

21ST OCT 2025 -

Designing a digital synth/sequencer

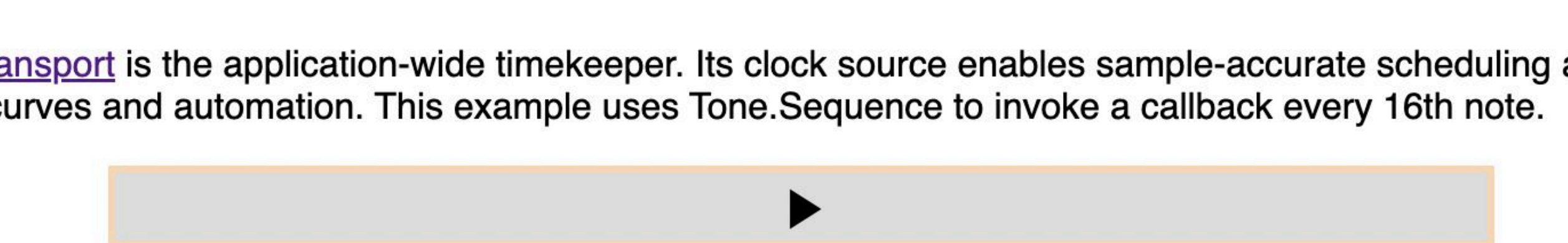
using ReactJS, tone.js, state managements, etc

ISSAC TING

Reference

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

Tone.Transport is the application-wide timekeeper. Its clock source enables sample-accurate scheduling as well as tempo-curves and automation. This example uses Tone.Sequence to invoke a callback every 16th note.



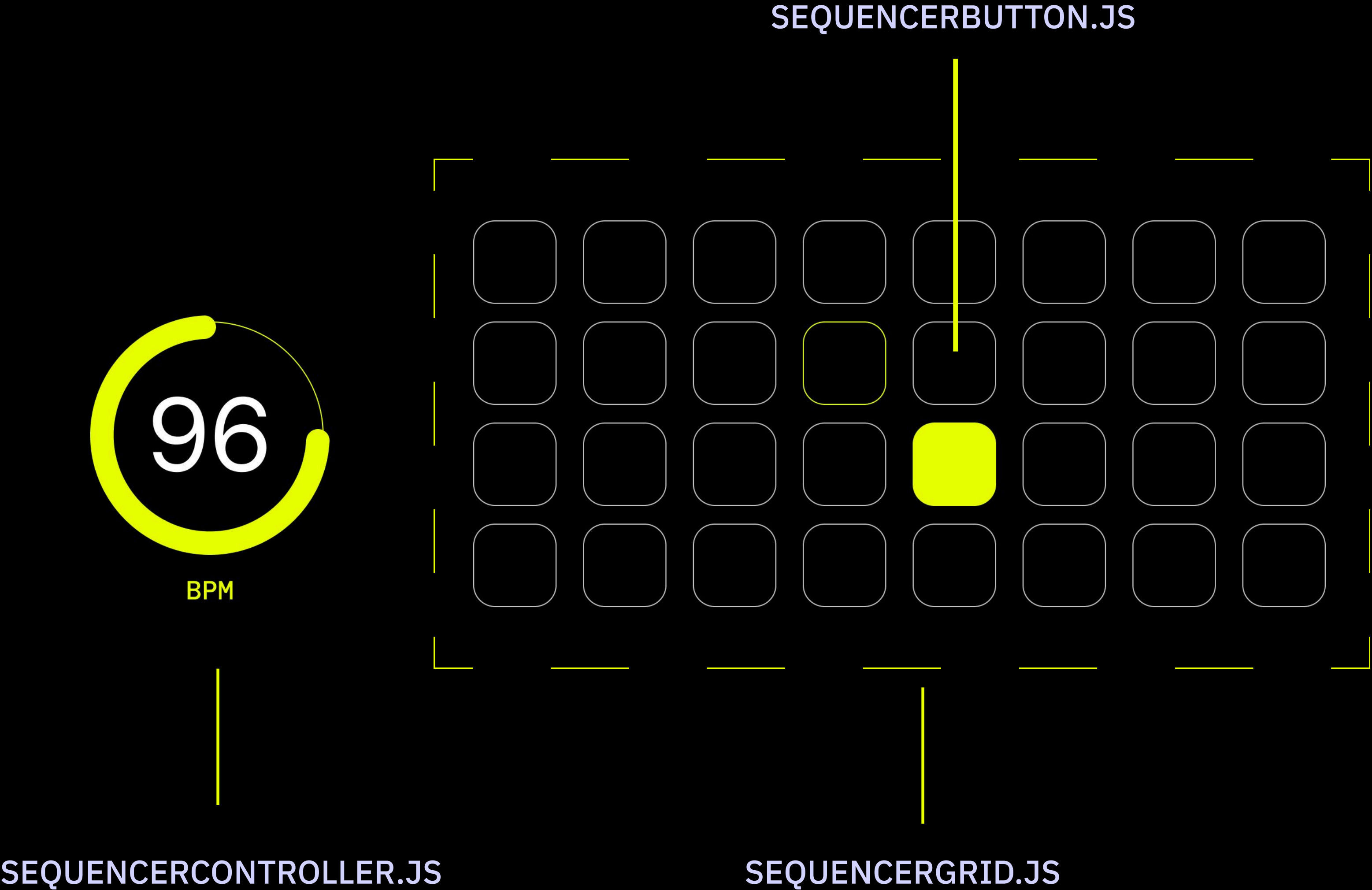
The diagram illustrates the internal structure of Tone.Transport. At the top is a horizontal bar representing the transport, with a play button icon in the center. Below this bar, the word "tempo" is on the left and "120 bpm" is on the right. A horizontal line with a black dot in the middle represents the tempo curve. Below the tempo line is a 4x16 grid of squares. The 10th square in the first row is highlighted in black, representing the 16th note in the sequence.

