydev/webview-bun
C#	webview/webview_csharp
C3	thechampagne/webview-c3
Crystal	naqvis/webview
D	thechampagne/webview-d, ronnie-w/webviewd
Deno	webview/webview_deno
Go	webview/webview_go
Harbour	EricLendvai/Harbour_WebView
Haskell	lettier/webviewhs
Janet	janet-lang/webview
Java	webview/webview_java
Kotlin	Winterreisender/webviewko
Nim	oskca/webview, neroist/webview
Node.js	Winterreisender/webview-nodejs
Odin	thechampagne/webview-odin
Pascal	PierceNg/fpwebview
Python	congzhangzh/webview_python, zserge/webview-python
PHP	0hr/php-webview
Ruby	Maaarcocr/webview_ruby
Rust	Boscop/web-view
Swift	jakenvac/SwiftWebView
V	mallsi/mui, ttytm/webview
Vala	taozuhong/webview-vala
Zig	thechampagne/webview-zig



TAURI

Is this like Electron?



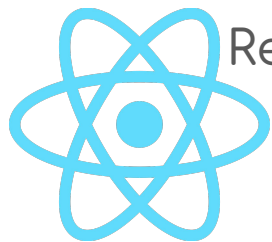
Web Frameworks

Svelte



Vue

Lit



React



Angular

Ember



Why Use a Webview?

Yay	Boo
- This is a well worn path - Accessibility - Testing - Design tools - Resources - Developers - Portability	- Optimisation - Existing codebase - "Don't wanna"

Front-end development

What we're going to build



Setting up the workshop project

- Go to the ADC25-Web-UIs workshop repository
<https://github.com/dynamic-cast/ADC25-Web-UIs>

```
git clone https://github.com/dynamic-cast/ADC25-Web-UIs.git
```



HTML

Hyper**T**ext **M**arkup **L**anguage

Defines the meaning and structure of web content

HTML

```
<html>
```

```
  <head>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```

head

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Document title</title>
```

```
  <link rel="stylesheet" href="./styles.css">
```

```
</head>
```

body

```
<body>  
  <form action="/submission">  
    <label for="name">Enter your name: </label>  
    <input type="text" name="name" id="name"  
required />  
  </form>  
</body>
```

Void Elements

`<img>`

`<input>`

`<br>`

`<source>`

`<meta>`

Replaced Elements

`<video>`

`<img>`

`<iframe>`

`<audio>`

Project Structure

karplus_strong/main.cmajorpatch

```
"view": {  
  "src": "ui/dist/index.js",  
  ...  
}
```



Lit



rollup

ui/rollup.config.js

```
export default {  
  input: 'index.js',  
  output: {  
    file: 'dist/index.js',  
    format: 'esm',          // can be cjs, esm, or iife  
  },  
  plugins: [resolve(), commonjs()],  
};
```

package.json

```
"scripts": {  
  "build": "rollup -c",  
  "watch": "rollup -c --watch"  
},  
"dependencies": {  
  "lit": "^3.3.1"  
},  
"devDependencies": {  
  "@rollup/plugin-commonjs": "^28.0.9",  
  "@rollup/plugin-node-resolve": "^16.0.3",  
  "rollup": "^4.52.5"  
}
```

ADC25-Web-UIs/karplus_strong/ui

```
npm i
```

ADC25-Web-Uls/karplus_strong/ui

```
npm run watch
```

Let's try it!

Go to `karplus_strong/ui`

`npm i`

`npm run watch`

ui/controls/KnobComponent.js

```
class KnobComponent {} returns svg
```

ui/controls/AmplitudeDisplay.js

```
class AmplitudeDisplay {} draws canvas
```

ui/controls/view.js ⇒ render() {

```
class KarplusStrongInterface extends LitElement {}
```

article

`<article></article>`

– represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable. Examples include: a forum post, a magazine or newspaper article, or a blog entry, a product card, a user-submitted comment, an interactive widget or gadget, or any other independent item of content

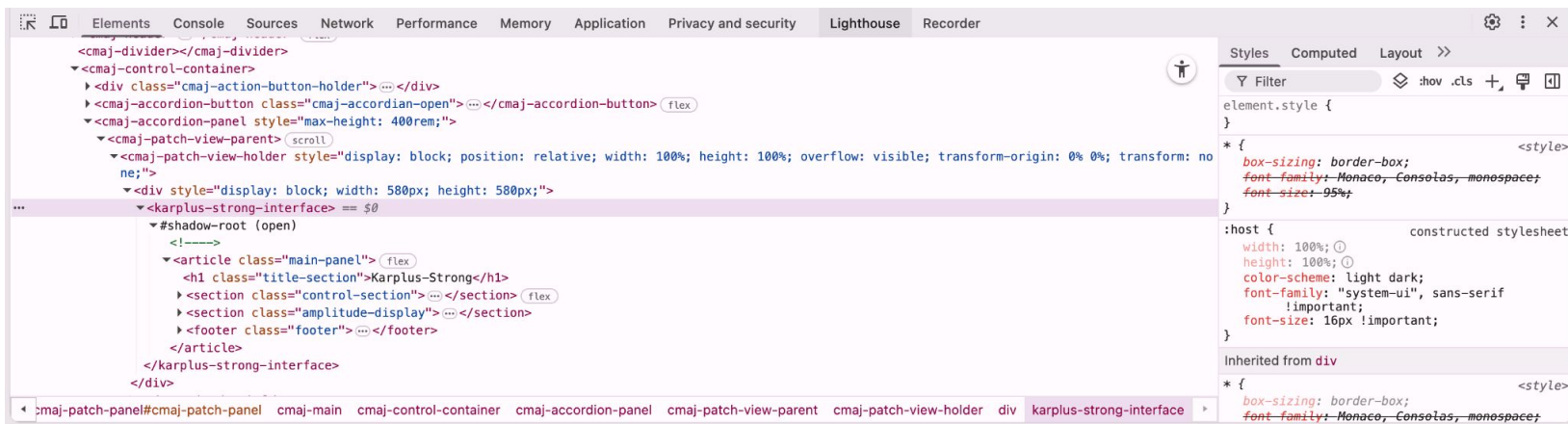
VS Code

CTRL/Cmd + shift + P
Run patch

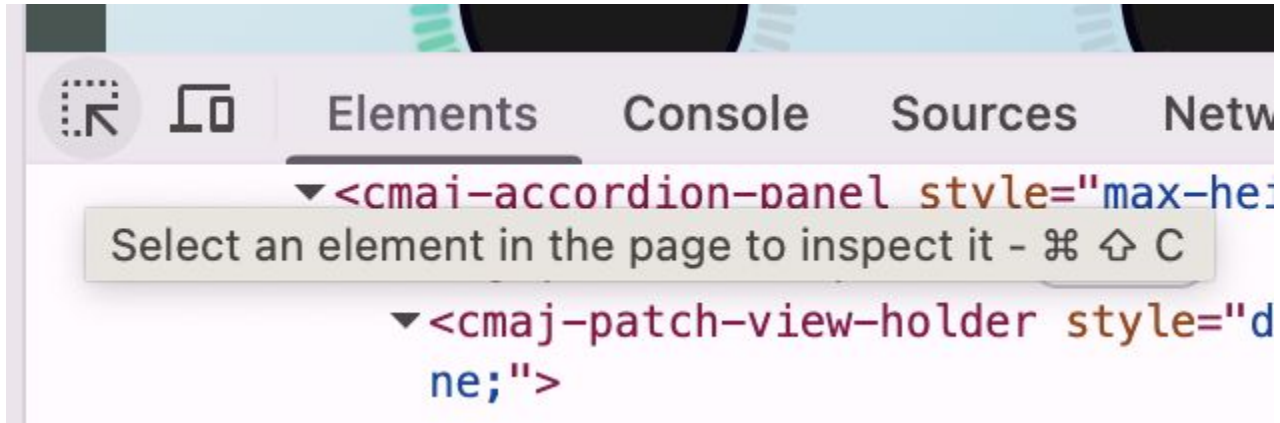


Browser

Right click on the element => inspect element



Browser



Check out branch stage1

```
git checkout stage1
```

ui/controls/view.js $\Rightarrow$ render(){}

```
<article class='main-panel'>  
  <h1 class='title-section'>Karplus-Strong</h1>  
</article>
```

