21ST OCT 2025

Designing a digital synth/sequencer

using ReactJS, tone.js, state managements, etc

https://digi-seq.vercel.app/

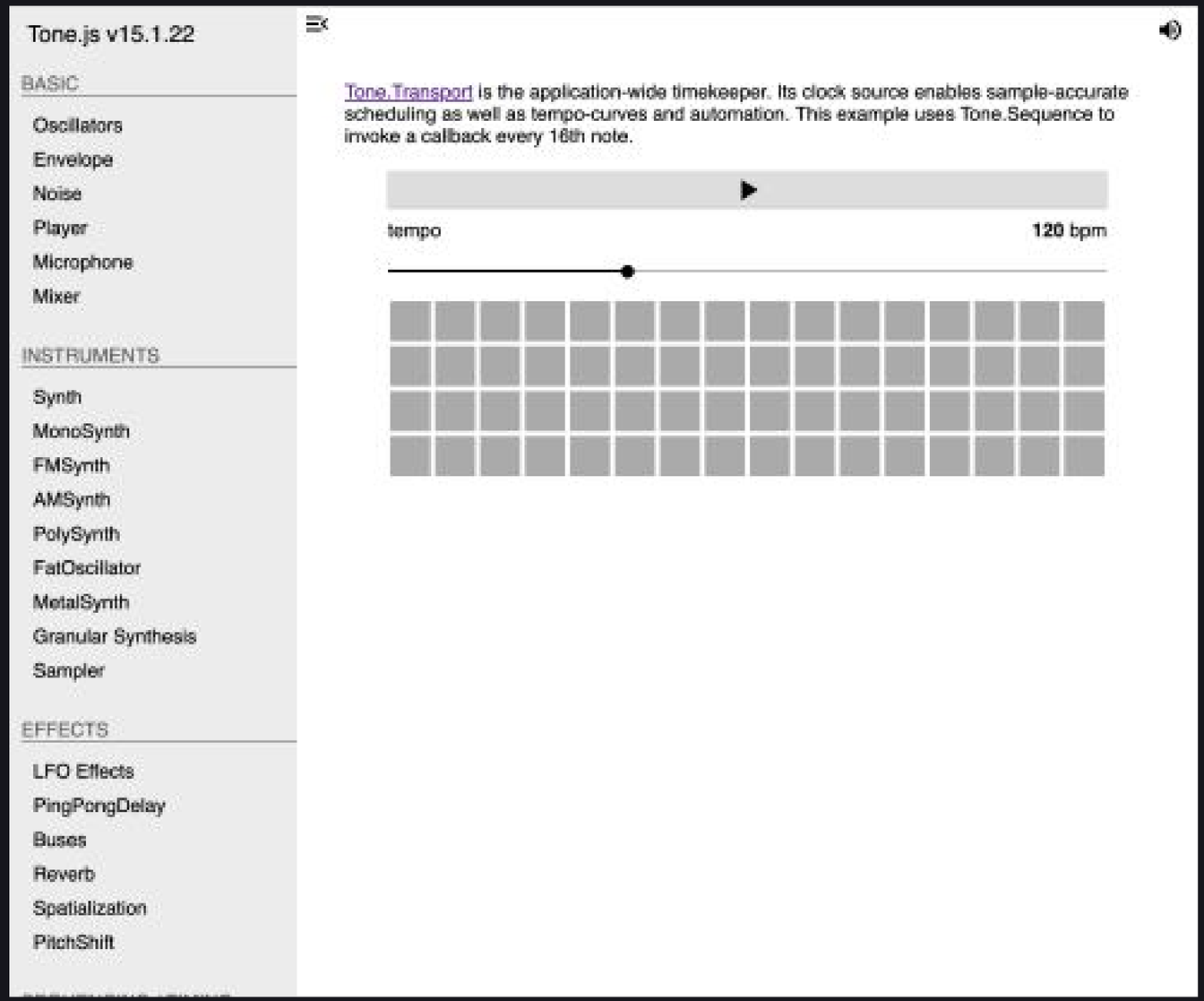
ISSAC TING

Reference

I wanted to create a similar set up as the example i found in Tone.js, but doing it in react.