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.



BPM Controls Demo

This page demonstrates the BPM controls using a circular slider.

The slider uses the [@fseehawer/react-circular-slider](#) library.

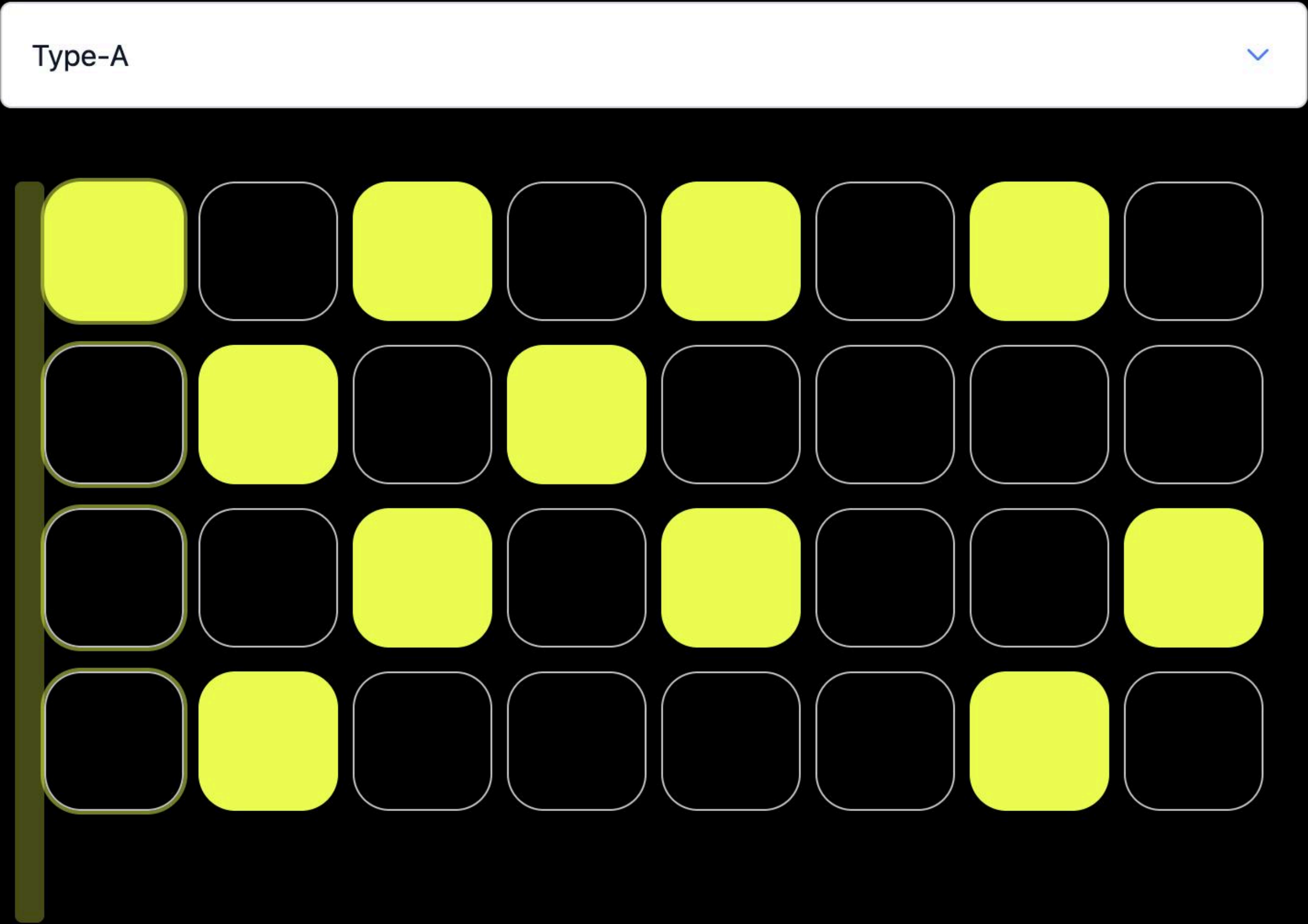
Use the play/stop button to control playback.



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>  
))}
```



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

```
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 })