ui/controls/view.js $\Rightarrow$ render(){}

```
<h1 class='title-section'>Karplus-Strong</h1>
```

```
<section class='control-section'>
```

```
</section>
```

ui/controls/view.js ⇒ render(){}

```
<section class='control-section'>
  <div class='knob-container'
id='impulse-knob-container'></div>
  <div class='knob-container'
id='filter-knob-container'></div>
  <div class='knob-container'
id='feedback-knob-container'></div>
</section>
```

ui/controls/view.js $\Rightarrow$ render(){}

```
<section class='control-section'>
```

```
...
```

```
</section>
```

```
<section class='amplitude-display'></section>
```

ui/controls/view.js ⇒ render(){}

```
<section class='amplitude-display'>  
  <canvas id="amplitude-canvas"  
class="amplitude-canvas" width="${this.CANVAS_WIDTH}"  
height="${this.CANVAS_HEIGHT}"></canvas>  
</section>
```

ui/controls/view.js ⇒ render(){}

...

</section>

<footer class="footer">

</footer>

</article>

ui/controls/view.js ⇒ render(){}

```
<picture>
  <source srcset="ui/public/dynamic_cast_logo_light.png"
media="(prefers-color-scheme: light)" />
  <source srcset="ui/public/dynamic_cast_logo_dark.png"
media="(prefers-color-scheme: dark)" />
  
</picture>
```

[Optional] Check out branch stage2

```
git checkout stage2
```

CSS

CSS
IS
AWESOME

CSS

Cascading **S**tyle **S**heets

— stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS

```
.header {  
    background-color: #ffffff;  
}
```

Box Model

Block boxes

Width, height properties are respected, take 100% of available width by default.

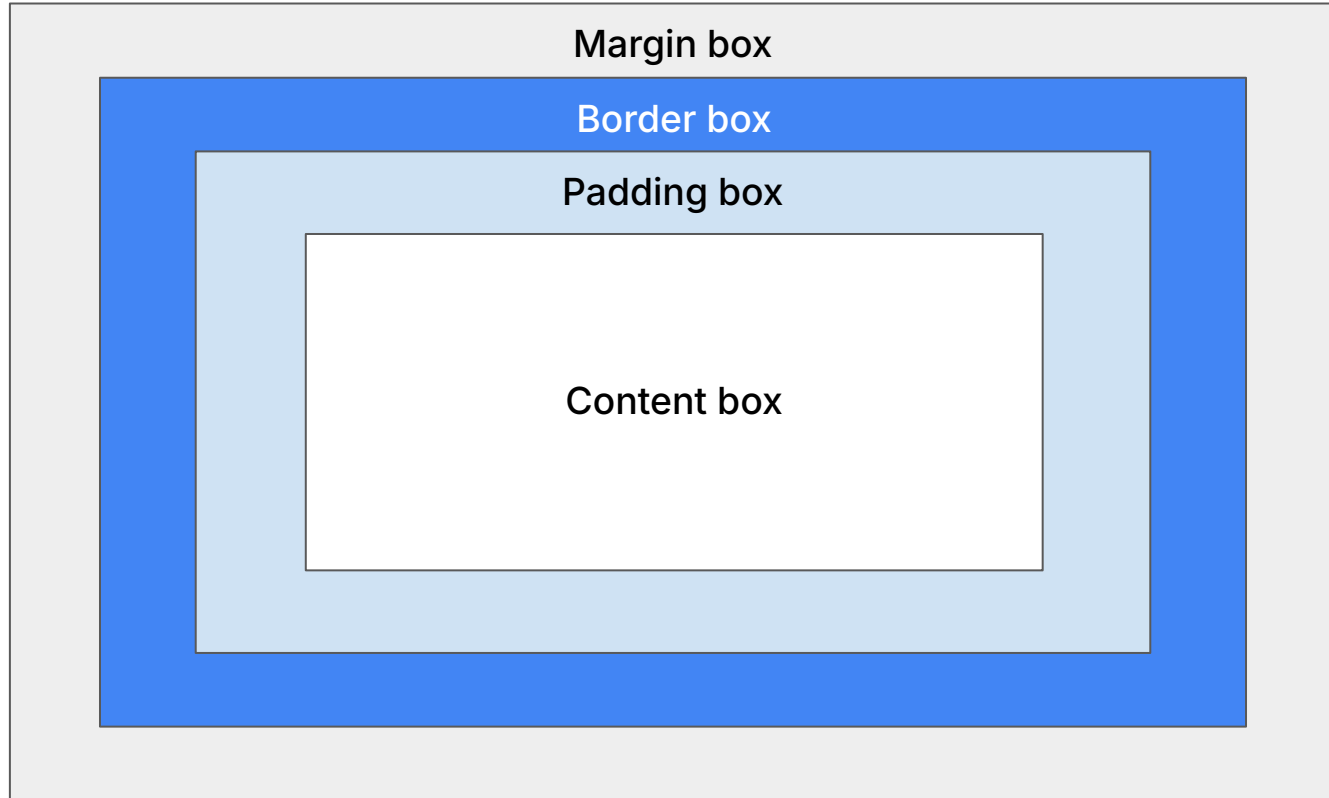
`<p>`, `<h1>`, `<div>`

Inline boxes

Width, height, top and bottom margins will have no effect, the box will not break onto a new line.

`<a>`, `<strong>`, `<em>`

Box Model



```
static styles = css`
```

```
* {  
  box-sizing: border-box;  
}
```

Selectors

`*` – universal selector (wildcard)

`button`, `section`, `h1` – type selectors

`.main-panel` – class selector

`#my-id` – id selector

`[href^="https"]` – attribute selector

```
static styles = css`
```

```
* {  
  box-sizing: border-box;  
  user-select: none;  
}
```

Dimensions: absolute

- px;
- unitless: 0.5 for opacity;
- percentages;
- cm – to print documents;
- many more;

	Recommended	Occasional use	Not recommended
Screen	em, px, %	ex	pt, cm, mm, in, pc
Print	em, cm, mm, in, pt, pc, %	px, ex	

Dimensions: relative

- `em` and `rem`: related to font-size;
- `vw`: 1% of viewport's width;
- `vh`: 1% of viewport's height;
- `lvw`: large viewpoint, relative to the viewport's visible space with all the optional browser UI hidden;

```
static styles = css`
```

```
:host {  
  width: 100%;  
  height: 100%;  
}
```

Pseudo-classes

- Element display state (:fullscreen, :picture-in-picture)
- Input (:disabled, :checked)
- Location (:visited, :link)
- Resource state (:paused, :muted)
- Tree structure (:root, :first-child)
- Shadow structure (:host)
- User action (:hover, :focus)
- Functional (:has(), :not())

```
static styles = css`
```

```
.main-panel {  
  width: 100%;  
  height: 100%;  
}
```

Display

Outer display type: whether it's inline or block, how it behaves with other elements (flow layout).