Functions I included

- Tone.Transport.stop()
- Tone.Loop((time)
- Tone.start()



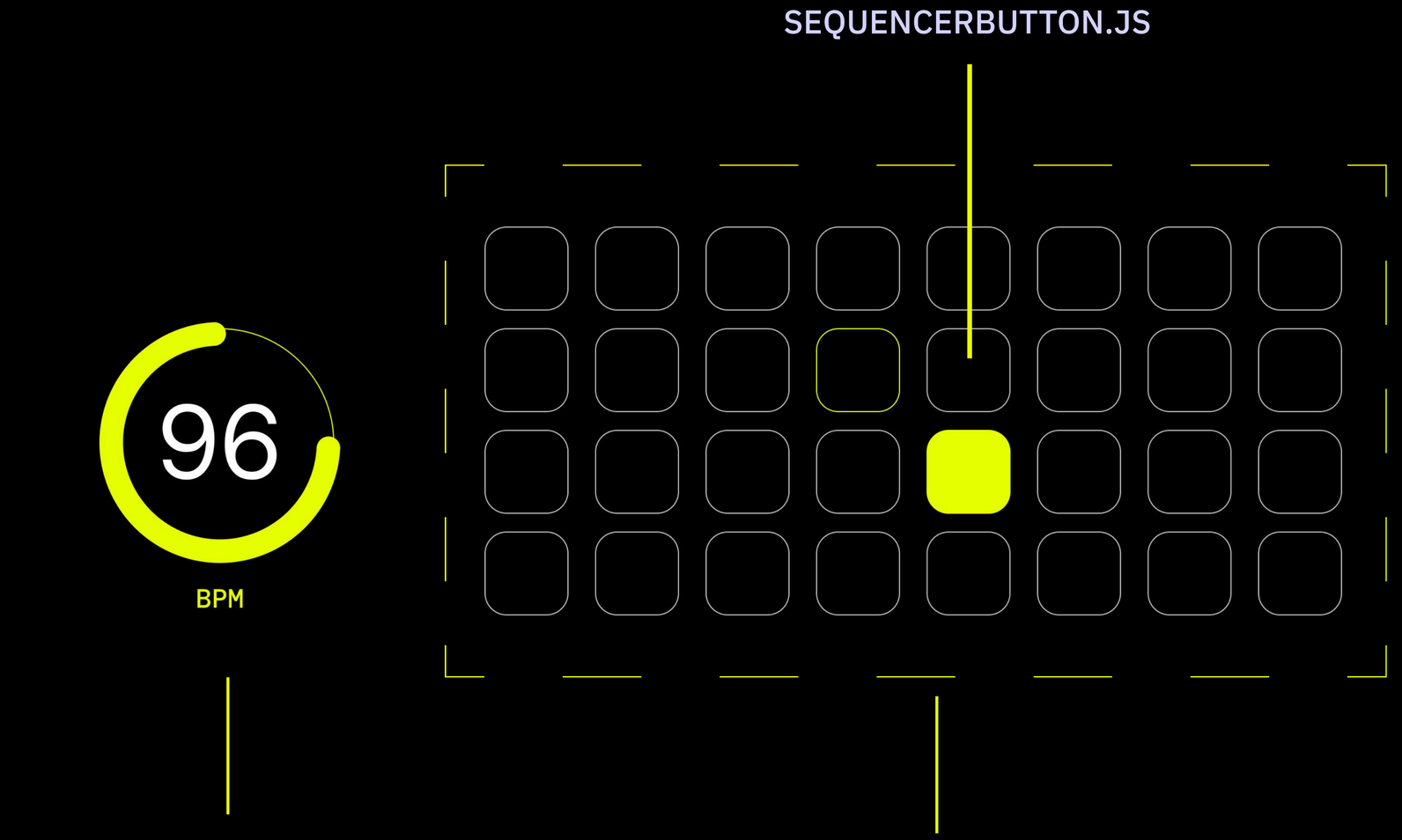
https://tonejs.github.io/examples/stepSequencer

Main Parts

- SequencerButton.js
- SequencerController.js
- SequencerGrid.js
- Pattern.js

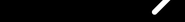
PLANNING

I started with some rough designs in Figma to figure out what I might want to do.