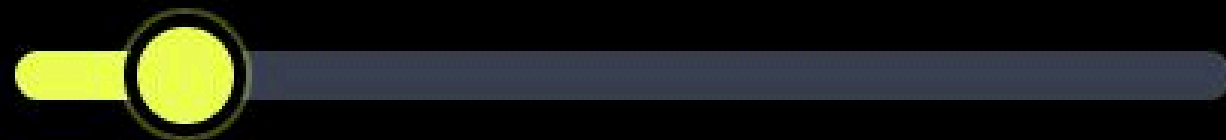
Attack 0.10



Decay 0.10



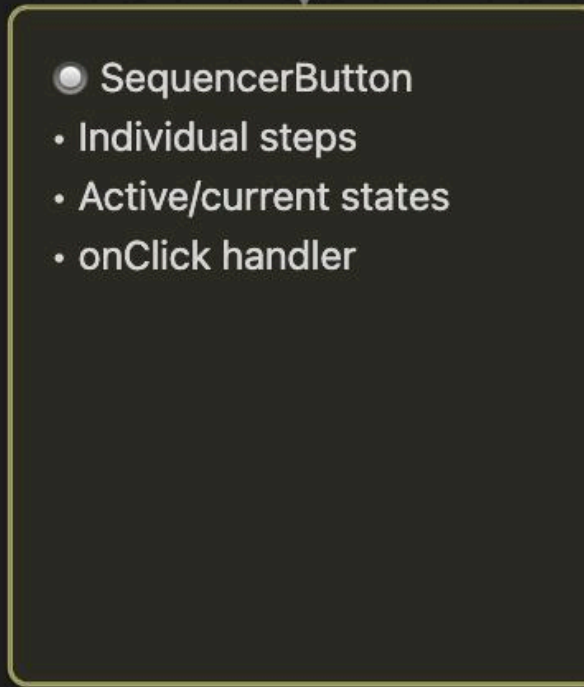
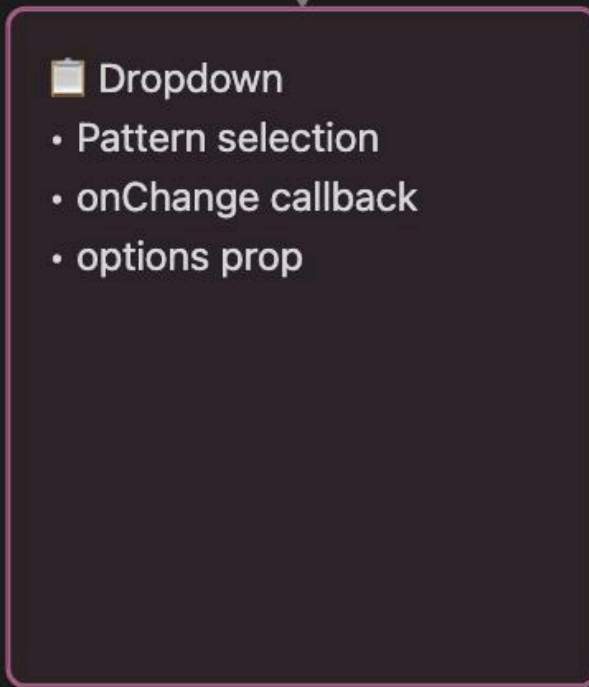