Inner display type: change layout of children elements.

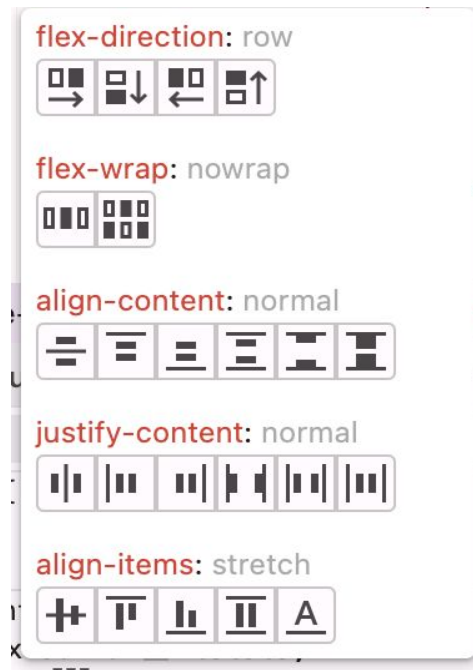
Normal flow of the elements: for English top to bottom, left to right.

Display: flex

CSS flexbox defines a CSS box model, the layout of items in one dimension.

Respects writing mode, can be a column or a row.

Has properties to align items on the main axis and the cross axis.



```
static styles = css`
```

```
.main-panel {  
  ...  
  display: flex;  
  flex-direction: column;  
  justify-content: space-evenly;  
}
```

Display: flex vs grid

Flex

- One dimension
- Easy to understand
- Simple layouts
- Allows items to grow or shrink if needed or wrap on a new line
- Content shapes layout

Grid

- Two dimensions
- More complex syntax (learning curve)
- Flexible for complex layouts and layouts that change depending on the screen size

Text

font-family

font-size

font-weight

text-align

text-transform

```
static styles = css`
```

```
:host {  
  ...  
  font-family: 'system-ui' !important;  
  font-size: 16px !important;  
}
```

```
static styles = css`
```

```
.main-panel {  
  ...  
  padding: 0.75rem;  
  border-radius: 0.75rem;  
  text-transform: uppercase;  
}
```

Cascade

1. Position and order of appearance: the order of which your CSS rules appear
2. Specificity: an algorithm which determines which CSS selector has the strongest match
3. Origin: the order of when CSS appears and where it comes from, whether that is a browser style, CSS from a browser extension, or your authored CSS
4. Importance: some CSS rules are weighted more heavily than others, especially with the !important rule type

Cascade: Origin Type

Origin Type

User-Agent

browsers, have
basic stylesheets
that give default
styles to any
document

Author

styles written by
web developers.
These styles can
reset user-agent
styles

User

the user of the
website can choose
to override styles
using a custom
user stylesheet

Cascade: order

```
.class {  
    background-color: red;  
}
```

```
.class {  
    background-color: blue;  
}
```

Cascade: Specificity

A: id-like specificity

B: class-like specificity

C: element-like specificity

$(1, 0, 0)$ vs $(0, 4, 3)$

* has $(0, 0, 0)$

Inheritance

```
body {  
    color: red;  
}
```

Color, font-family, text-align

Colour: rgb

color: **rgb**(34, 12, 64); 0 ... 255

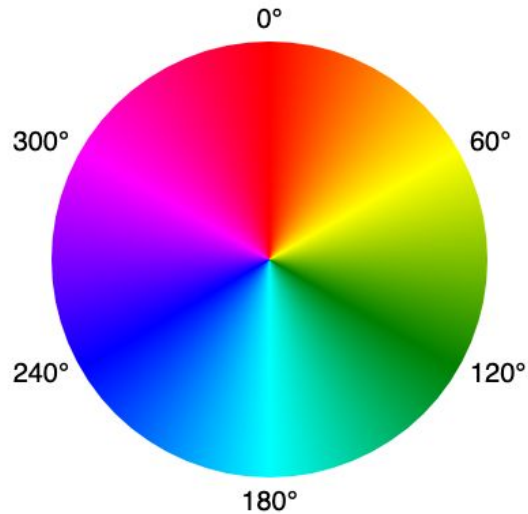
color: **#090**; 0 ... f

color: **#009900**; 00 ... ff

Colour: HSL

HSL: hue [angle], saturation (0-100%), and lightness (0-100%)

`hsl(120deg 75% 25%)`



High definition colours

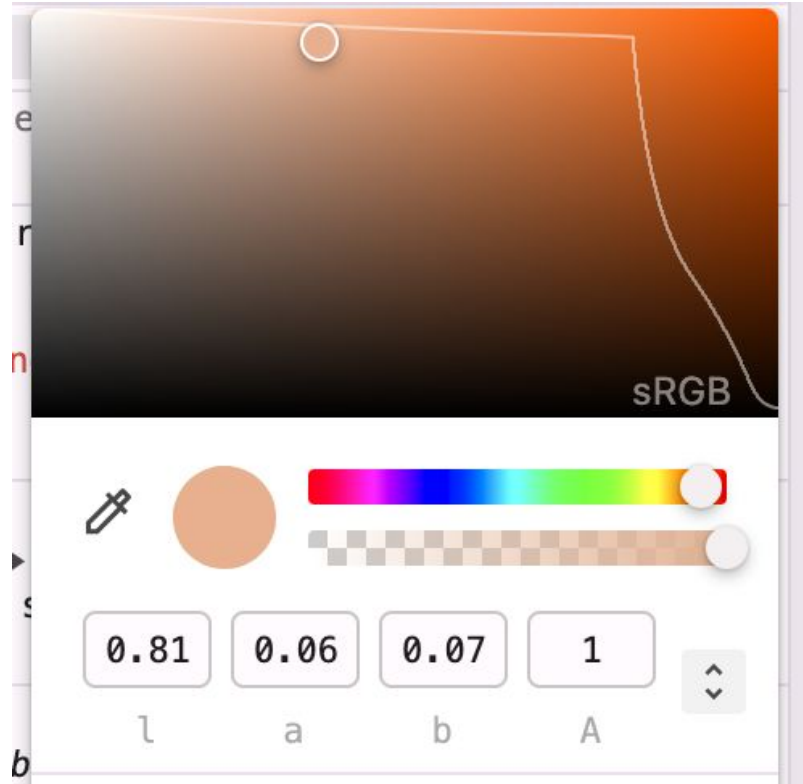
`oklab(40.1% 0.1143 0.045)`

lightness,

a-axis (from green to red, -0.4
to 0.4 or %)

b-axis (from blue to yellow, -0.4
to 0.4 or %)

50% more colours



```
static styles = css`
```

```
.main-panel {  
  ...  
  background: linear-gradient(145deg, #2a2a2a,  
    #1a1a1a);  
  box-shadow: 0 0.75rem 3rem rgb(0,0,0,0.3);  
}
```

```
static styles = css`
```

```
.control-section {  
  width: 100%;  
  display: flex;  
  justify-content: space-evenly;  
}
```

```
static styles = css`
```

```
.title-section {  
  margin: 0;  
  color: #fff;  
  font-size: 1.5rem;  
  text-align: center;  
}
```

```
static styles = css`
```