@FSEEHAWER/REACT-CIRCULAR-SLIDER


```
// bpm and play/stop controls
export default function
SequencerControls({
  isPlaying,
  handlePlay,
  handleStop,
  bpm,
  setBpm,
  minBpm = 60,
  maxBpm = 240,
})
```

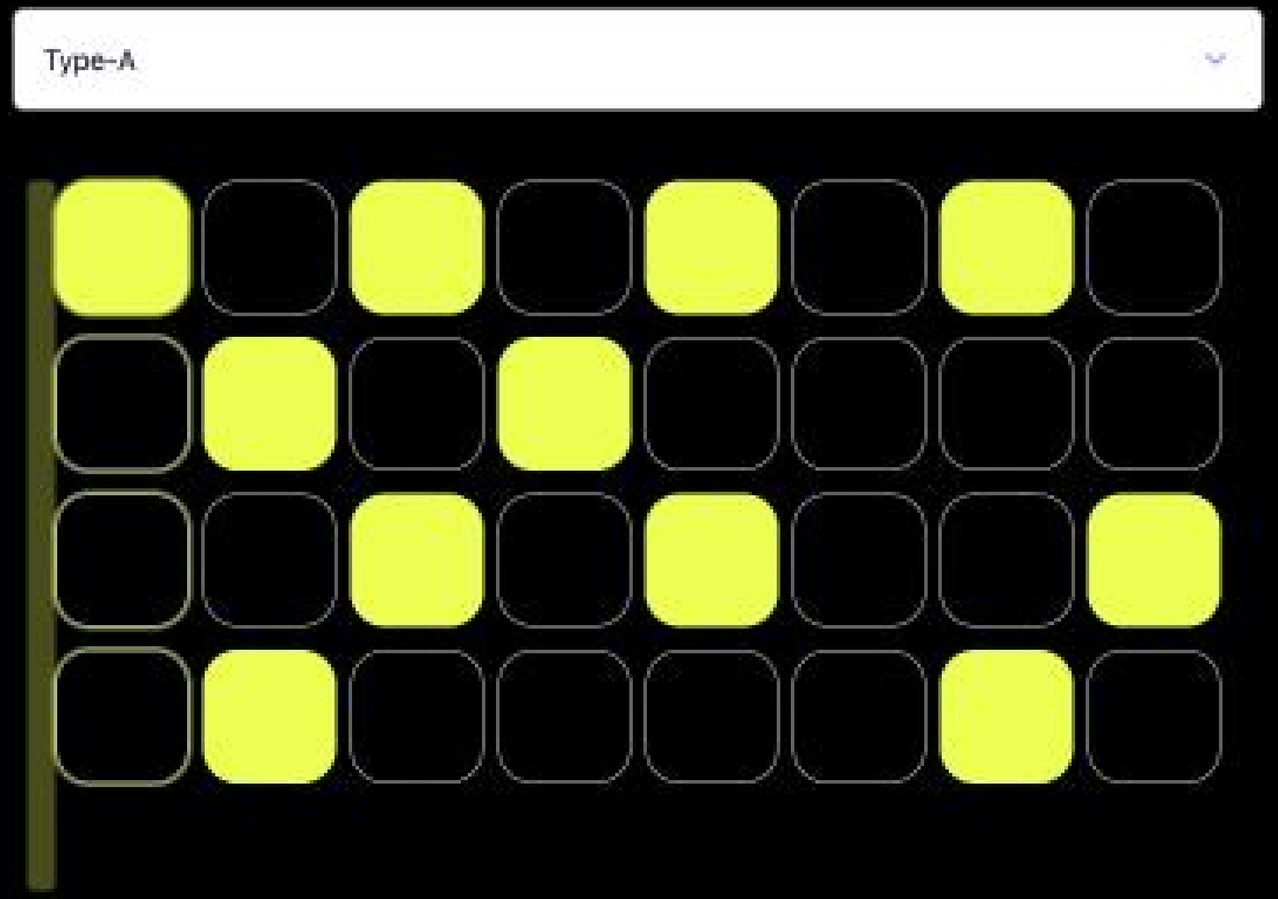


```
<SequencerControls
        isPlaying={isPlaying}
        handlePlay={play}
        handleStop={stop}
        bpm={bpm}
        setBpm={setBpm}
      />
```

