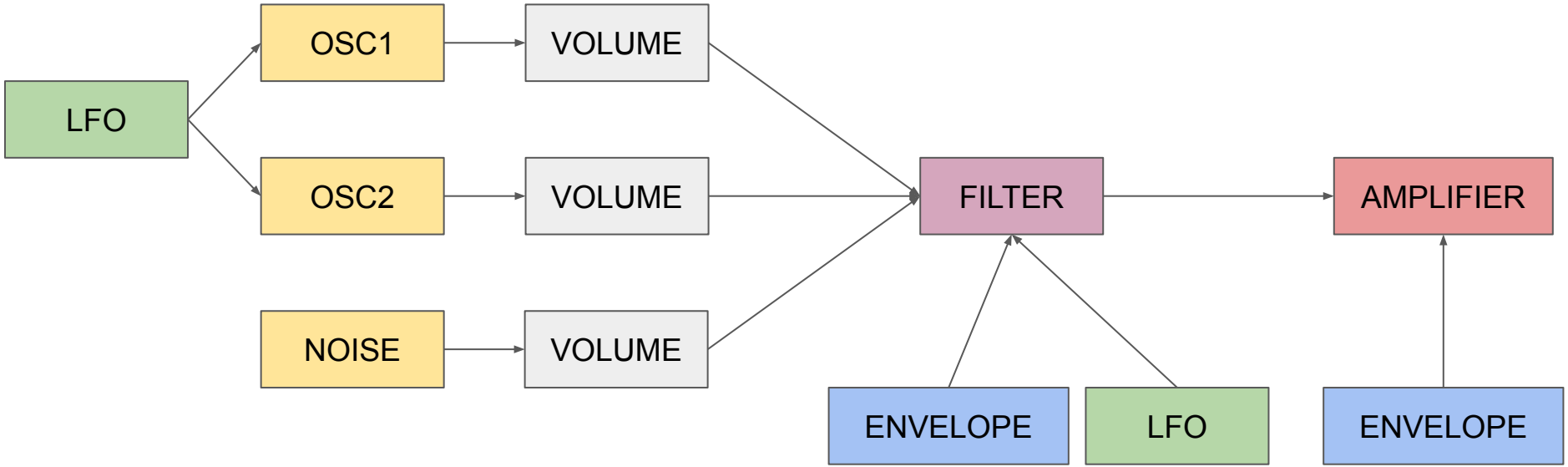
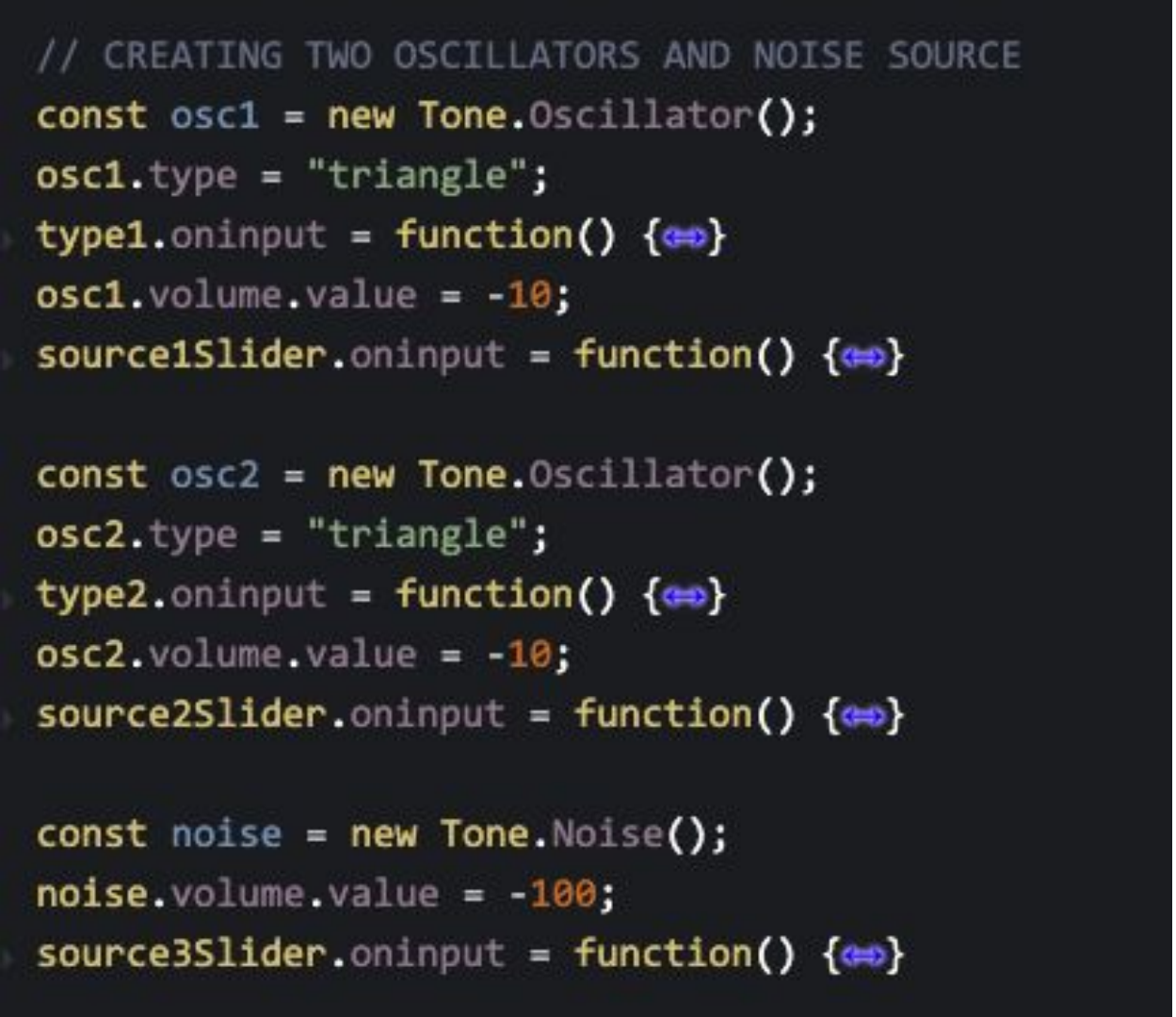
SUBTRACTIVE SYNTH ARCHITECTURE