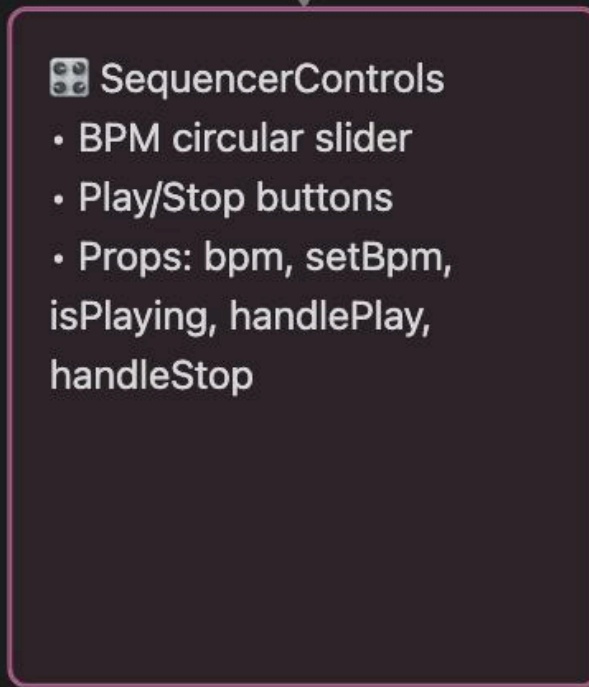
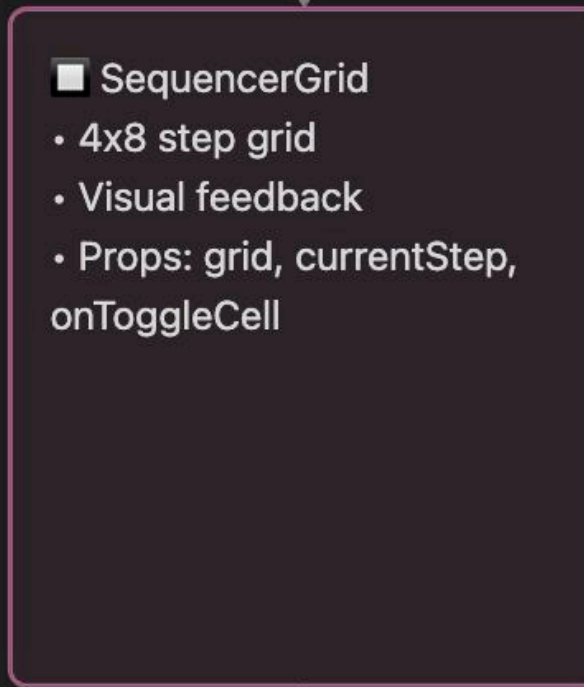
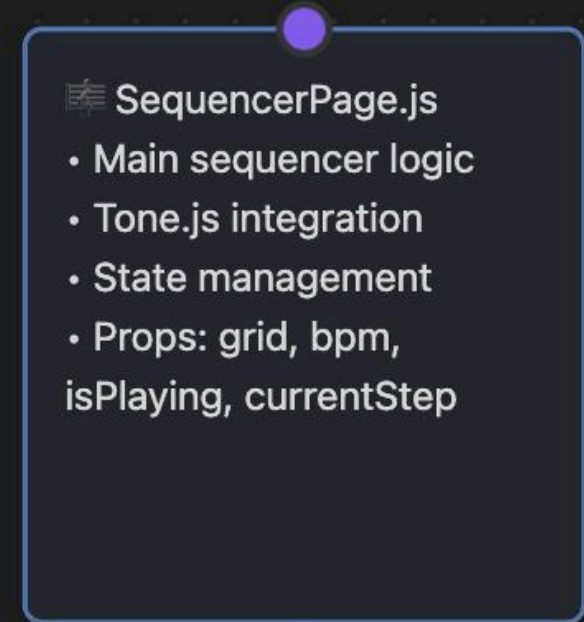
Sustain 0.10