Creating the grid

- Mapping a 2D Array

```
{grid && grid.map((row, rowIndex) => (
      <div key={rowIndex} className="flex gap-2 mb-2">
       {row.map((cell, cellIndex) => (
         <div key={cellIndex}>
          <SequencerButton
           isActive={cell}
           isCurrentStep={cellIndex === currentStep}
           onClick={() => onToggleCell(rowIndex, cellIndex)}
          />
         </div>
       ))}
      </div>
    ))}
```




```
<SequencerGrid
        grid={grid}
        currentStep={currentStep}
        onToggleCell={toggleStep}
      />
```

```
// keep grid updated
  useEffect(() => {
    gridRef.current = grid;
  }, [grid]);
```

← Get the latest update

Presets

I wanted to create different presets that you can select.

```
let [grid, setGrid] = useState(makePattern("basic"));
```

Holds the **current sequencer grid**

Function to update the grid state

Select the different patterns by passing a string into a state for Make Pattern.

```
export const patterns = {
  basic: [
    [1,0,0,0,0,0,0,0],
    [0,0,1,0,0,0,0,0],
    [0,0,0,0,1,0,0,0],
    [0,0,0,0,0,0,1,0]
  ],

  checkerboard: [
    [1,0,1,0,1,0,1,0],
    [0,1,0,1,0,1,0,1],
    [1,0,1,0,1,0,1,0],
    [0,1,0,1,0,1,0,1]
  ],
```

import { patternOptions, makePattern }
from "../data/patterns";

SEQUENCERPAGE.JS

```
// start playing
const play = async () => {
  await Tone.start();

  if (!synthRef.current) {
    synthRef.current = new Tone.PolySynth({
      oscillator: { type: "sawtooth" },
      envelope,
    }).connect(Tone.Master);
  }

  let step = 0;
  loopRef.current = new Tone.Loop((time) => {
    setCurrentStep(step);   // update UI first

    // check if the current step is on in the grid
    for (let row = 0; row < ROWS; row++) {
      if (gridRef.current[row][step]) {
        synthRef.current.triggerAttackRelease(CHORD_NOTES[row], "8n", time);  // play the chord
      }
    }

    step = (step + 1) % COLS;  // move to the next step
  }, "8n");

  setCurrentStep(0);  // reset the current step
  loopRef.current.start(0);  // start the loop
  Tone.Transport.start();  // start the transport
  setIsPlaying(true);  // set the playing state to true
};
```

Enables audio in the browser

Creates synthesizer only once

PolySynth - Can play multiple notes simultaneously. **Sawtooth** is the type of sound and **Envelope** controls the sound effect

Loops through the **grid** to see what notes to play

Move to next step

setCurrentStep(step); - Updates the visual indicator

ENVELOPE - HOW A SOUND CHANGES OVER TIME.