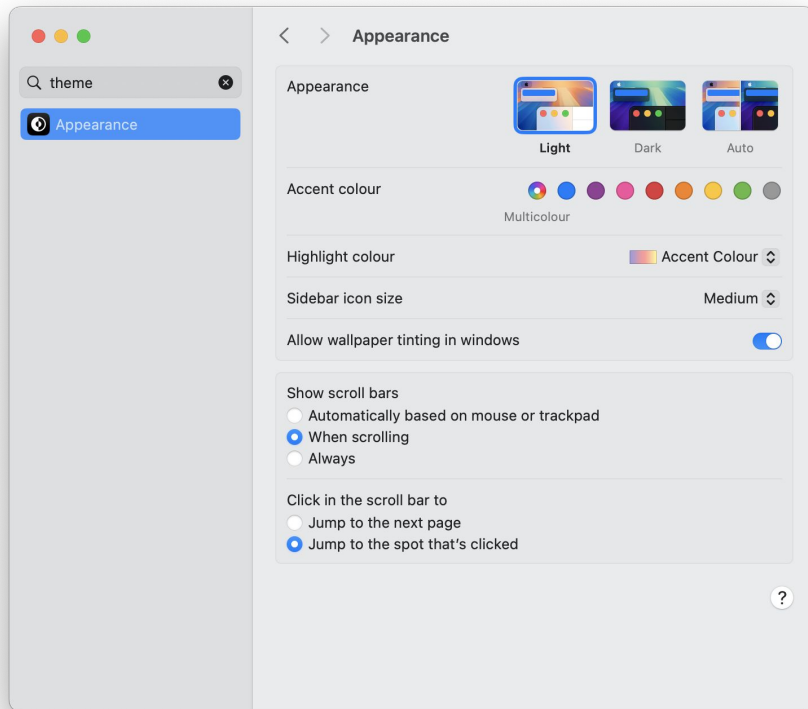
```
.control-label {  
  color: #ccc;  
  text-align: center;  
  font-size: 0.75rem;  
}
```

```
static styles = css`
```

```
.footer {  
  text-align: center;  
  margin-top: 2.8rem;  
}
```

Themes

```
color: light-dark(#000, #fff);
```



```
static styles = css`
```

```
:host {  
  color-scheme: light dark;  
}
```

```
static styles = css`
```

```
.main-panel {  
  --gradient: linear-gradient(  
    145deg,  
    light-dark(#dfecee, #2a2a2a),  
    light-dark(#84cce9, #1a1a1a)  
  );  
}
```

```
static styles = css`
```

```
.main-panel {  
  background: var(--gradient);  
}
```

```
static styles = css`
```

```
.title-section {  
  color: light-dark(#000, #fff);  
}
```

```
static styles = css`
```

```
.control-label {  
  color: light-dark(#4b4747, #cccccc);  
}
```

Cmajor

Cmajor

Cmajor is a C-styled language that is designed for DSP signal processing code.

Aims:

- To match/beat the performance of C/C++
- Be simple to learn
- Make code portable across processor architectures (CPUs, GPUs, DSPs, etc)

Uses an LLVM JIT compiler, to optimise and hot reload code

More info: <https://cmajor.dev/>



Keywords / Ideas

Cmajor Patch

- A bundle/ folder containing metadata files, program files and other resources (GUI scripts, audio files, etc)

Keywords / Ideas

Processors and Graphs

- The main high-level structures in Cmajor.
- A Graph declares a set of nodes, and a list of connections between them
 - These nodes can be processor objects or other graphs
- A Processor contains functions to perform mathematical operations.
 - Processors declare a main() function, containing a loop which reads its inputs, performs some processing, writes to outputs and repeats.

Example Graph

```
graph TwoGainsInSeries [[main]]
{
    // This section declares the inputs and outputs of the graph:
    input  stream float in;
    output stream float out;

    // This section declares the nodes:
    node gain1 = GainProcessor; // declare two nodes, each one a GainProcessor
    node gain2 = GainProcessor;

    // And here we list the connections between the nodes:
    connection in -> gain1 -> gain2 -> out;
    // send our input through both gains, and the result to our output
}
```

Example Processor

```
processor GainProcessor
{
    input  stream float in;    // declare an input and output stream of floats
    output stream float out;

    void main()
    {
        loop    // infinite loop
        {
            out <- in * 0.5f; // read our next input value, multiply by 0.5, and send it to
our output
            advance();        // advance to the next frame
        }
    }
}
```

.cmajorpatch File

Cmajor patches are a format for bundling together code and other resources, so that they can be loaded into audio hosts such as DAWs, to provide instruments or effect plugin functionality.

.cmajorpatch File

```
1  {
2    "CmajorVersion": 1,
3    "ID": "dev.dynamic-cast.workshops.adc25",
4    "version": "1.0",
5    "name": "Karplus-Strong Synthesizer",
6    "description": "A polyphonic Karplus-Strong string synthesis implementation",
7    "category": "generator",
8    "manufacturer": "Dynamic Cast",
9    "isInstrument": true,
10
11    "source": "main.cmajor",
12
13    "view": {
14      "src": "ui/dist/index.js",
15      "width": 580,
16      "height": 580,
17      "resizable": false
18    }
19  }
```

.cmajorpatch File

There are a few required properties that a patch must define:

CmajorVersion - this is the version of Cmajor for which this patch was written

ID - a universally unique ID for the patch, which should be in the form of a reverse-URL that includes the company name/website.

version - a version number for your patch. This is just a string - there are no restrictions on its format.

name - a human-readable name for your patch

.cmajorpatch File

Other optional properties include:

description - a longer description that a host can display to its users

manufacturer - the name of you or your company

category - hosts will be given this string, but how they choose to interpret it will be host-dependent

isInstrument - if specified, this marks the patch as being an instrument rather than an effect. Some hosts may treat a plugin differently depending on this flag.

.cmajorpatch File

The source property in the manifest tells the host which .cmajor files to compile for the Cmajor source code. This property can either be a single string containing a file path (relative to the folder containing the patch), or an array of string filenames if there are multiple files.

```
10  
11 "source": "main.cmajor",  
12
```

.cmajorpatch File

Specifying a custom GUI for a patch

To add a custom GUI to your patch, your .cmajorpatch file must declare a view property, e.g.

```
13     "view": {  
14         "src": "ui/dist/index.js",  
15         "width": 580,  
16         "height": 580,  
17         "resizable": false  
18     }
```

The view property should contain a src property providing a relative URL to a javascript module. Cmajor expects a web component here.

Cmajor

More information: <https://cmajor.dev/>



🔍 Search Cmajor Documentation

Home



About Us

Licence



Getting Started



Examples



Welcome to Cmajor!

The programming language for writing fast, portable audio software.

You've heard of C, C++, C#, objective-C... well, *Cmajor* is a C-family language designed specifically for writing DSP signal processing code.

