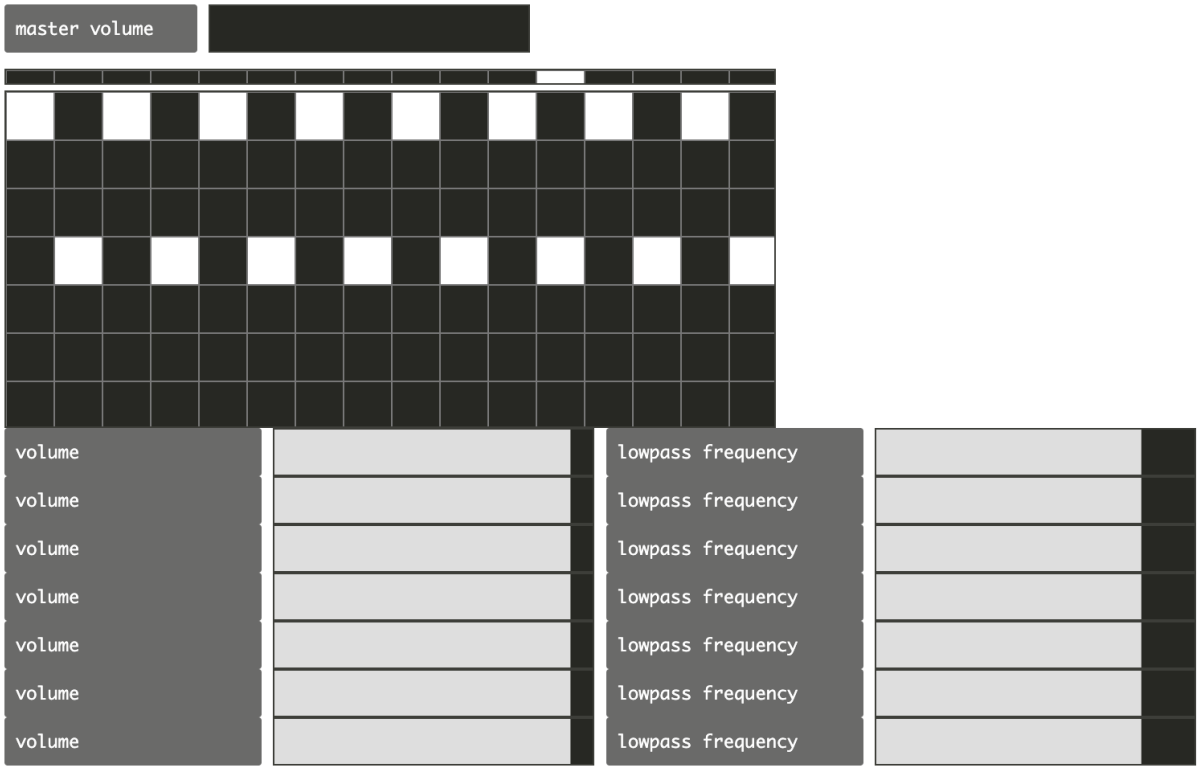
Project - Simple step sequencer



To conclude this introduction to the Web Audio API you will implement a simple step sequencer and propose some improvements. This project is built upon the *playing-audio-buffer* exercise and makes use of the *scheduler*.

The project is divided in two parts:

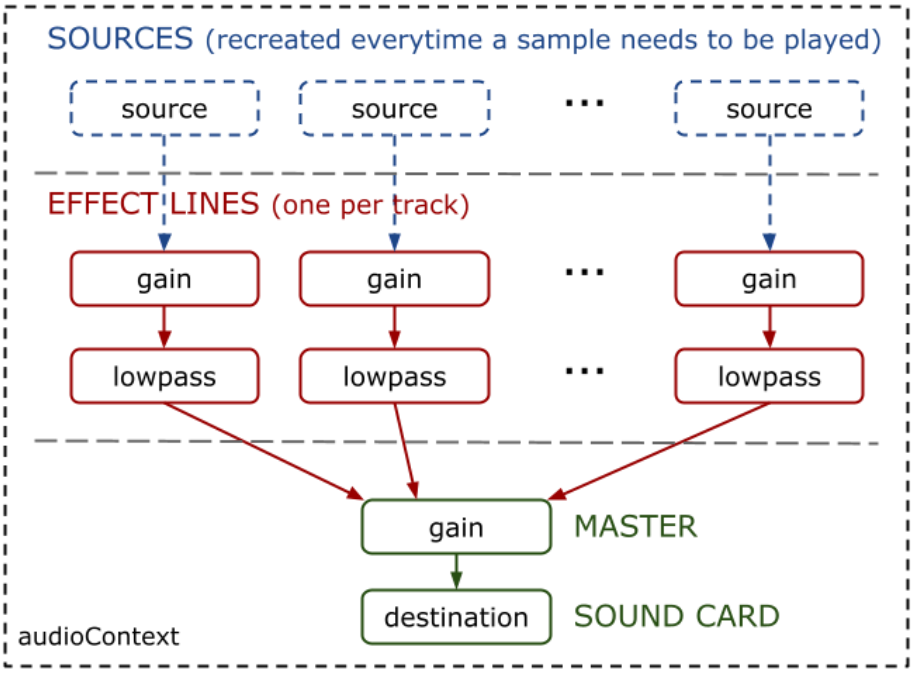
I. The basic implementation of the step sequencer: you are guided through the creation of a simple graph with a global volume slider, an effet line for each track, and the buffer source generations.
II. The improvement of the sequencer: you will improve this sequencer any way you want. We propose some possibilities but you are free to choose what you want to add or modify to make it your own.