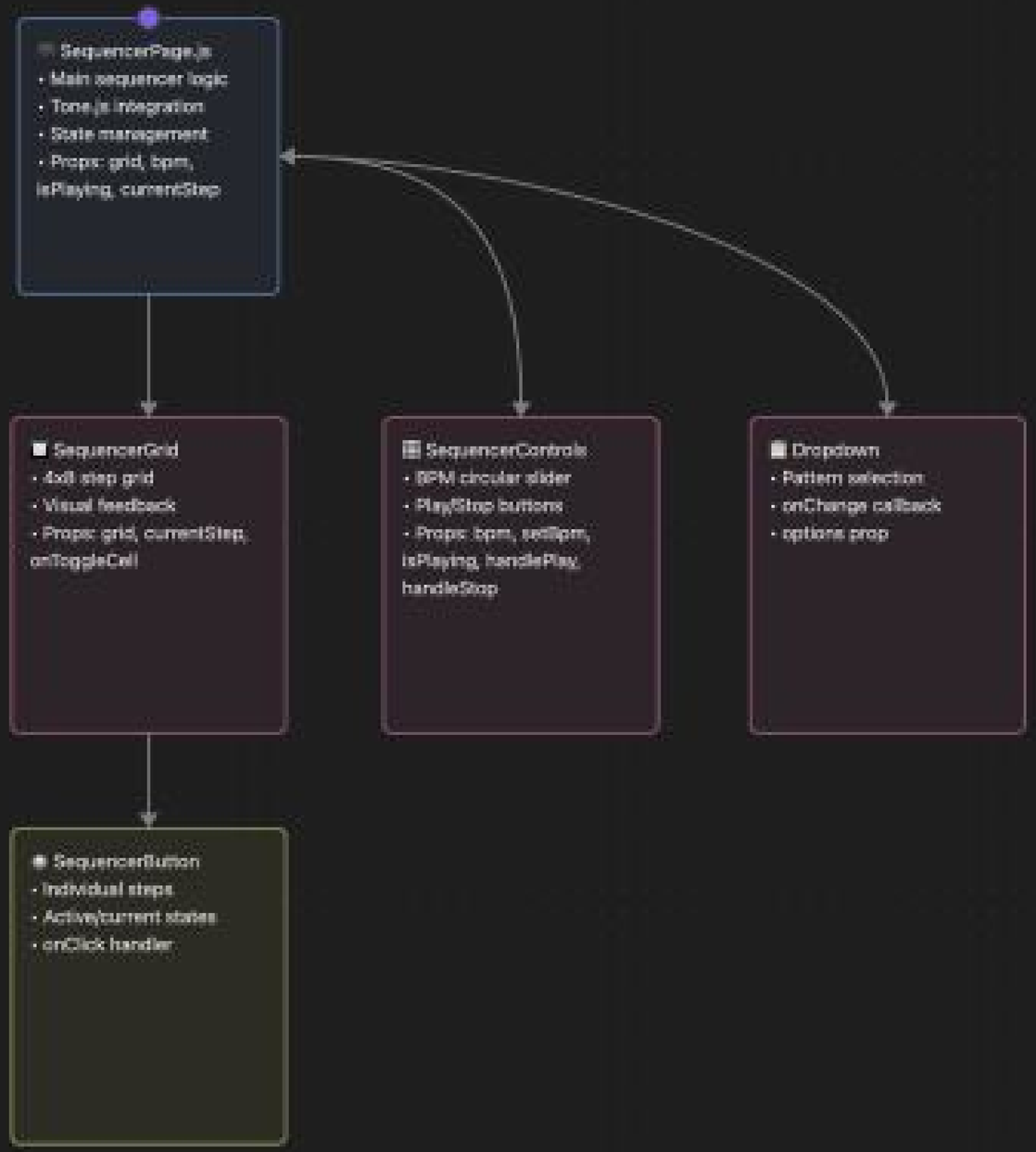
Creates one slider for each **ADSR** parameter

setEnv is a function factory.
Takes a parameter name (like "attack", "decay", etc.).
Returns a specialized function for that parameter

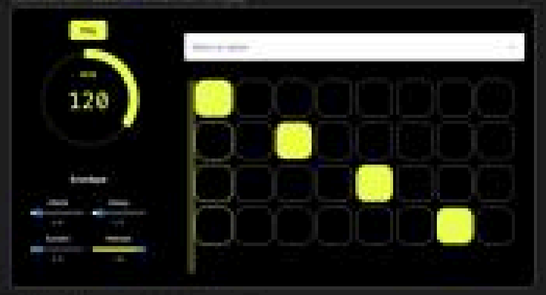
Loop through
{sliderDefs.map(({ key, min, max, step })

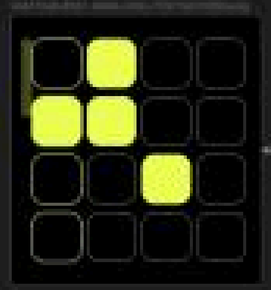

```
export default function SequencerPage() {
  // envelope - ADSR,
  const [envelope, setEnvelope] = useState({
    attack: 0.1,
    decay: 0.1,
    sustain: 0.1,
    release: 1.0,
  });

  function setEnv(key) {
    return function (value) {
      switch (key) {
        case "attack":
          setEnvelope(prev => ({ ...prev, attack: value }));
          break;
        case "decay":
          setEnvelope(prev => ({ ...prev, decay: value }));
          break;
        case "sustain":
          setEnvelope(prev => ({ ...prev, sustain: value }));
          break;
        case "release":
          setEnvelope(prev => ({ ...prev, release: value }));
          break;
        default:
          console.warn(`Unknown envelope parameter: ${key}`);
      }
    }
  }
```