Installing Cmajor

<https://github.com/cmajor-lang/cmajor/releases>

Releases

Tags

Find a release

19 hours ago



cesaref



1.0.3029



da0533e

Compare

1.0.3029

Pre-release

Disable clang warning

► Assets 8



Installing Cmajor

<https://github.com/cmajor-lang/cmajor/releases>

Useful for:

- Command Line Tools
- VST3 Patch Host
- Test Suite

Installing Cmajor

<https://github.com/cmajor-lang/cmajor/releases>

Useful for:

- Command Line Tools
- VST3 Patch Host
- Test Suite

```
zsh: Command not found: cmaj
harriet.drury@H6TW91HX54 ~ % ./cmaj

Cmajor Tools (C)2024 Cmajor Software Ltd.
https://cmajor.dev
Cmajor Version: 1.0.3019
Build date: Oct 19 2025 16:27:39

cmaj <command> [options]

Runs the given command. Options can include the following:

--0011121314      Set the optimisation level to the given value
--debug            Turn on debug output from the performer
--sessionID=n      Set the session id to the given value
--eventBufferSize=n Set the max number of events per buffer
--engine=<type>    Use the specified engine - e.g. llvm, webview, cpp
                  When using cpp, the additional options --overrideCompiler=<v>,
                  --extraCompileArgs=<v> and --extraLinkerArgs=<v> can be specified to
                  alter the compiler and arguments used
--simd             WASM generation uses SIMD/non-SIMD at runtime (default)
--no-simd          WASM generation does not emit SIMD
--simd-only        WASM generation only emits SIMD
--worker=<webview|quickjs> Specify the type of javascript engine to use for patch workers
```

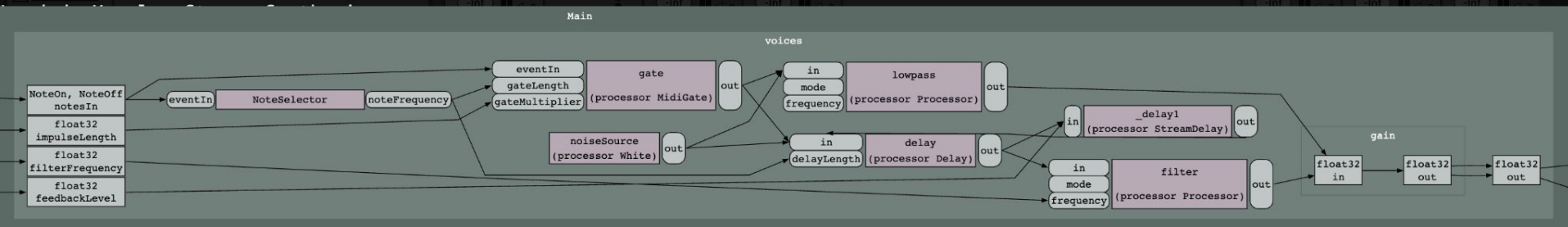
Installing Cmajor

<https://github.com/cmajors-lang/cmajors/releases>

Useful for:

- Command Line Tools
- VST3 Patch Host
- Test Suite

```
harriet.drury@H6TW91HX54 ~ % ./cmaj generate --target=graph /Users/harriet.drury/ADC25-Web-UIs/karplus_strong/main.cmajorpatch /Users/harriet.drury/ADC25-Web-UIs/main.html
```

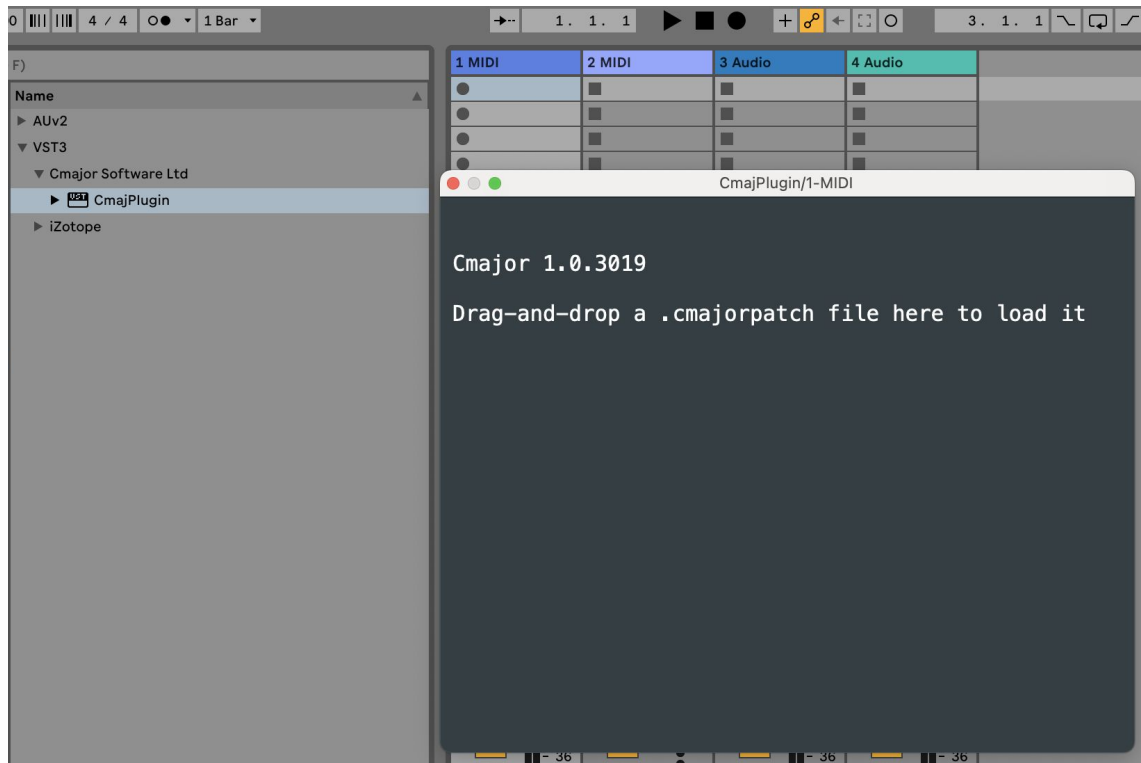


Installing Cmajor

<https://github.com/cmajors-lang/cmajors/releases>

Useful for:

- Command Line Tools
- VST3 Patch Host
- Test Suite

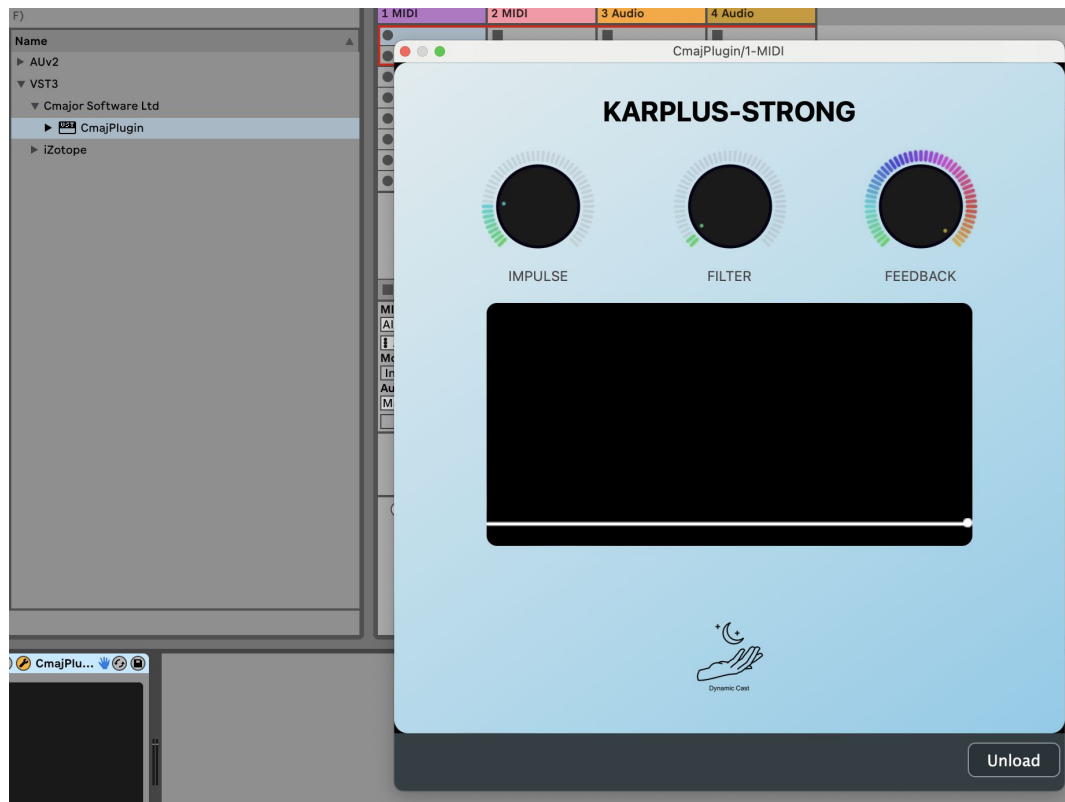


Installing Cmajor

<https://github.com/cmajor-lang/cmajor/releases>

Useful for:

- Command Line Tools
- VST3 Patch Host
- Test Suite



VSCode Cmajor Extension



Cmajor Tools

Cmajor Software Ltd

2,285

★★★★☆ (1)

Cmajor language tools

Disable

Uninstall



Auto Update

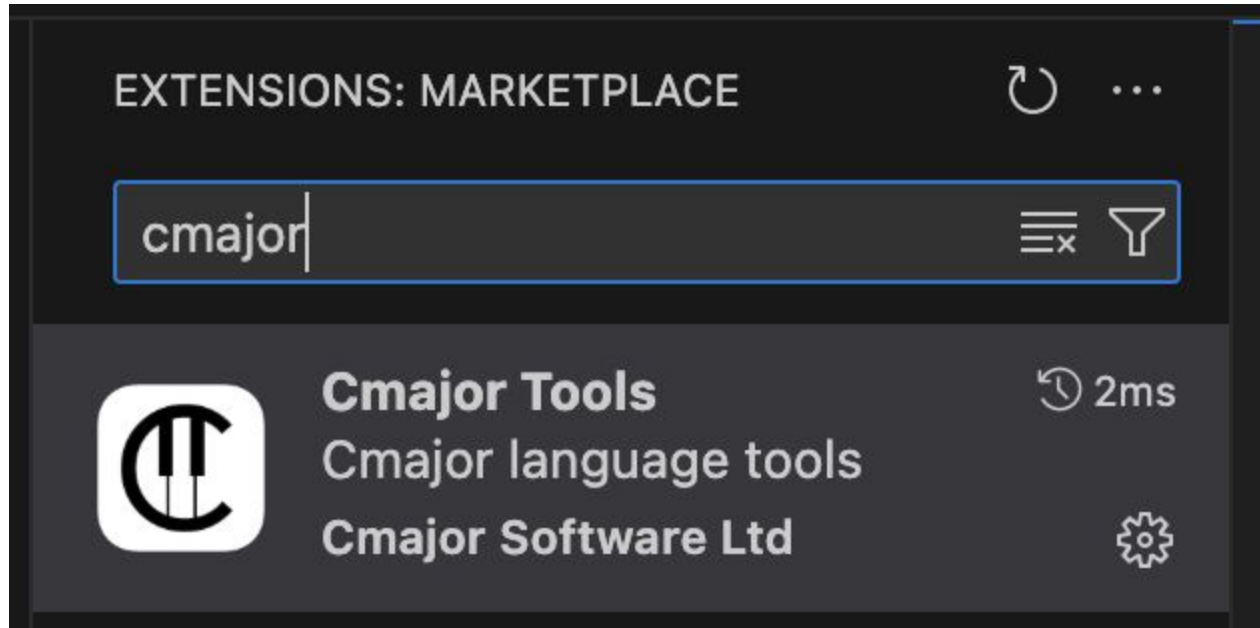


DETAILS

FEATURES

VSCode Cmajor Extension

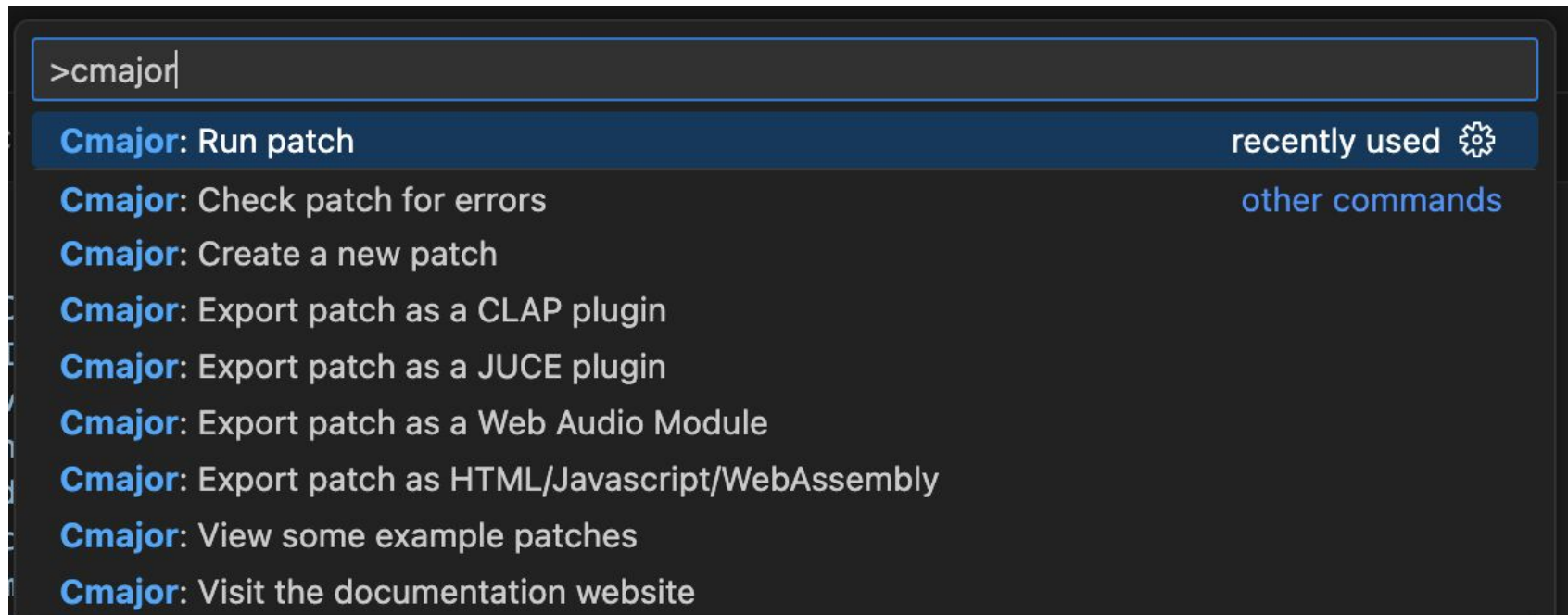
- Open VSCode
- Open Extensions
- Search Cmajor