Release 1.00

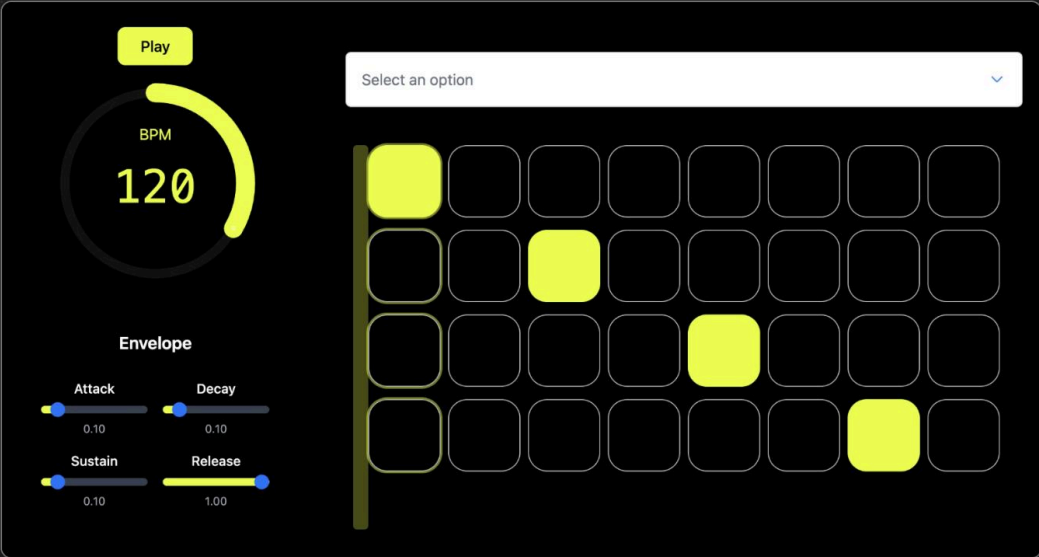


```
export default function SequencerPage() {  
  // envelope - ADSR,  
  const [envelope, setEnvelope] = useState({  
    attack: 0.1,  
    decay: 0.1,  
    sustain: 0.1,  
    release: 1.0,  
  });  
  
  function setEnv(key) {  
    return function (value) {  
      switch (key) {  
        case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

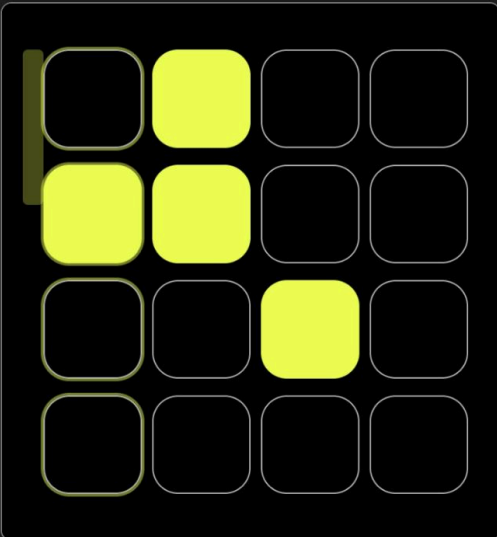


Putting it together

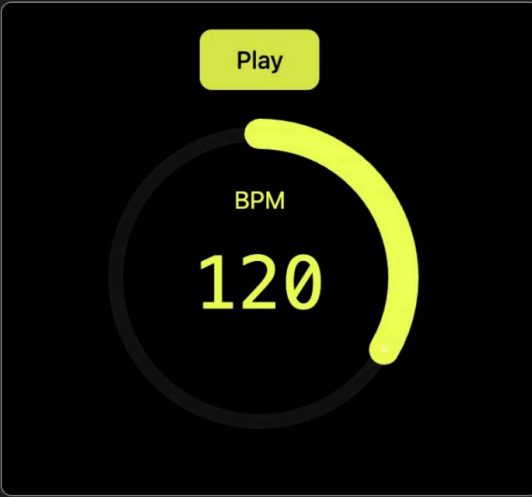
E8A9E246-B611-46A2-8DFC-72EFCB1F757E.jpeg



47A413A6-8587-4946-A290-F7EE7D0CE0BB.jpeg



35F47404-1472-4307-A4CE-137C2D478C23.jpeg



Tab.jpeg