CREATING SOUND SOURCES

EACH OSCILLATOR HAS ITS OWN TYPE, VOLUME AND FREQUENCY

```
//EQUAL TEMPERAMENT SYSTEM
const semitone = Math.pow(2, 1/12);
var c = 440 * Math.pow(semitone,3);
var notes = [];
for (i=0; i<12; i++){
  notes[i] = c * Math.pow(semitone, i);
}

// CREATING AMPLITUDE ENVELOPE AND IT'S NODE
const ampEnv = new Tone.Envelope({})
const ampEnvNode = new Tone.Gain();

ampEnv.attack = 0.001;
ampEnv.decay = 0.5;
ampEnv.sustain = 0;
ampEnv.release = 0.2;

att2Slider.oninput = function() {⟺}
dcy2Slider.oninput = function() {⟺}
stn2Slider.oninput = function() {⟺}
```

IMPLEMENTING EQUAL TEMPERAMENT SYSTEM

CREATING AMPLITUDE ENVELOPE

```
//CREATING LOWPASS FILTER AND ITS CONTROL SIGNAL
const fltEnv = new Tone.Envelope();
const mainSignal = new Tone.Signal();
const freqNode = new Tone.Gain();

fltEnv.attack = 0.001;
fltEnv.decay = 0;
fltEnv.sustain = 1;
fltEnv.release = 0.001;


att1Slider.oninput = function() {⇔}
dcy1Slider.oninput = function() {⇔}
stn1Slider.oninput = function() {⇔}
rls1Slider.oninput = function() {⇔}

const filter1 = new Tone.Filter();
filter1.type = "lowpass";
/*filter1.frequency.value = 12000;*/
filter1.Q.value = 0.7;

mainSignal.value = 12000
```

CREATING FILTER

```
//CREATING THE LFOs
const lfoCutoff = new Tone.LFO;
lfoCutoff.frequency.value = 0;
lfoCutoff.min = 0;
lfoCutoff.max = 0;

const vibrato = new Tone.Vibrato();
vibrato.frequency.value = 0;
vibrato.depth.value = 0;
lfoRateSlider.oninput = function() {
  vibrato.frequency.value = this.value/10;
  lfoCutoff.frequency.value = this.value/10;
}
lfoAmnt1Slider.oninput = function() {
  vibrato.depth.value = this.value/100;
}
lfoAmnt2Slider.oninput = function() {⟺}
```

CREATING LFOs

USER INTERFACE