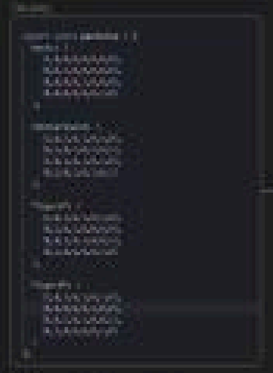


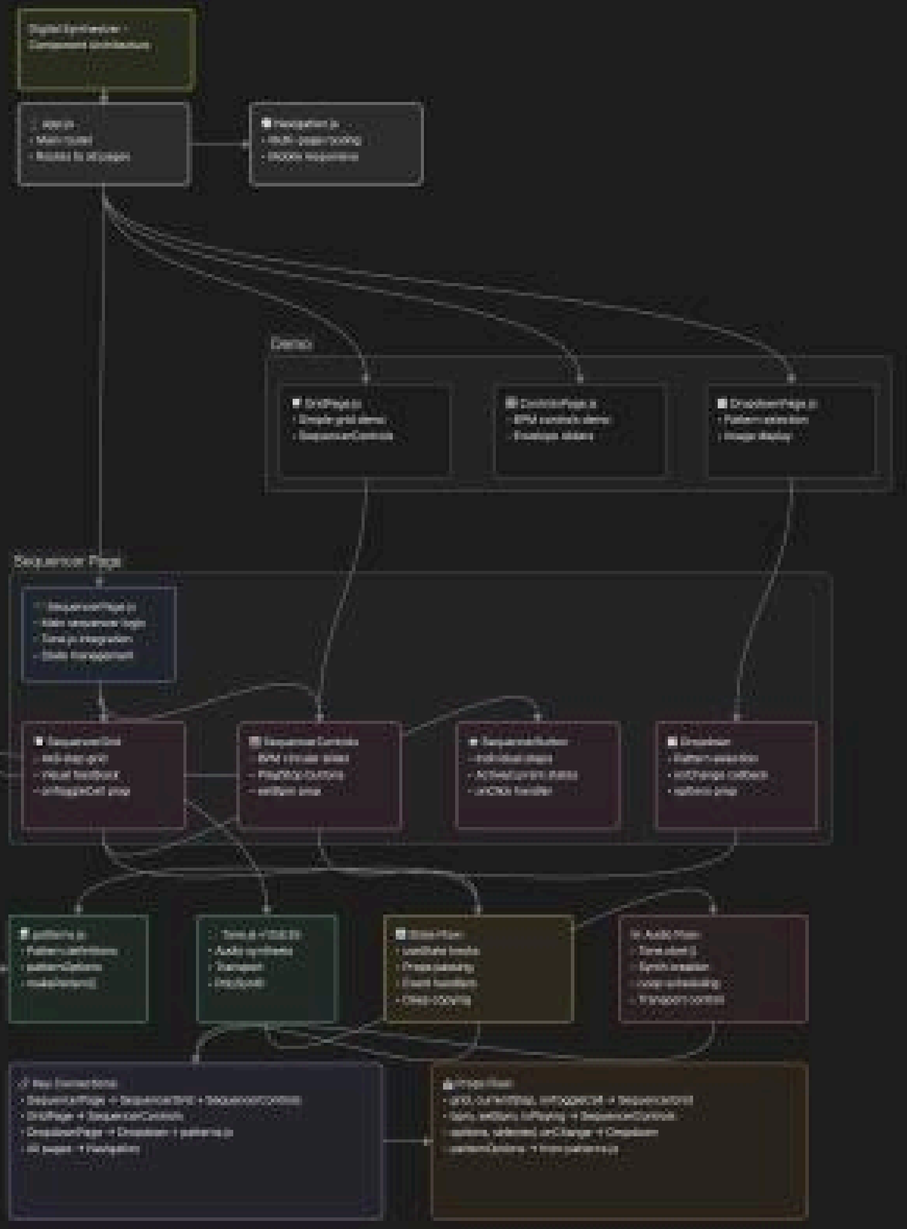
Putting it together


Play
BPM
120




Play
BPM
120




Demo
Sequencer Page

Next steps

What I wish I had done.