```
export const patterns = {
  basic: [
    [1,0,0,0,0,0,0,0],
    [0,0,1,0,0,0,0,0],
    [0,0,0,1,0,0,0,0],
    [0,0,0,0,0,1,0,0]
  ],
  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]
  ],
  "Type-A": [
    [1,0,1,0,1,0,1,0],
    [0,1,0,1,0,0,0,0],
    [0,0,1,0,1,0,0,1],
    [0,1,0,0,0,0,1,0]
  ],
  "Type-B": [
    [1,0,1,0,1,0,1,0],
    [0,0,0,0,1,0,0,0],
    [0,0,1,0,1,0,1,1],
    [0,1,0,0,0,0,1,0]
  ]
}
```

Digital Synthesizer -
Component Architecture

App.js

- Main router
- Routes to all pages

Navigation.js

- Multi-page routing
- Mobile responsive

Demo

GridPage.js

- Simple grid demo
- SequencerControls

ControlsPage.js

- BPM controls demo
- Envelope sliders

DropdownPage.js

- Pattern selection
- Image display

Sequencer Page

SequencerPage.js

- Main sequencer logic
- Tone.js integration
- State management

SequencerGrid

- 4x8 step grid
- Visual feedback
- onToggleCell prop

SequencerControls

- BPM circular slider
- Play/Stop buttons