Cmajor Commands

Open the command palette with:

CTRL/ CMD + Shift + P



Cmajor Commands

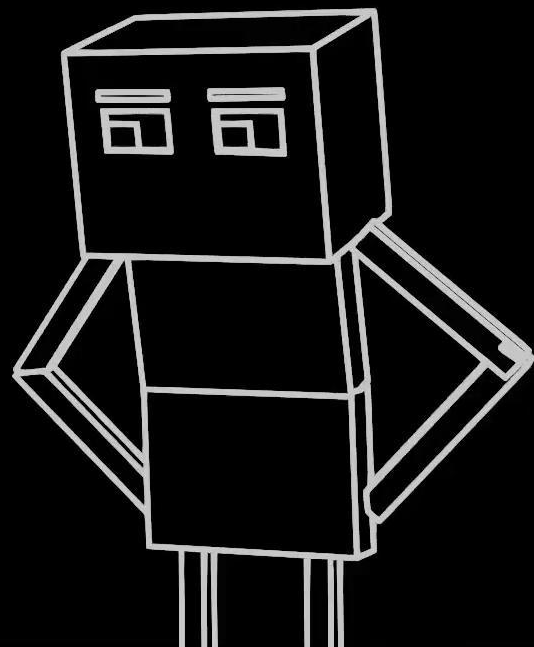
The screenshot shows the Cmajor Karplus-Strong Synthesizer interface within a web browser. The browser's address bar displays "ADC25-Web-UIs". The interface is divided into several sections:

- EXPLORER (Left Panel):** Lists the project structure, including folders like "include", "karplus_strong", and "ui", and files like "Karplus Strong.md", "main.cmajor", "main.cmajorpatch", "physical-modelling.png", "tests", ".gitignore", ".gitmodules", "package-lock.json", and "README.md".
- Extension: Cmajor Tools:** Shows the loaded patch "main.cmajorpatch".
- Cmajor: Karplus-Strong Synthesizer (Main Panel):**
 - Header:** Displays the Cmajor logo, version "1.0.3029", and a "Stop Audio" button. A status bar indicates "CPU: 0.4%".
 - Description:** A polyphonic Karplus-Strong string synthesis implementation. Version: 1.0. Manufacturer: Dynamic Cast. Category: generator. Main processor: Main. Audio: 1 output channel. MIDI: 1 input. Parameters: 3.
 - Controls:** Buttons for "Reset patch", "Copy state to clipboard", and "Paste state from clipboard".
 - GUI:** A section with a "generic" dropdown menu.
 - Karplus-Strong Synthesizer:** The main control area featuring three knobs labeled "length", "500.00", and "feedback".
 - Inputs:** A section at the bottom showing a piano roll with white keys on a dark background.

To VSCode...

Let's look at a Cmajor patch

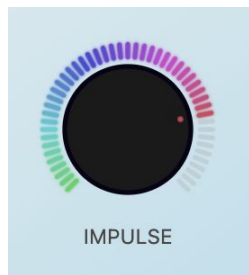
Karplus-Strong string synthesis



Connecting the UI

Connecting parameters to the UI

The UI representation of parameter controls will always need to get connected to the parameters defined in the audio processors.

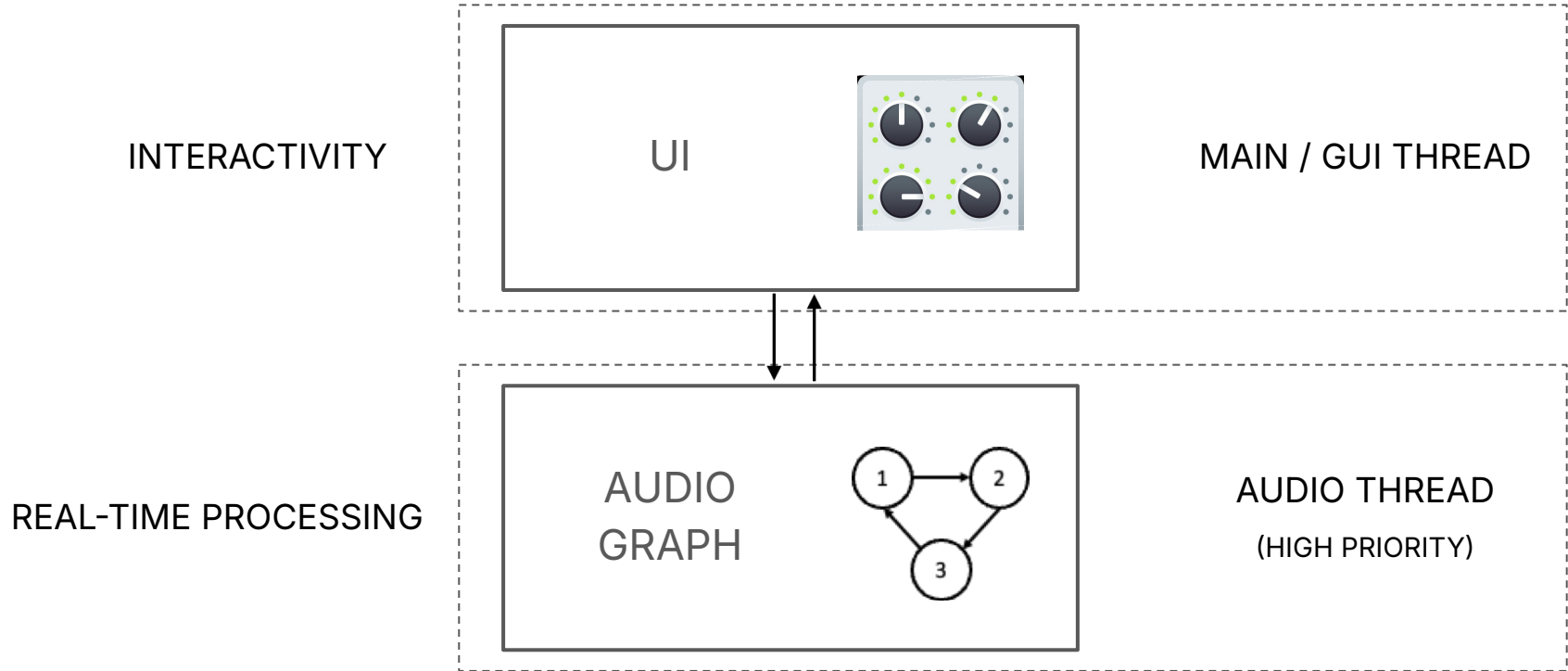


KnobComponent (KnobComponent.js)



```
input event float impulseLength [[ name: "length", min: 0.1f, max: 5.0f, init: 1.0f]];
                                (main.cmajor)
```

Two-way communication



Two-way communication

Reading / changing parameter values needs to be synchronised across threads to ensure data consistency:

- reading a value, modifying it, and writing it back may involve multiple steps that can be interrupted by other threads
- if a value is being read while it's being changed, we may end up with inconsistent data

Luckily, here the framework is taking care of it for us.

Two-way communication

In CMajor, the web UI can communicate with the running patch through the [PatchConnection](#) object.

The object is provided by the CMajor runtime (the command line patch player, a Cmajor plugin loader, or - in our case - VScode extension) as an argument to a function called to create a view for our patch.

The function needs to be the default export of the javascript module pointed to in the manifest file (`main.cmajorpatch`).

