```
// Any parameter that is a number can be defined as a function rather than a
static variable. This can be used to
// pass variables to functions. For example, the following uses the x position
of the mouse
// to control the frequency of the oscilator
osc(()=> mouse.x).out()

// Note: the above is the equivalent of writing the following:
// osc(function() {
//   return mouse.x
// }).out()

// mouse x and y controlling two different oscillators
osc(()=> mouse.x)
  .rotate(0.5)
  .modulate(osc(() => mouse.y))
  .out()

// modify the number of repetitions as a function of time.
gradient().repeat( () => 10*Math.sin(time*0.6) ).out()


// you can also define functions as variables and then reuse them
sides = () => Math.floor(10 * Math.sin(time * 0.6))
//
shape(sides)
  .diff(shape(sides, 0.1))
  .out()

// return a function from a function
scaledSides = scale => () => (Math.floor(scale * Math.sin(time * 0.6))+ scale +
3)
// use the function scaled sides, passing a different value in for scale each
time
shape(scaledSides(8))
  .diff(shape(scaledSides(4), 0.2))
  .out()


// Variables can also be arrays that represent patterns of values in time
shape([3, 8, 4]).out()

// by default, each number in the array takes the time of a single 'beat', at
60 beats per minute
// the speed of a pattern can be sped up or slowed down using fast()
shape([3, 8, 4].fast(3.0)).out() // speed up

shape([3, 8, 4].fast(0.4)).out() //slow down


// combining multiple patterns at multiple speeds
shape([3, 8, 4])
  .color(
    [-1, 1, -0.5].fast(0.5),
    0.4,
    [-1, -0.6, -0.8].fast(0.4)
  )
  .repeat([10, 100, 2, 3], 3)
  .out()


  // the global speed can be changed with speed or bpm

speed = 1

bpm = 30
```