- setBpm prop

SequencerButton

- Individual steps
- Active/current states
- onClick handler

Dropdown

- Pattern selection
- onChange callback
- options prop

patterns.js

- Pattern definitions
- patternOptions
- makePattern()

Tone.js v13.8.25

- Audio synthesis
- Transport
- PolySynth

State Flow:

- useState hooks
- Props passing
- Event handlers
- Deep copying

Audio Flow:

- Tone.start()
- Synth creation
- Loop scheduling
- Transport control

Key Connections:

- SequencerPage → SequencerGrid + SequencerControls
- GridPage → SequencerControls
- DropdownPage → Dropdown + patterns.js
- All pages → Navigation

Props Flow:

- grid, currentStep, onToggleCell → SequencerGrid
- bpm, setBpm, isPlaying → SequencerControls
- options, selected, onChange → Dropdown
- patternOptions → from patterns.js

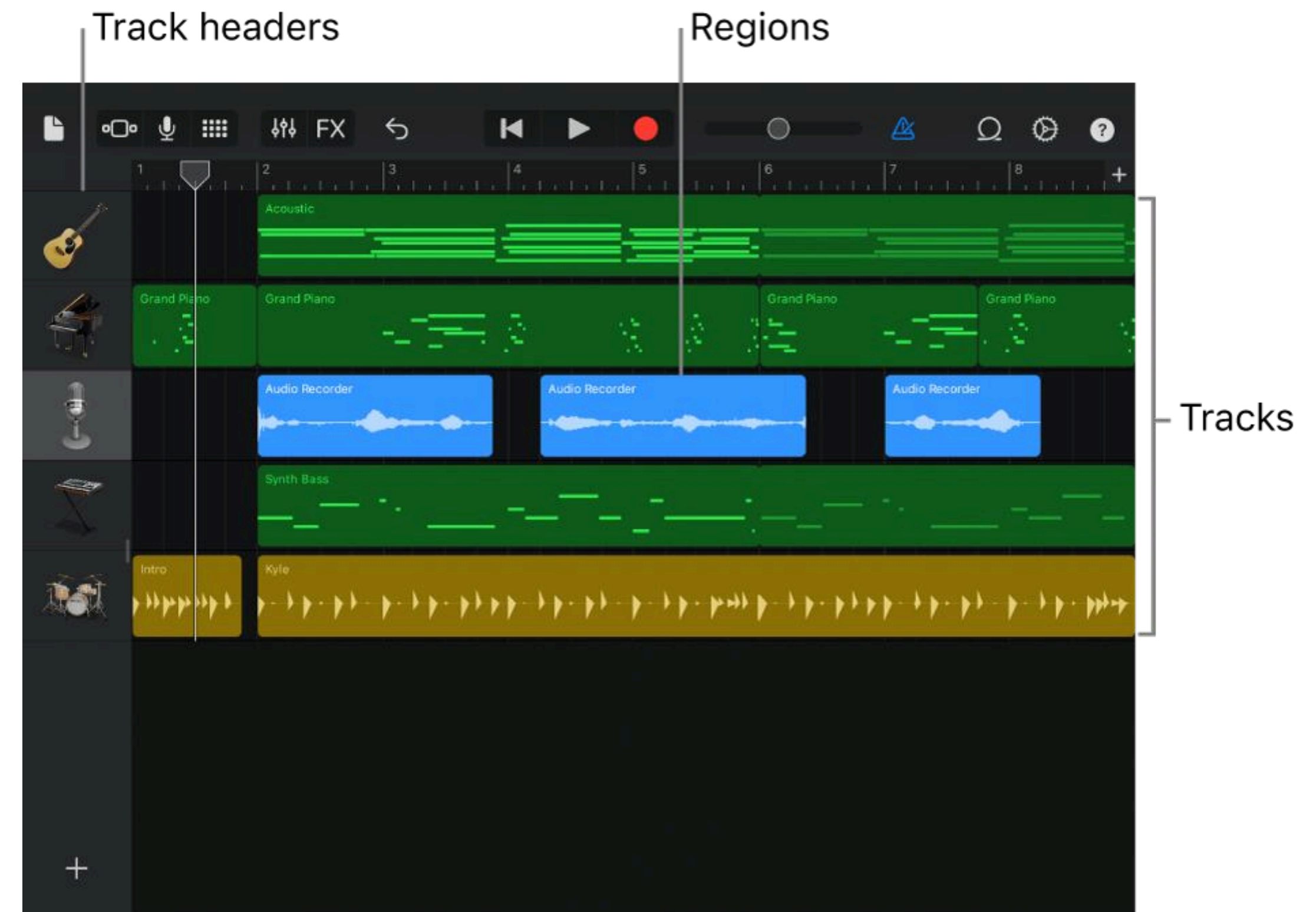
Next steps

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.



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