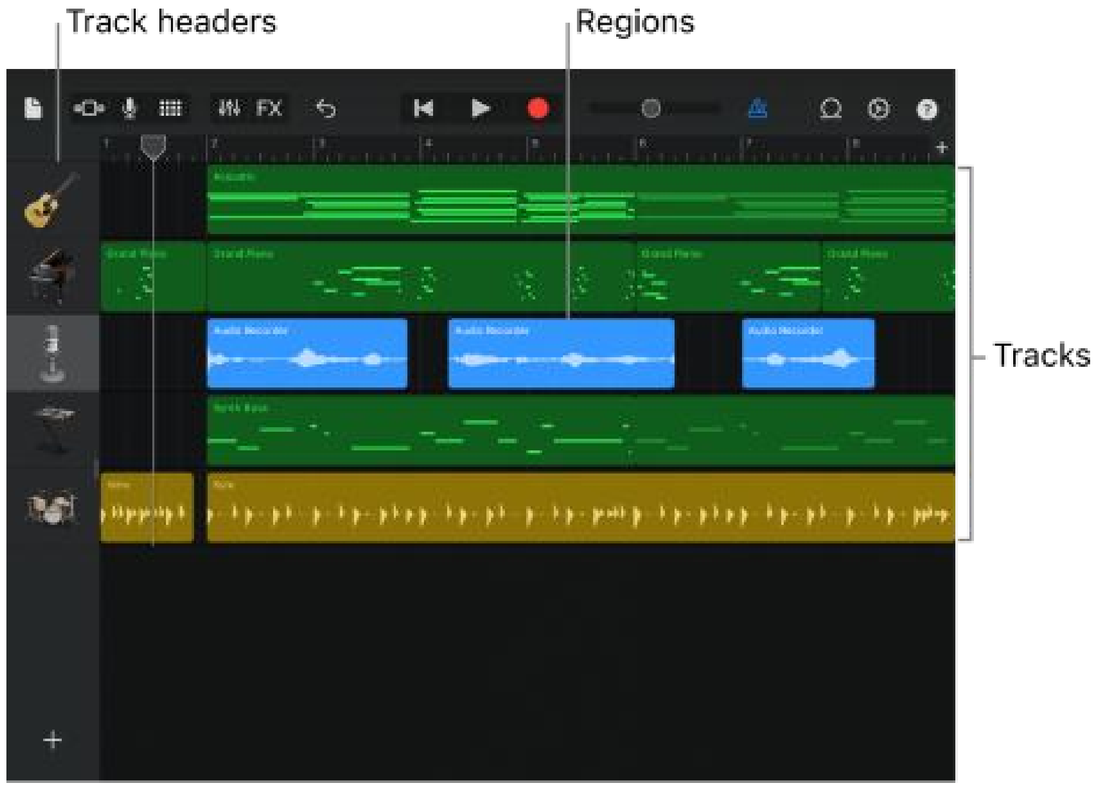
Right now the sequencer only plays one chord.
const CHORD_NOTES = ["Bb4", "D4", "F4", "A4"];.

It would be cool to be able to drag different chords and arrange how they are played.

I also wish I used different oscillator type

oscillator: { type: "sawtooth" },

Maybe connect it to keyboard or midi?



Challenges

- Tone.js updates
    - In Tone.js v13.8.25, the destination is **Tone.Master**, not **Tone.Destination.**
    - **I didn’t know that.**

```javascript
// ❌ Wrong (v13.8.25)
synth.connect(Tone.Destination);

// ✅ Correct (v13.8.25)
synth.connect(Tone.Master);
```

- Converting HTML to ReactJS wasn’t quite intuitive
- A lot of other functions in tone.js remains unexplored.
- I wanted to create an audio visualiser using **audiomotion-analyzer** but I ran out of time