You will find the project folder in the Git repository. Don't forget to install the dependencies:

cd ./path/to/project/step-sequencer
npm install
npm run dev

Since we only covered a tiny part of what is possible with JavaScript and the Web Audio API you will find anything that you need in the Mozilla Developer Network documentation at https://developer.mozilla.org/en-US/docs/Web, from JavaScript syntax for beginners to various Web Audio nodes that we didn't cover (and that could be needed).

I. Basic Step Sequencer Implementation (13 pts)



The step sequencer is made of one effect line **per** sound file (8 by default). Each is connected to a master volume connected to the sound card. Each effect line is composed of a gain and of a lowpass filter (a node that we didn't cover during the class but that is natively available in the Web Audio API).

The sequencer follows a *score* to decide which sample to play at each step. All code lines that generate and modify the score are already there, you just need to know that the score is a table filled with 0s or 1s, 1 meaning that the sample needs to be played at the given step.

Before starting with the questions, spend some time reading the comments, understanding the global variables that are already declared and the global structure of the document.

Master gain and effect lines (4 pts)

First, head to the *main()* function where we will create the end of the graph: the master gain and the effect lines.

Q1 Master (1 pts)

Create a gain called *master* and connect it to the audioContext destination

Q2 Effect lines (3 pts)

For each sound file, implement the *effect lines* part of the graph (in red). Notice that the *lowpass* and *gain* nodes are stored in the *effects* object.

Samples triggering and score following (6 pts)

Now, go to the *StepSquencer* class definition which already contains a constructor that doesn't require any changes. You will code the behavior of the step sequencer: what it does when it is called and when it needs to be called next.

Q3 BufferSource creations (4 pts)

Play each sound file that needs to be played at the given stepIndex. Take care that the sound is played through its own effect line.

Q4 Move forward in the score (2 pts)

Now that the right sounds were played at this step we need to move forward in the score at the right moment according to the bpm. Update the *stepIndex* and *return* the time at which the *advanceTime* method needs to be called next.

Beat follower track and sliders (4 pts)

Notice the small bar above the score on the first figure. It is a beat follower track that shows what step is being played. Its implementation is done in the *DisplaySteps* class.

Q5 Beat follower track (3 pts)

The modification of the bar is already coded in the *advanceTime* method. Now the GUI must be refreshed at the right moment (using the *renderGUI()* function). Similar to Q4, don't forget the necessary updates at the end of the method.

Q6 dbToLinear (1 pts)

Code the dbToLinear function that allows for sliders in dB.

II. Improvements (6 pts)

Now you are free to make as many improvements to this sequencer as you'd like. Depending on what you wish for, you might have to dig in the documentations for stuff we didn't cover together. Don't hesitate to reach us if you have any questions or don't know where to start with a particular idea. Be creative and have fun! Here are some ideas if you need inspiration:

- Make effect lines more complex or even create your own effect
- Add global effects to the master (filters, reverb…)

Feel free to work on the UI as much as you want but we require that you have at least one modification that uses the Web Audio API.

The script *index.js* must be sent before **January 22, 2023 at 23:59** to Benjamin.Matuszewski@ircam.fr, Alienor.Golvet@ircam.fr and Victor.Paredes@ircam.fr. Please rename the file to [first_name].[last_name].index.js so that we can keep track of who is who. You must comment your code to make it understandable by others. Please also join a pdf explaining what you did to the sequencer and how to use it if necessary.