PatchConnection

`/* This is the function that a host (the command line patch player, or a Cmajor plugin loader, or our VScode extension, etc) will call in order to create a view for your patch.`

`Ultimately, a DOM element must be returned to the caller for it to append to its document. However, this function can be `async` if you need to perform asynchronous tasks, such as fetching remote resources for use in the view, before completing.`

```
*/  
export default function createPatchView (patchConnection)  
{  
    return new my_View (patchConnection);  
}
```

PatchConnection
object passed by host



Return an HTMLElement



PatchConnection

PatchConnection provides a range of methods for controlling and querying the state of the patch.

- Updating parameter values:
 - **sendEventOrValue (endpointID, value, rampFrames)** - update value; optional ramp parameter specifies number of frames over which the current value ramps to the new one
 - **sendParameterGestureStart (endpointID)** - tells the patch that a series of changes (gesture) is about to take place for the given endpoint
 - **sendParameterGestureEnd (endpointID)** - tells the patch the gesture has finished

Wrapping parameter changes in a gesture ensures the host recognizes them as a single used action, which is important for features like recording automation or undo/redo.

PatchConnection

- Listening to parameter changes:
 - **addParameterListener (endpointID, listener)** - attaches a listener function which will receive updates with a new value whenever a given parameter's value changes.
 - **removeParameterListener (endpointID, listener)** - removes a listener
 - **requestParameterValue (endpointID)** - triggers an asynchronous callback to any parameter listeners that are attached, providing them with an up-to-date current value for the given endpoint

Receive the current parameter value:

```
patchConnection.addParameterListener (endpointID, callback);  
patchConnection.requestParameterValue (endpointID);
```

PatchConnection

- Listening to changes of events or audio data:
 - **addEndpointListener (endpointID, listener, granularity)** - attaches a listener function which will receive updates with the **events or audio data** that is being sent or received by an endpoint. If the endpoint has the right shape to be treated as "audio" then the callback will receive a stream of updates of chunks of data that is flowing through it. There will be one callback per chunk of data, and the size of chunks is specified by the optional granularity parameter.
 - **removeEndpointListener (endpointID, listener)** - removes a listener

PatchConnection

- Getting remaining parameter information

In order to correctly display the parameter **range** in the UI, we need to query the parameter's **min** and **max** values from the patch. Such information is stored as *endpoint description* and is part of patch's *status*. The status information can be obtained through the following API:

- **addStatusListener (listener)** - attaches a listener function which will be called when the patch's status changes. The current status information will be passed as a parameter.
- **removeStatusListener (listener)** - removes a listener that was previously added

Patch status

The patch status contains many properties describing the current state, including:

- patch manifest

```
"manifest": {  
  "CmajorVersion": 1,  
  "ID": "dev.dynamic-cast.workshops.adc25",  
  "version": "1.0",  
  "name": "Karplus-Strong Synthesizer",  
  "description": "A polyphonic Karplus-Strong string synthesizer",  
  "category": "generator",  
  "manufacturer": "Dynamic Cast",  
  "isInstrument": true,  
  "source": "main.cmajor",  
  "view": {  
    "src": "ui/dist/index.js",  
    "width": 580,  
    "height": 580,  
    "resizable": false  
  }  
},
```

Patch status

- endpoint descriptions
(truncated output)

```
"details": {  
  "mainProcessor": "Main",  
  "mainProcessorLocation": "/path/to/proj/main.cmajor:1:7:",  
  "inputs": [  
    {  
      "endpointID": "filter",  
      "endpointType": "event",  
      "dataType": {  
        "type": "float32"  
      },  
      "annotation": {  
        "name": "filter",  
        "min": 100,  
        "max": 10000,  
        "init": 500  
      },  
      "purpose": "parameter",
```

Patch status

- endpoint descriptions
(truncated output)

```
"outputs": [  
  {  
    "endpointID": "amplitude",  
    "endpointType": "event",  
    "dataType": {  
      "type": "float32"  
    },  
    "annotation": {  
      "name": "amplitude",  
      "min": 0,  
      "max": 1  
    },  
    "source": "/path/to/proj/main.cmajor:5:18:"  
  }  
]
```

Patch status

- other metadata

```
"warning": "",  
"sampleRate": 44100,  
"host": "Cmajor Player"
```

Connecting patch data to the UI

Let's use the PatchConnection API to connect
our patch data to the UI

Back to the code

Connecting patch data to the UI

In `createPatchView` of `index.js`

```
patchConnection.addStatusListener ((currentStatus) => {  
  });
```

```
patchConnection.requestStatusUpdate();
```

Connecting patch data to the UI

body of `patchConnection.addStatusListener`:

```
const createParameterBinding = (targetEndpointID) => {  
  const endpointDetails = currentStatus?.details?.inputs?.find(  
    ({ endpointID }) => endpointID === targetEndpointID);  
  
  if (!endpointDetails) return null;  
};
```

Connecting patch data to the UI

body of createParameterBinding, continued:

```
const { endpointID } = endpointDetails;
return {
  minValue: endpointDetails?.annotation?.min,
  maxValue: endpointDetails?.annotation?.max,
  defaultValue: endpointDetails?.annotation?.init,
  startGesture: () => patchConnection.sendParameterGestureStart (endpointID),
  updateValue: (newValue) => patchConnection.sendEventOrValue (endpointID, newValue),
  endGesture: () => patchConnection.sendParameterGestureEnd (endpointID),
};
```

Connecting patch data to the UI

body of createParameterBinding, continued:

```
endGesture: () => patchConnection.sendParameterGestureEnd (endpointID),  
attachListener: (callback) =>  
{  
    patchConnection.addParameterListener (endpointID, callback);  
    patchConnection.requestParameterValue (endpointID);  
  
    return () => patchConnection.removeParameterListener (endpointID, callback);  
}
```

Connecting patch data to the UI

body of `patchConnection.addStatusListener`, continued:

```
const createAmplitudeBinding = (targetEndpointID) => {  
  const endpointDetails = currentStatus?.details?.outputs?.find(  
    ({ endpointID }) => endpointID === targetEndpointID);  
  
  if (!endpointDetails) return null;  
  return {  
    attachListener: (callback) => {  
      patchConnection.addEndpointListener(endpointDetails.endpointID, callback);  
      return () => patchConnection.removeEndpointListener(  
        endpointDetails.endpointID, callback);  
    }  
  };  
};
```

Connecting patch data to the UI

Move `createKarplusView` to `patchConnection.addStatusListener`

```
const karplusUI = createKarplusView({  
  impulseLength: createParameterBinding("impulseLength"),  
  filterFreq: createParameterBinding("filter"),  
  feedbackAmount: createParameterBinding("feedback"),  
  amplitude: createAmplitudeBinding("amplitude"),  
});  
  
mainContainer.appendChild(karplusUI);
```

Running the patch in a DAW

<https://github.com/cmajor-lang/cmajor/releases/tag/1.0.3029>



Recap

- Use WebViews to make good-looking UIs!
- HTML is easy to understand, MDN for reference
- Box model
- Outer display: inline, block
- Inner display: flex, grid
- Colours: rgb, hsl and modern colour spaces

Resources

- [MDN web docs](#)
- [web.dev/learn](#)
- [codepen.io](#)
- [smashing magazine](#)
- [CSS tricks](#)
- [Evil martians on modern colours](#)
- [can I use](#)

Thank you